New Computational Methods for Gene Analysis from Microarray Data

A thesis submitted to the Jadavpur University in partial fulfillment for the Degree of Doctor of Philosophy in Engineering

Shubhra Sankar Ray

Center for Soft Computing Research: A National Facility
Indian Statistical Institute
Kolkata-700108, INDIA

JADAVPUR UNIVERSITY

KOLKATA-700 032, INDIA

INDEX NO. 64/06/Engg.

1. Title of the thesis:
   New Computational Methods for Gene Analysis from Microarray Data

2. Name, Designation & Institution of the Supervisor/s:

   1. Prof. Sankar K. Pal
      Director
      Indian Statistical Institute
      Kolkata-700108, India

   2. Dr. Sanghamitra Bandyopadhyay
      Associate Professor
      Machine Intelligence Unit
      Indian Statistical Institute
      Kolkata-700108, India

3. List of publications:

   1. S. K. Pal, S. Bandyopadhyay, and S. S. Ray. Evolutionary Computation in Bioinformatics: A Review. *IEEE Transactions on Systems, Man, and Cybernetics, Part-C*, 36(5):601–615, 2006.

   2. S. S. Ray, S. Bandyopadhyay, P. Mitra, and S. K. Pal. Bioinformatics in Neurocomputing Framework. *IEE Proc. Circuits Devices & Systems*, 152:556–564, 2005.

   3. S. S. Ray, S. Bandyopadhyay, and S. K. Pal. Genetic Operators for Combinatorial Optimization in TSP and Microarray Gene Ordering. *Applied Intelligence*, 26(3):1830–195, 2007.

   4. S. S. Ray, S. Bandyopadhyay, and S. K. Pal. Gene Ordering in Partitive Clustering using Microarray Expressions. *Journal of Biosciences*, 32(5):1019–1025, 2007.

   5. S. S. Ray, S. Bandyopadhyay, and S. K. Pal. Dynamic Range Based Distance Measure for Microarray Expressions and a Fast Gene Ordering

Algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part-B*, 37(3):742–749, 2007.

6. S. S. Ray, S. Bandyopadhyay, and S. K. Pal. Combining Multi-Source Information through Functional Annotation Based Weighting: Gene Function Prediction in Yeast. *IEEE Transactions on Biomedical Engineering*, 2008 (under revision).

7. S. S. Ray, S. Bandyopadhyay, P. Mitra, and S. K. Pal. Bioinformatics in Neurocomputing Framework. *The International Conference on Computers and Devices for Communication, CODEC-04*, page 94, January 1-3, Kolkata, India, 2004.

8. S. S. Ray, S. Bandyopadhyay, and S. K. Pal. New Operators of Genetic Algorithms for Traveling Salesman Problem. *The 17th International Conference on Pattern Recognition, ICPR-04* volume 2, pages 497–500, Cambridge, UK, 23-26 August 2004.

9. S. S. Ray, S. Bandyopadhyay, and S. K. Pal. New Genetic Operators for Solving TSP: Application to Microarray Gene Ordering. *The First International Conference on Pattern Recognition and Machine Intelligence, PReMI 2005*, pages 617-622, December, Kolkata, India, 2005.

10. S. S. Ray, S. Bandyopadhyay, and S. K. Pal. Gene Ordering in Partitive Clustering using Microarray Expressions. *International Conference on Bioinformatics, INCOB 2006*, page 33, 18-20 December, New Delhi, 2006.

11. S. S. Ray, S. Bandyopadhyay, and S. K. Pal. New Distance Measure for Microarray Gene Expressions using Linear Dynamic Range of Photo Multiplier Tube. *International Conference on Computing: Theory and Applications, Kolkata, India*, pages 337–341, 2007.

12. S. S. Ray, S. Bandyopadhyay, and S. K. Pal. Predicting Gene Function in Yeast through Adaptive Weighting of Multi-Source Information. *The Eighth International Conference on Systems Biology, ICSB 2007*, online proceedings, no. H03, October 1-6, Long Beach, California, USA, 2007.

4. List of Patents: Nil

5. List of Presentations in National/International:

   1. Bioinformatics in Neurocomputing Framework. *The International Conference on Computers and Devices for Communication, CODEC-04*, January 1-3, Kolkata, India, 2004.
   2. New Genetic Operators for Solving TSP: Application to Microarray Gene Ordering. *The First International Conference on Pattern Recognition and Machine Intelligence, PReMI 2005*, December, Kolkata, India, 2005.
   3. New Distance Measure for Microarray Gene Expressions using Linear Dynamic Range of Photo Multiplier Tube. *International Conference on Computing: Theory and Applications, ICCTA, March 5-7, Kolkata, India*, 2007.

Dedicated to my beloved father

CERTIFICATE

This is to certify that the thesis entitled “New Computational Methods for Gene Analysis from Microarray Data” submitted by Mr. Shubhra Sankar Ray, who got his name registered on “08.09.06” for the award of Ph. D. (in engineering) degree of Jadavpur University, is absolutely based upon his own work under the supervision of Prof. Sankar K. Pal and Dr. Sanghamitra Bandyopadhyay and that neither his thesis nor any part of the thesis has been submitted for any degree/diploma or any other academic award anywhere before.

Prof. Sankar K. Pal
Director
Indian Statistical Institute
Kolkata-700108, India

Dr. Sanghamitra Bandyopadhyay
Machine Intelligence Unit
Indian Statistical Institute
Kolkata-700108, India

ACKNOWLEDGEMENTS

First of all I express my gratitude to my supervisors, *Prof. (Dr.) Sankar K. Pal and Dr. Sanghamitra Bandyopadhyay*, without whose guidance, encouragement and affection this thesis could not have been completed. They introduced me to the interesting field of *Bioinformatics* and continued helping me in every stage of my endeavors to delve into its depths. Thanks are due to them for their kind permission to include the joint research work in this thesis.

Word seems to be insufficient to thank Dr. Pabitra Mitra, Department of Computer Science, Indian Institute of Technology, Kharagpur, who guided me like a path finder to start my research carrier. I must also thank Prof. C. A. Murthy, Prof. S. Mitra, Dr. Rajat K. De, Dr. D. P. Mandal and Dr. B. Uma Sankar for their affection, advice and encouragement during this work. I owe a lot to my colleagues and friends, Mr. B. L. Narayan, Mr. Suman Saha Mr. Debashis Sen, Dr. Kuntal Ghosh and Dr. Pradipto Maji for providing me a cooperative research environment. I am thankful to Mr. Surya Bhattacharya, Mr. Bikram Roy, Mr. Joydeb Gupta, Mrs. Neyoti Das, Mr. Sujit Basak, Mr. Chandrasekhar Das and Mr. Sachchidananda Mahato for helping me many ways with all machines, softwares and official work related problems.

I would like to acknowledge Dr. Martin Mnsterktter, CYGD coordinator at MIPS, for fixing the bugs in downloading functional categories and Dr. Maria C. Costanzo, Senior Scientific Curator of Saccharomyces Genome Database, for mapping the gene names from Yeast GO-Slim process annotations to ORFs. Supports of the Dept. of Science and Technology, Govt. of India to the Center for Soft Computing Research through its IRHPA scheme and the Council of Scientific and Industrial Research (CSIR), New Delhi, to the grant no. 22(0346)/02/EMR-II are also acknowledged. I express my sincere thanks to the authorities of ISI for the facilities extended to carry out my research work.

Finally, I shall forever remain indebted to my family members for their constant help, encouragement and love throughout my academic carrier, without whose co-operation it would have been impossible for me to have done all this.

ISI, Kolkata
April, 2008. *Shubhra Sankar Ray*

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

Over the past few decades, major advances in the field of molecular biology, coupled with advances in genomic technologies, have led to an explosive growth in the biological information generated by the scientific community. This deluge of genomic information has, in turn, resulted in the introduction of bioinformatics, computational genomics and proteomics, large-scale analysis of complete genomes and the proteins that they encode for relating specific genes to diseases, specialized tools to view and analyze the data, and computerized databases to store, organize and index the data.

Bioinformatics is an interdisciplinary field involving biology, computer science, mathematics and statistics to analyze biological sequence data, genome content and arrangement, and to predict the function and structure of macromolecules [77]. It can be viewed as *the use of computational methods to make biological discoveries* [12]. The ultimate goal of the field is to enable the discovery of new biological insights as well as to create a global perspective from which unifying principles in biology can be derived [3]. There are three important sub-disciplines within bioinformatics:

- Development of new algorithms and models to assess different relationships among the members of a large biological data set in a way that allows researchers to access existing information and to submit new information as they are produced

- Analysis and interpretation of various types of data including DNA, RNA, mRNA, amino acid sequences, protein domains, and protein structures; and
- Development and implementation of tools that enable efficient access and management of different types of information.

The most powerful and commonly used technique for efficiently managing and analyzing huge amount of genomic data is that involving microarray, which has enabled the monitoring of the expression levels of more than thousands of genes simultaneously. A microarray is typically a glass slide, onto which thousands of genes are attached at fixed locations (spots). It allows a massive number of characterized samples to be assayed in parallel, saving time and money, and sometimes allowing the elucidation of pathways and interactions which ordinarily would not have been discovered using traditional methods. They require little labelled probe and small amount of space, deliver data quickly and relatively easy to use, and are stable for long periods of time. Microarrays can be manufactured in several different ways, and their flexibility makes them appropriate for all types of research needs [67]. By performing biological experiments gene expression levels are obtained from microarray [105]. Gene expression is the process by which a gene's coded information is converted into the structures present and operating in the cell. Expressed genes include those that are transcribed into mRNA and gene expression involves the study of the level of mRNA in cells under different conditions. A key step in the analysis of microarray gene expression data is the identification of groups of genes that manifest similar expression patterns. This translates to the algorithmic problem of clustering and ordering of gene expression data.

The present thesis provides some new results of investigation concerning clustering and ordering of gene expression microarray data, development of new distance measures for genes using an microarray experiment specific normalization factor, and prediction of biological functions of unclassified genes by integrating gene expression information with other biological data sources. Before we describe the scope of the thesis in Section 1.7, an overview of the basic concepts of cell biology, involving DNA, RNA, mRNA and protein is first presented in Section 1.2. The basics of gene expression along with its relation with cell biology are explained in Section 1.3. Methodologies for preparing microarray are mentioned in Section 1.4. In Section 1.6 various computational methods available to analyze and interpret the microarray data are explained. Finally, the outline of the thesis and conclusions including some future research directions are presented.

1.2 Basic Concepts of Cell Biology

DNA (deoxyribonucleic acid) and proteins are biological macromolecules built as long linear chains of chemical components. DNA strand consists of a large sequence of nucleotides, or bases. For example, there are more than 3 billion bases in human DNA sequences. DNA plays a fundamental role in different bio-chemical processes of living organisms in two respects. First it contains the templates for the synthesis of proteins, which are essential molecules for any organism [99]. The second role in which DNA is essential to life is as a medium to transmit hereditary information (namely the building plans for proteins) from generation to generation. Proteins are responsible for structural behavior.

The units of DNA are called nucleotides. One nucleotide consists of one nitrogen base, one sugar molecule (deoxyribose) and one phosphate. Four nitrogen bases are denoted by one of the letters A (adenine), C (cytosine), G (guanine) and T (thymine). A linear chain of DNA is paired to a complementary strand. The complementary property stems from the ability of the nucleotides to establish specific pairs (A-T and G-C). The pair of complementary strands then forms the double helix that was first suggested by Watson and Crick in 1953. Each strand therefore carries the entire information and the biochemical machinery ensures that the information can be copied over and over again even when the “original” molecule has long since vanished.

A gene is primarily made up of ‘sequence of triplets’ of the nucleotides (exons). Introns (non coding sequence) may also be present within gene. Not all portions of the DNA sequences are coding. Coding zone indicates that it is a template for a protein. As an example, for the human genome only 3%-5% of the sequence are coding, i.e., they constitute the gene. The promoter is a region before each gene in the DNA that serves as an indication to the cellular mechanism that a gene is ahead. For example, the codon AUG is a protein which codes for methionine and signals the start of a gene. Promoters are key regulatory sequences that are necessary for the initiation of transcription. Transcription is a process in which ribonucleic acid (RNA) is formed from a gene and through translation aminoacids are formed from RNA. There are sequences of nucleotides within the DNA that are spliced out progressively in the process of transcription and translation. A comprehensive survey of the research done in this field is in [123]. In brief, the DNA consists of three types of non-coding sequences.

1. Intergenic regions: Regions between genes that are ignored during the process of transcription
2. Intragenic regions (or Introns): Regions within the genes that are spliced out from the transcribed RNA to yield the building blocks (Exons) of the genes.
3. Pseudogenes: Genes that are transcribed into the RNA and stay there, without being translated, due to the action of a nucleotide sequence.

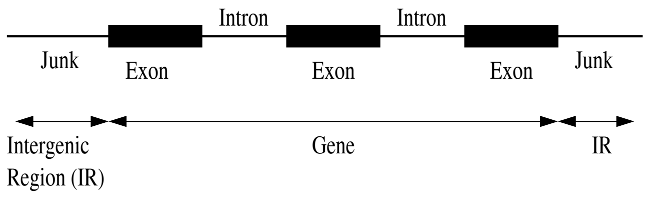


Figure 1.1: Various parts of a DNA

Proteins are polypeptides, formed within cells as a linear chain of amino acids [99]. Amino acid molecules bond with each other by eliminating water molecule and form peptides. 20 different amino acids (or "residues") are available, which are denoted by 20 different letters of the alphabet. Each of the 20 amino acids is coded by one or more triplets (or codons) of the nucleotides making up the DNA. Based on the genetic code the linear string of DNA is translated into a linear string of amino acids, i.e., a protein via mRNA (messenger RNA). For example, the DNA sequence GAACTACACACGTGTAAC codes for the amino acid sequence ELHTCN (shown in Fig. 1.2).

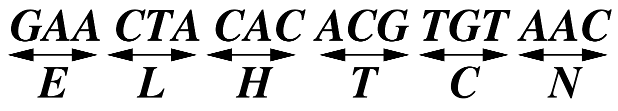


Figure 1.2: Coding of amino acid sequence from DNA sequence

Identical protein sequences result in identical 3-D structures. So it follows that similar sequences may result in similar structures, and this is usually the case. The converse, however, is not true: identical 3-D structures do not necessarily indicate identical sequences. It is because of this that there is a distinction between "homology" and "similarity". There are examples of proteins in the databases that have nearly identical 3-D structures, and are therefore homologous, but do not exhibit significant (or detectable) sequence similarity. Pairwise comparisons do not readily show positions that are conserved among a whole set of sequences and tend to miss subtle similarities that become visible when observed simultaneously among many sequences. Thus one wants to simultaneously compare several sequences.

Structural genomics is the prediction of 3-dimensional structure of a protein from the primary amino acid sequence [23]. This is one of the most challenging tasks in bioinformatics. The five levels of protein structure are described below. Four of them are illustrated in Figure 1.3.

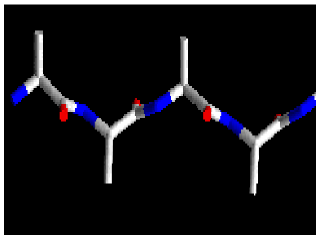
a) Primary structure is the sequence of amino acids that compose the protein.

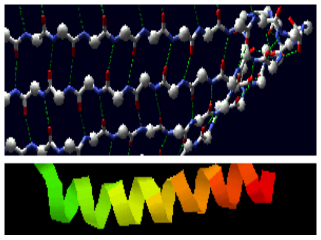
b) The secondary structure of a protein is the spatial arrangement of the atoms constituting the main protein backbone. Linus Pauling was the first to develop a hypothesis for different potential protein secondary structures. He developed the α-helix structure and later the β-sheet structure for different proteins. An α-helix is a spiral arrangement of the protein backbone in the form of a helix with hydrogen bonding between side-chains. The β-sheets consist of parallel or antiparallel strands of amino acids linked to adjacent strands by hydrogen bonding. Collagen is an example of a protein with β-sheets serving as its secondary structure.

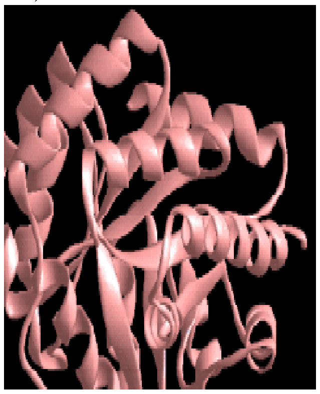
c) The Super-secondary structure (or motif) is the local folding patterns built up from particular secondary structures. For example the EF-hand motif consists of an α-helix, followed by a turn, followed by another α-helix.

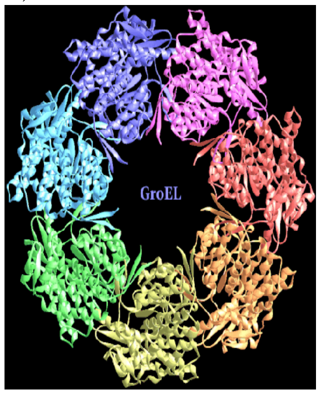
d) Tertiary structure is formed by packing secondary structural elements linked by loops and turns into one or several compact globular units called domains i.e., the folding of the entire protein chain.

e) Final protein may contain several protein subunits arranged in a quaternary structure.


a) Primary structure


b) Secondary structure


c) Tertiary structure



d) Quaternary structure

Figure 1.3: Different levels of protein structures

1.3 Basics of Gene Expression

An important and interesting question in biology, regarding the variation of gene expression levels is how genes are regulated. Virtually all cell function is carried out by proteins. So a readout of what and how many proteins are in a cell, at a particular moment, gives us great deal of information about the state of the cell and how the genes are expressed in that cell. It was discovered relatively early on in molecular biology that the abundance and distribution of proteins in cells are correlated to a large extent to the levels of mRNA (messenger RNA) [19]. Since almost all cells in a particular organism have an identical genome, differences in gene expression and not the genome content are responsible for cell differentiation during the life of the organism. For gene regulation an important role is played by a type of proteins called transcription factors [99]. The transcription factors bind to specific parts of the DNA, called transcription factor binding sites (i.e., specific, relatively short combinations of A, T, C or G), which are located in promoter regions. Specific promoters are associated with particular genes and are generally not too far from the respective genes, though some regulatory effects can be located as far as 30,000 bases away, which makes the definition of the promoter difficult. Transcription factors control gene expression by binding to the gene's promoter and either activating (switching on) the gene or repressing it (switching it off). Transcription factors are gene products themselves, and therefore in turn can be controlled by other transcription factors. Transcription factors can control many genes, and some (probably most) genes are controlled by combinations of transcription factors. Feedback loops are possible. Therefore we can talk about gene regulation networks. Conventional wisdom is that gene products that interact with each other are more likely to have similar expression profiles than if they do not [69].

1.4 Basics of Microarray

Various methods are developed for detecting and quantifying the amount of mRNA or gene expression level. The methods take advantage of the sequence complimentarity of DNA. The key observation was that single stranded DNA binds strongly to nitrocellulose membranes which prevents strands from reassociating with each other but permits the hybridization to complementary RNA [44]. This led to blotting methods, the first of which combined filter hybridization with gel separation of restriction digests [104]. The blotting methods are serial in nature and the mRNA is measured one at a time. On the other hand, DNA microarrays allow one to interrogate the mRNA population expressed by thousands of genes at once rather than serially as in the blotting methods. Another distinction between DNA microarrays and blotting methods is that the use of impermeable rigid substance such as glass to bind the DNA sequences in microarrays is practically advantageous over porous membranes and gel pads in blotting methods.

The two basic types of commonly used DNA microarrays are spotted arrays and oligonucleotide arrays. In the spotted array methods [31] a large number of cDNAs are prepared from a cDNA library and then spotted onto a glass slide by a robot. Each cDNA corresponds to one gene or one exon in the genome of length 100-1000 bp and also referred as a probe. Each probe corresponds to a particular spot in a microarray slide. In the course of hybridization, RNA from experimental samples (taken at selected times during the process) is labelled during reverse transcription with the red-fluorescent dye Cy5 and is mixed with a reference sample (cDNA) labelled in parallel with the green-fluorescent dye Cy3 [31]. After hybridization and appropriate washing steps, the arrays are scanned to produce images and the images are further processed by an image analysis program to produce measured red and green foreground and background intensities for each spot on each array. Before the gene expression profiles of the RNA samples can be analyzed and interpreted, the red and green intensities must be normalized (discussed in Section 1.6.2) relative to one another so that the red/green ratios for all target elements are as far as possible an unbiased representation of relative expression. If R (red) and G (green) are the spot-specific, quantified, fluorescent intensities of the target and reference expression signals respectively, relative gene expression is defined as the log ratio $M = log_2 \frac{R}{G}$. For microarray data table each cell represents the measured Cy5/Cy3 fluorescence ratio (M value or $\frac{R}{G}$ value) at the corresponding target element [31] obtained from the gene under that experimental condition. All $\frac{R}{G}$ values are log transformed (base 2 for simplicity) to treat inductions or repressions of identical magnitude but with opposite sign.

In the oligonucleotide arrays [66], each gene or EST is represented by multiple probes of length 20 bp. These probes are synthesized base by base and are placed in hundreds of thousands of different positions on a glass plate, using photolithography. As in the cDNA microarray RNA from experimental samples is hybridized with a reference sample (probe). The arrays are then scanned and the quantitative flourescence image along with the known position of the probes is used to asses whether a gene or EST is present and its abundance. In the oligonucleotide arrays the flourescence image is an absolute measure of the abundance of mRNA of a sample.

1.5 Drawbacks of Microarray Technology

Microarray technology is a very powerful tool for molecular biology, but it has certain drawbacks that limit the effectiveness of microarray experiments. First, biological variations, artifacts of preparing samples for analysis, inconsistencies of array chip printing, all contribute to low signal-to-noise ratio in microarray technology, which frequently results in inconsistent results. The limitations can be minimized by performing replicate experiments and using analysis methods that can take into account the estimated noise of the system. Another issue is that, measurements of mRNA levels can not be used as a direct indicator of corresponding protein activity due to complicated multi-step post-translational modifications. Although it limits the interpretation of microarray results, it is still possible to use changes in expression levels as downstream indicators of biological processes of the cell.

Correct functional assignments of gene and other related annotations are also of great importance for analysis and interpretation of results of microarray data. Improper probe annotations can be a result of various uncertainties. Mismatches of clones from cDNA libraries used for spotted microarray construction can produce a significant amount of error in identification of these probes. Therefore, availability of the most recent annotation information is crucial for the correct analysis of gene expression data.

1.6 Tasks Involved in the Analysis of Gene Expression Data

Computational methods involving gene expression microarray data play a major role in attempts to discover gene's functions, pathways and networks from various observations. Many unanswered and important questions could potentially be answered by correctly selecting, assembling, analyzing, and interpreting microarray data. Functional and pathway relation among genes, whose protein sequence similarity is negligible, can be discovered through microarray by measuring the expression levels of thousands of genes at the same time. Inferring a genes function from gene expression data obtained by DNA microarray is considered as one of the most challenging problems in the field of bioinfomatics [1].

Prior to microarray data analysis, several pre-processing steps are required. First, scanned images are quantified to determine the signal intensity of each spot. This is usually done by a software specially designed for a specific microarray platform on which the experiment was performed. Second, normalization of microarray data is performed to remove dye-related differences between two channels (Cy5 and Cy3) and various slide-specific artifacts that can exist between different microarray. After pre-processing, the normalized microarray data can be analyzed using statistical and pattern recognition methods like clustering, ordering, classification etc. The different tasks related with microarray data analysis are missing value prediction, normalization, clustering, ordering, and interpretation of results. In the subsequent sections, we will discuss about various pre-processing and analyzing methods for microarray in brief and missing value prediction using LSimpute, between-slide normalization, average linkage, k-means, and self-organizing map in details, as they are used for analysis and comparison with our proposed methods in different chapters of the thesis.

1.6.1 Missing Value Prediction

Estimating missing values is a classical problem in statistics, and iterative algorithms based on the EM algorithm are widely used. An implementation of the EM algorithm for missing value estimation is described by Johnson and Wichern [53]. Microarray experiments generate data sets with information on the expression levels of thousands of genes in a set of biological samples. Unfortunately, such experiments often produce multiple missing expression values, normally due to various experimental problems. Typical problems include spotting problems, scratches on the slide, dust or hybridization failures. This in turn results in values missing from the gene expression matrix. Thus in every microarray related investigation, one needs to determine how to treat missing values. Repeating the experiment is often not a realistic option, for economic reasons or because of limitations in available biological material. The data produced by microarray experiments can be analyzed by various methods in order to visualize the information inherent in the data. Analysis results obtained using clustering algorithms, such as hierarchical clustering, K-means clustering and self-organizing maps, or data dimension reduction and projection methods such as singular value decomposition or principal component analysis, will be influenced by the estimates replacing the missing values [8]. Thus it is desirable to have accurate estimates of the missing values to get results from the analysis methods that are as realistic as possible.

Some methods for estimation of missing values in microarray data sets are presented in [8]. The methods, based on the least squares principle, are available in a software package called LSimpute and utilize correlations between both genes and arrays. In LSimpute software, two basic methods based on least squares principle, one utilizing correlations between genes (LSimpute_gene) and the other utilizing correlations between arrays (LSimpute_array), are used to estimate missing values. A robust method for using weighted averages of the estimates from LSimpute_gene and LSimpute_array into adaptive estimates is also available in LSimpute. The superior missing value estimation accuracy of LSimpute, as compared to KNNimpute [110], is also demonstrated with the widely used microarray data sets by randomly knocking out data (labeling as missing). Additionally, a more classic approach to missing value estimation based on expectation maximization (EM) is also examined in [8]. The results indicate that on average, the estimates from best performing LSimpute method are at least as accurate as those from the best EMimpute algorithm.

1.6.2 Normalization

To identify relationship among genes, involved in multiple biological functions or processes, many microarray experiments with different biological origins are conducted. These experiments with multiple microarray slides are sources of non-biological variation between slides such as dye biases, sample preparation or hybridization differences, scanner calibrations, slide printing variations, volume of initial RNA, etc. Some of these variabilities can be corrected by data normalization before analysis of the data. The main assumption behind normalization of microarray data is that most of the genes on the slide do not change their expression levels and the numbers of up- and down-regulated genes on the array are roughly equal. Most methods try to adjust expression levels of the genes such that the overall average expression remains the same across different arrays. Additional steps in normalization can be performed by removing saturated signals from microarray, background correction, low expression genes correction, etc. In cDNA microarray related investigations, many different methods are developed in order to compensate for dye-effects and other non-biological variations between arrays. The existing normalization methods for microarray data are broadly classified in the following groups [127]:

1. Within-slide normalization: Normalization in specific location
   - Global normalization
   - Intensity dependent normalization
   - Within-print-tip-group normalization
2. Within-slide normalization: Normalization with slide dependent scaling factor

- Median absolute deviation (MAD)
- Variance regularization [84]

3. Paired-slides normalization (dye-swap)

4. Between-slide or Multiple-slide normalization

    - Median absolute deviation (MAD)
    - Variance regularization [84]

Global normalization methods assume that the red and green intensities are related by a constant factor k (that is, $R = k \cdot G$). Generally the center of the distribution of log-ratios is shifted to zero by subtracting the median or mean of the log-ratios for a particular gene. Global normalization approaches also include rank invariant methods but do not take into account the spatial or intensity dependent die biases. On the other hand, LOWESS (locally weighted scatter plot smooth), a locally weighted linear regression method [25] accounts for such effects and has been proven to be a robust, powerful normalization method for correcting intensity-dependent ratio bias in different types of two color (R and G) microarray experiments [127]. Lowess is used for microarray data with separate R and G value for each experimental condition. For microarray datasets with given ratio values ($log_2 \frac{R}{G}$), it is assumed that intensity-dependent ratio bias is corrected by the data providers [101]. Within-print-tip-group normalization accounts for differences in the length, opening, and deformation of the tips, used for printing grids in a microarray. After this normalization all the normalized log-ratios from the different print-tip-groups will be centered around zero. However, there is a possibility that the log-ratios from the various print-tip-groups have different spread and there are substantial scale differences between them, because of changes in the photomultiplier tube settings of the scanner or for other reasons. In these circumstances within-slide scale normalization is required. Scale-normalization is a simple scaling of the log-ratios from a series of arrays and the scaling factor is estimated using median absolute deviation (MAD) or variance of log-ratios for individual print-tip-groups. Similar type of scale normalization, using MAD and variance, is performed in between-slide or multiple-slide normalization when, log-ratios from the various slides (experiments) have different spread. Here, we will discuss how scale normalization is performed using MAD or variance.

Let us assume that there are m different types of experiments (or slides or print-tip-groups), experiment type i (eg., sporulation) has a total of n_i (where, $i = 1,\ 2, \cdots m$) no. of log ratios, and Set_i denotes the total set of log ratios for experiment i. The scaling factor S_MAD_i for experiment of type i, in terms of median absolute deviation (MAD), is defined as [127]

$$S\_MAD_i = \frac{MAD_i}{\sqrt[m]{\prod_{i=1}^{m} MAD_i}}, \tag{1.1}$$

where, MAD_i is defined as

$$MAD_i = median\{|Set_i - median(Set_i)|\}. \tag{1.2}$$

After the calculation of scaling factors, all the gene expression values (log-ratios) for experiment of type i are normalized (divided) by S_MAD_i. This normalization procedure assumes that a relatively small proportion of the genes will vary significantly in expression between the two mRNA samples. In addition, it assumes that the spread of the distribution of the log-ratios should be roughly the same for all experiments (or slides or print-tip-groups) [127].

Similarly, for variance regularization, the scaling factor $S\_Variance_i$ for experiment of type i is defined as [84]

$$S\_variance_i = \frac{\sigma_i^2}{\sqrt[m]{\prod_{i=1}^{m} \sigma_i^2}}, \tag{1.3}$$

where, σ_i^2 is defined as

$$\sigma_i^2 = \frac{1}{n_i}\sum_{j=1}^{n_i}(M_{ij} - \overline{Set_i})^2. \tag{1.4}$$

In Eq. 1.4 M_{ij} indicates the jth log ratio in Set_i and $\overline{Set_i}$ is the mean log ratio of Set_i.

Paired-slides normalization applies to dye-swap experiments where, two hybridizations are performed for two mRNA samples and the dye assignment is reversed in the second hybridization [127]. A good evaluation of new normalization methods and their comparisons are available in [124] and [127], respectively.

A different approach is required to normalize the affymetrix arrays, that use only one channel to measure abundance of mRNA levels. Several methods are developed in this regard. The default method, for minimizing biological and technical variations in GeneChip expression microarray, uses a constant scaling factor (SF), for every gene on an array. The SF is obtained from a trimmed average signal of the array after excluding the 2% of the probe sets with the highest and the lowest values. Lu [68] showed that normalization factors, obtained with log transformed signals, performed the best and suggested to use the mean of the logarithm transformed data for normalization, rather than the arithmetic mean of signals. In general, the various normalization methods used in affymetrix arrays are natural extensions of cDNA methodologies.

1.6.3 Clustering Methods

Cells react to different internal and external environmental conditions with a response that results in activating a set of proteins required to use or oppose these conditions. Conditions can be of various kinds of samples, e.g. different treatments, time points, patients, etc. Within a particular experimental condition, genes whose products function together are usually co-regulated, involved in the same cellular processes, and coordinately expressed in response to stimuli. This expression property is used to identify genes that share similar expression profiles.

Clustering is commonly used in microarray experiments to identify groups of genes that share similar expressions [31]. Each group is then associated with a specific biological function or biological process. Therefore, clustering suggests functional relationships between groups of genes. It may also help in identifying promoter sequence elements that are shared among genes. In addition, clustering can be used to analyze the effects of specific changes in experimental conditions and may reveal the full cellular responses triggered by those conditions. There are many widely used clustering algorithms for analysis of microarray data, including hierarchical clustering [2, 31, 51, 54], CLICK (CLuster Identification via Connectivity Kernels) [101], k-means [49], and self-organizing map [109]. These methods differ from each other considerably and often lead to different results, even within the same method when using different distance metric as a measure of similarity between genes.

Although various clustering methods can usefully organize tables of gene expression measurements, the resulting massive collection of numbers remains difficult to assimilate. Therefore, for visual interpretation and evaluation of the clustering results, clustering methods are combined with a graphical representation of the primary data by representing each data point with a color that quantitatively and qualitatively reflects the original experimental observations [31]. Data points with log ratios of 0 are colored black, increasingly positive log ratios with reds of increasing intensity, and increasingly negative log ratios with greens of increasing intensity. The end product is a representation of complex gene expression data (see Figure 1.4) that, through statistical organization and graphical display, allows biologists to assimilate and explore the data in a natural intuitive manner.

Average linkage hierarchical clustering is one of the first clustering algorithms applied to microarray data [22, 31]. Using a distance metric, the method builds a hierarchical binary tree (called a dendrogram). Given a set of N data points to be clustered, and an $N \times N$ distance (or similarity) matrix, the basic steps of hierarchical clustering [54] are explained below.

S1) Start by assigning each item to a cluster, so that if there are N items there are N clusters, each containing just one item. So, the distances (similarities) between the clusters are the same as the distances (similarities) between the items they contain.

S2) Find the closest (most similar) pair of clusters and merge them into a single cluster, so that there is one less cluster.

S3) Compute distances (similarities) between the new cluster and each of the old clusters.

S4) Repeat S2 and S3 until all items are clustered into a single cluster of size N.

The data points are thus fashioned into a binary tree whose branch lengths represent the degree of similarity between the sets. Once the complete hierarchical tree is computed, the tree can be cut at some branch according to a threshold value to obtain clusters of required characteristics. To get k clusters one has to cut the k-1 longest links. Step 3 can be done in different ways, which is what distinguishes single-linkage from complete-linkage and average-linkage clustering. Figure 1.4 shows the average linkage clustering of genes and conditions related with

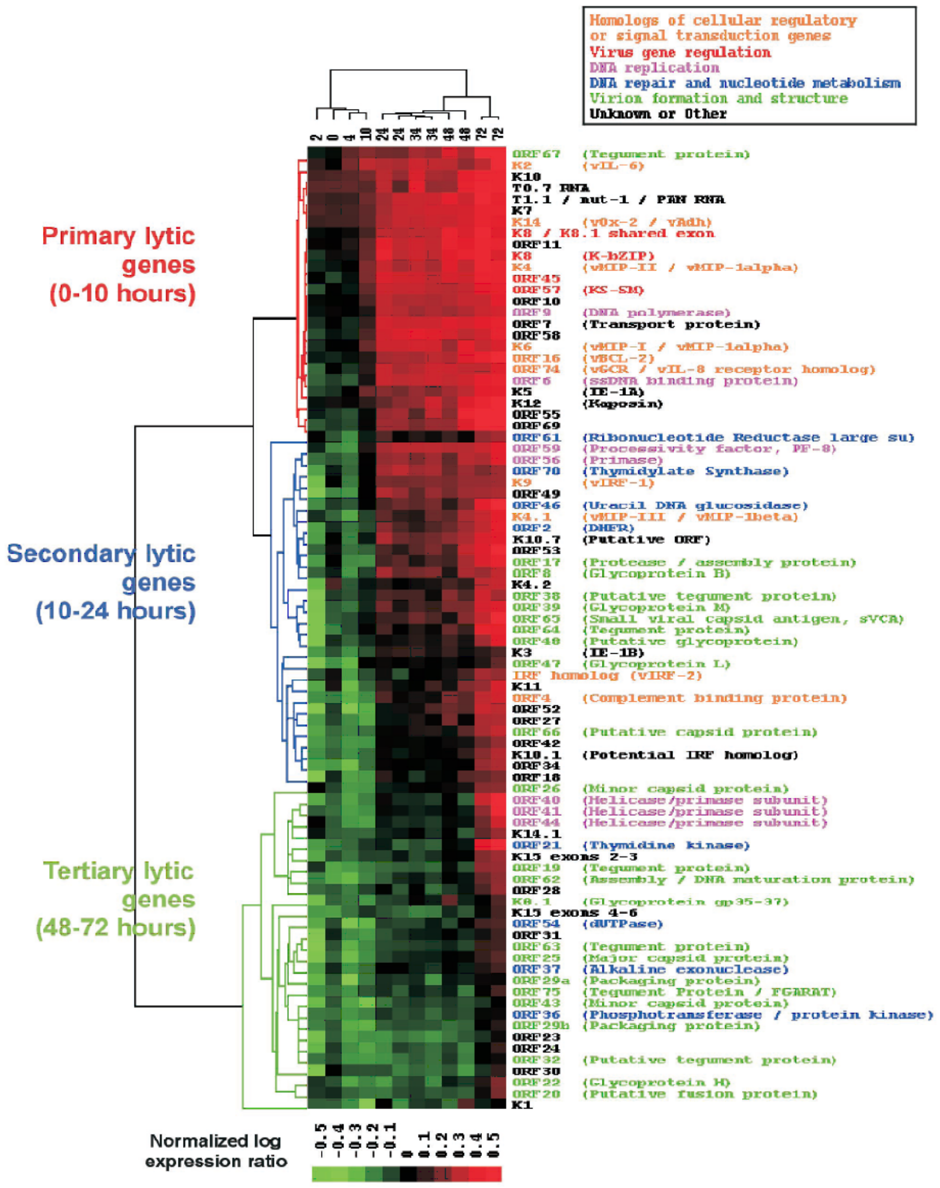


Figure 1.4: Average linkage clustering of genes and conditions related with DNA microarray gene expression of Herpes virus [51]. The dendrogram at the left represents the relatedness of the patterns of gene expression. The three major clusters are color coded according to the class of genes they represent and the times at which expression is first detected: primary lytic genes (0 to 10 h), secondary lytic genes (10 to 24 h), and tertiary lytic genes (48 to 72 h). Each column represents a condition taken at different times in hours after TPA induction (labelled above). The dendrogram at the top relates the conditions according to the pattern of gene expression.

DNA microarray gene expression of Herpes virus [51]. The genes are ordered using a self-organizing map algorithm [31, 51]. The normalized log expression ratio is color coded according to the scale at the bottom. ORFs (Open Reading Frames) and corresponding gene names are listed on the right and color coded according to putative function shown by the key above.

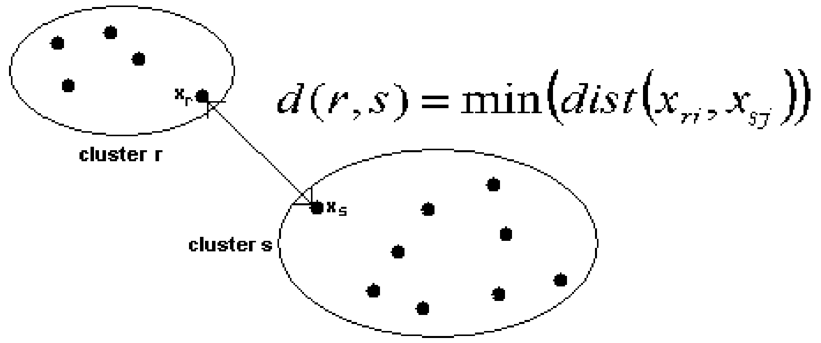


Figure 1.5: Computation of distance ($d(r, s)$) between clusters in single linkage clustering. Here the distance between every possible object pair (x_{ri},x_{sj}) is computed, where object x_{ri} is in cluster r and object x_{sj} is in cluster s. The minimum value of these distances is said to be the distance between clusters r and s. In other words, the distance between two clusters is given by the value of the shortest link between the clusters. At each stage of clustering, the clusters r and s , for which $d(r, s)$ is minimum, are merged.

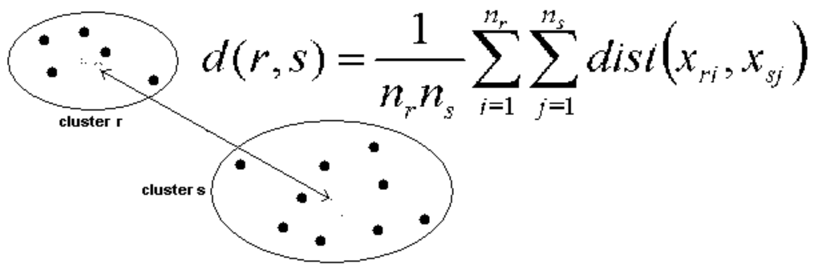

cluster r
cluster s
$d(r,s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} dist(x_{ri}, x_{sj})$

Figure 1.6: Computation of distance ($d(r, s)$) between clusters in average linkage clustering. Here the distance between two clusters is defined as the average of distances between all pairs of objects, where each pair (x_{ri},x_{sj}) is made up of one object from each group. Object x_{ri} is in cluster r, object x_{sj} is in cluster s, and n_r and n_s are sizes of clusters r and s, respectively.

In single-linkage clustering (also called the connectedness or minimum method), the shortest distance from any member of one cluster to any member of the other

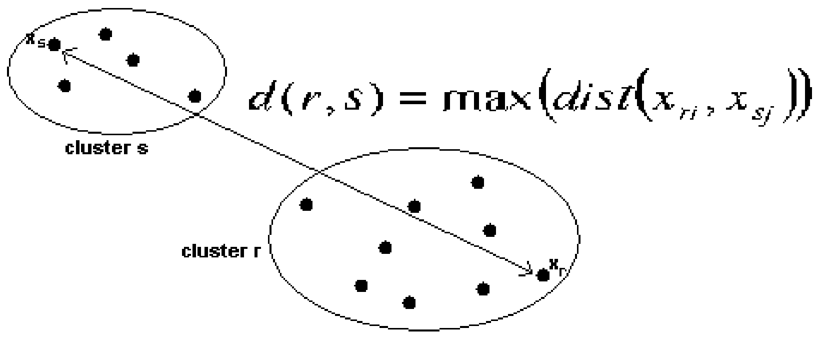


Figure 1.7: Computation of distance ($d(r,s)$) between clusters in complete linkage clustering. Here the distance between every possible object pair (x_{ri},x_{sj}) is computed, where object x_{ri} is in cluster r and object x_{sj} is in cluster s. The maximum value of these distances is said to be the distance between clusters r and s.

cluster is considered as the distance between one cluster and another cluster (shown in Fig. 1.5). If the data consist of similarities, the similarity between one cluster and another cluster is considered to be equal to the highest similarity from any member of one cluster to any member of the other cluster. In complete-linkage (also called the diameter or maximum method) and average-linkage clustering, the distance between one cluster and another cluster is considered to be equal to 'the largest distance from any member of one cluster to any member of the other cluster' (shown in Fig. 1.7) and 'average distance from any member of one cluster to any member of the other cluster' (shown in Fig. 1.6), respectively. A variation on average-link clustering uses the median distance and is more outlier-proof than the average distance. This kind of hierarchical clustering is called agglomerative because it merges clusters iteratively. There is also a divisive hierarchical clustering which does the reverse by starting with all objects in one cluster and subdividing them into smaller pieces.

Although, hierarchical clustering is simple and clear for representation, it has a number of shortcomings for the study of gene expression. First, the deterministic nature of hierarchical clustering can cause points to be grouped based on local decisions, with no opportunity to reevaluate the clustering. It is known that the resulting trees can lock in accidental features, reflecting idiosyncrasies of the agglomeration rule [109]. Second, dendrograms and corresponding heatmaps, which are used for visualization and analysis of the clustering results, suffer from inversion problems that complicate interpretation of the hierarchy [75]. And finally, complexity of dendrograms for larger data sets makes them difficult to understand, and the choice of location for tree cut to receive final clusters is unclear.

The k-means clustering [49, 71] is one of the simplest partitive unsupervised algorithms that partitions the data into k clusters. The main idea is to define k centroids, one for each cluster. These centroids should be initially placed as far as possible from each other. The next step is to assign each data point to the nearest centroid. When all the points are so assigned then k new centroids are recalculated from the data points of each cluster. The whole process is repeated until no more changes are observed in the locations of all the k centroids. The algorithm is composed of the following steps:

S1) Place k points into the space represented by the data points that are being clustered. These k points represent initial group centroids.

S2) Assign each data point to the group that has the closest centroid.

S3) When all data points have been assigned, recalculate the positions of the k centroids.

S4) Repeat S2 and S3 until the centroids no longer move. This produces a separation of the data points into groups from which the objective function, to be minimized, can be calculated using

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n_j} ||x_i^{(j)} - c_j||^2 \tag{1.5}$$

where, n_j is the number of data points assigned to jth cluster and $||x_i^{(j)} - c_j||^2$ is an indicator of the distance between a data point $x_i^{(j)}$ and the cluster center c_j with a chosen distance measure.

While simplicity and speed are the main advantages of the k-means, the disadvantage is that the method proceeds in an entirely local fashion and produces an unorganized collection of clusters that is not conducive for interpretation of microarray data [109]. Although it can be proved that the k-means will always terminate, the algorithm does not necessarily find the most optimal configuration, corresponding to the global minimum of the objective function. The algorithm is also significantly sensitive to the initial randomly selected cluster centers and can be run multiple

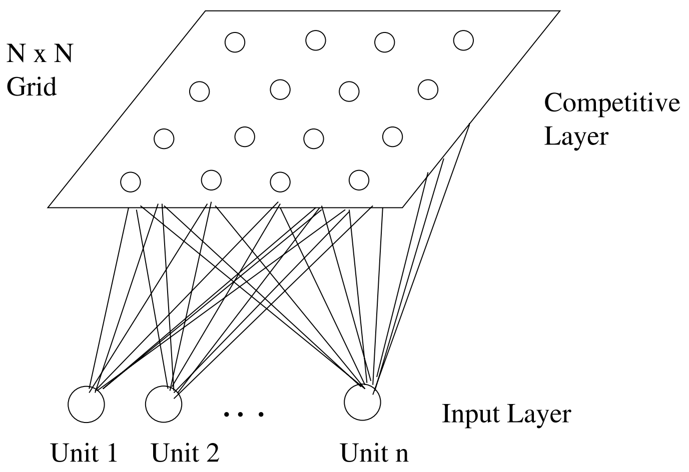


Figure 1.8: The basic network structure for the SOM.

times to reduce this effect. Another obstacle is to estimate the number of clusters prior to analysis of the data. Several approaches to minimize this obstacle are available in [13].

Self-Organizing Maps (SOMs) [59] are suited to exploratory data analysis, allowing one to impose partial structure on the clusters (in contrast to the rigid structure of hierarchical clustering and the non-structure of k-means clustering) and facilitating easy visualization and interpretation of gene expression patterns [109].

In SOM, k-dimensional data points are randomly projected into an initial map (usually one- or two-dimensional) of nodes, represented by k-dimensional weight vectors. The feature map is a two-layered network. The first layer of the network is the input layer. The second layer, called the competitive layer, is usually organized as a two-dimensional grid of nodes. All interconnections go from the first layer to the second (see Fig. 1.8). In the second layer, each node has two components. The first part is its weight vector which is of the same dimensions as the input vectors. The second part of a node is its natural location in the map. At each iteration of the algorithm an input vector is chosen randomly, and all the nodes in the competitive layer compare the inputs with their weights and compete with each other to become the winning node having the lowest difference.The best-matching weight vector with the shortest distance or highest similarity is the winner node. It has a location, neighboring nodes that are close to it, and been updated the most compared to more distant neighboring nodes.

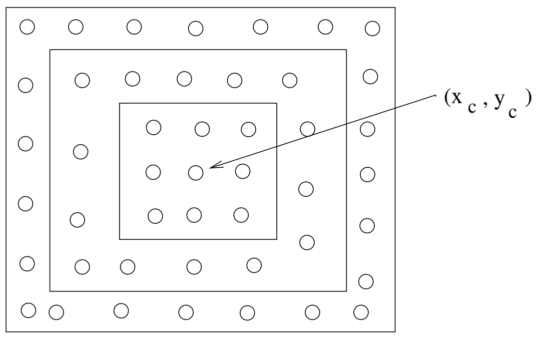


Figure 1.9: Neighborhood N_c, centered on unit c (x_c, y_c). Three different neighborhoods are shown at distance d = 1, 2, and 3.

In biophysically inspired neural network models, correlated learning by spatially neighboring cells can be implemented using various kinds of lateral feedback connections and other lateral interactions. Here the lateral interaction is enforced directly in a general form, for arbitrary underlying network structures, by defining a neighborhood set N_c around the winner node c. The weight vectors of the winner node and its neighbors are updated in the direction of the input vector and depending on learning rate (α) and similarity/distance between a node and that input vector. There are two parts in updating the winner and its neighbors: determining which nodes are considered as neighbors and how much each node can learn to become more like the input vector. In the first part, the neighborhood set can be determined using a number of different methods, such as, concentric squares, hexagons, and gaussian function where, every point with a value above zero is considered a neighbor. The number of neighbors are also decreased over time by decreasing the width or radius of N_c , so that, input vectors can first move to an area where they will probably be and then they compete for position. In fact, for good global ordering, it has experimentally turned out to be advantageous to let N_c be very wide in the beginning and shrink monotonically with time (shown in Fig. 1.9). This process is similar to coarse adjustment followed by fine tuning and allows the topological order of the map to be formed.

The second part involves the learning process of the nodes. At each learning step, all the nodes within N_c are updated, whereas nodes outside N_c are left intact.

The update equation is:

$$\Delta m_{ij} = \begin{cases} \alpha(x_j - m_{ij}) & \text{if unit } i \text{ is in the neighborhood } N_c, \\ 0 & \text{otherwise,} \end{cases} \tag{1.6}$$

and

$$m_{ij}^{\text{new}} = m_{ij}^{\text{old}} + \Delta m_{ij} \tag{1.7}$$

Here α is the learning rate. This adjustment results in both the winning node and its neighbors, having their weights modified, becoming more like the input pattern. The winner then becomes more likely to win the competition should the same or a similar input pattern be presented subsequently. An attribute of this learning process is that, the farther away the neighbor is from the winner node, the less it learns. The rate at which a node can learn decreases with iteration and can also be set to a user defined value. Using a Gaussian function the learning rate will return a value ranging between 0 and 1.

The next step is to self-organize a two-dimensional map that reflects the distribution of input patterns. After a number of iterations involving the above mentioned processes, nodes represent clusters with self-organized neighbor nodes in the map defining related clusters. The algorithm is composed of the following steps:

S1) Initialize the k-dimensional weight vector of each node in a one- or two-dimensional map.

S2) Randomly select a k-dimensional input vector and search the map of nodes (weight vectors) to find that which best represents that input vector.

S3) Introduce two new user defined variables, the neighborhood set (N_c) for the winner node and the learning rate (α).

S4) Update the weight vector of the winner node according to the learning rate (g), so that it becomes more similar to the randomly selected input vector. If the current g value is set to a maximum (i.e. to 100%) then the winner node is actually made to be exactly the same as the selected input vector.

S5) Update the weight vectors of the neighboring nodes within neighborhood set (N_c), so that, they become more like the randomly selected input vector in S2.

S6) Decrease learning rate and neighborhood set.

S7) Repeat S2 to S6 for more than 1000 iterations.

SOM can also yield different decompositions of the data depending on the choice of initial geometries of nodes, such as maps, rings, and lines, with different numbers of nodes. Although SOMs are easy to implement, reasonably fast, and scalable to large data sets, sensitivity of SOM to incomplete data is a problem which can only be tackled with missing value imputation methods [8].

The clustering results of SOM and K-means are not unique across different runs and depend on starting positions of centroids or nodes. These algorithms also need a prior assumption on the number of clusters or their structure. To overcome these issues CLICK is proposed in [101]. The algorithm recursively partitions a weighted graph into components using minimum cut computations. The edge weights and the stopping criterion of the recursion are assigned probabilistic meaning, which gives the algorithm higher accuracy than the related methods like hierarchical clustering and self organizing maps. CLICK is also used in the identification of common regulatory motifs in the promoters of co-regulated genes, and in the classification of samples into disease types based on their expression profiles.

The clustering methods, discussed so far, are used to relate one gene to a single co-expression cluster, when many individual genes are involved in more than one process [30, 38] of the cell and therefore co-express in multiple groups. Some attempts [42] are made to relate genes with multiple biological processes using fuzzy k-mean clustering that allows the genes to belong to more than one cluster, with a variable degree of ‘membership’. Each gene has a total membership of 1.0 that is apportioned to clusters on the basis of the similarity between the gene’s expression pattern and that of each cluster centroid. Importantly, genes can be assigned significant memberships to more than one cluster, thus revealing genes whose expression is similar to multiple, distinct groups of genes.

1.6.4 Ordering Methods

Hierarchical clustering is one of the most popular methods for clustering gene expression data. The method assembles input elements into a single tree, and subtrees represent different clusters. Thus, using hierarchical clustering one can analyze and visualize relationships in scales that range from large groups (clusters) to single genes. However, hierarchical clustering does not provide decisions about clusters, making it hard to distinguish between internal nodes that are roots of a cluster and nodes which only hold subsets of a cluster [14]. Therefore, the leaves of the binary tree, computed from hierarchical clustering, are usually ordered in a linear fashion, so that genes or groups of genes with similar expression patterns are adjacent [14,31]. The ordered leaves can then be displayed graphically with a representation of the tree to identify the clusters and to indicate the relationships among genes and subclusters (see Figure 1.4). So in the framework of hierarchical clustering a gene ordering algorithm helps the user to identify clusters by means of visual display and interpret the data [14]. For hierarchical clustering based approaches, as well as outside the framework of hierarchical clustering, microarray gene ordering (MGO) using gene expression information is necessary for the following reasons:

1. Gene ordering helps to identify clusters and subclusters by means of smooth visual display of the ordered gene expression data [14], where the functionally related genes are nearer in the ordering [18].

2. It is believed that genes are influenced on an average by no more than eight to ten other genes [26]. Such a relation can be achieved by gene ordering, which enables molecular biologists concentrate on a sensible subset of genes and infer genetic networks [29].

3. Genes that are adjacent in a linear ordering are often functionally co-regulated and involved in the same cellular process [18, 31]. Biological analysis is often done in the context of this linear ordering [14].

The problem of ordering the leaves of a binary hierarchical clustering tree dates back (1972) to the investigation of Gruvaeus and Wainer [47]. Over the years, many different heuristics have been suggested for solving this problem [31, 39, 47]. These heuristics either use a local method, where decisions are made based on local observations, or a global method, where an external criteria is used to order the leaves. Eisen et. al [31] used a simple method for ordering leaves (genes) in a hierarchical clustering solution. They used the procedure of weighting genes, such as average expression level, time of maximal induction, or chromosomal position, and placed the element with the lower average weight earlier in the final ordering [31].

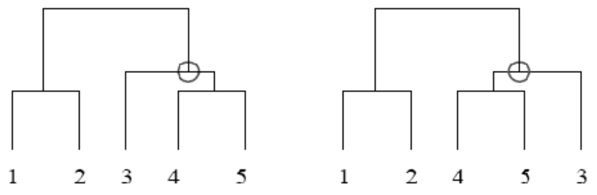


Figure 1.10: Change in the leaf ordering due to an internal node flip [14]. Different orderings are obtained by flipping the two subtrees rooted at the circled node while, maintaining the same tree structure. Since there are $n-1$ internal nodes there are 2^{n-1} possible orderings of the tree leaves.

However, the ordering of the leaves, which plays an important role in analyzing and visualizing hierarchical clustering results, is not defined in [31].

An optimal leaf ordering method in hierarchical clustering framework is demonstrated in [14]. The method maximizes the similarity of adjacent genes in the ordering by flipping the internal nodes in a hierarchical solution. First, the optimal leaf ordering problem is formalized and then the related algorithm [14] is described. For a tree T with n leaves, denote by $z_1, \cdots, z_n$ the leaves of T, and by $v_1, \cdots, v_{n-1}$ the $n-1$ internal nodes of T. A linear ordering consistent with T is defined to be an ordering of the leaves of T generated by flipping internal nodes in T (that is, changing the order between the two subtrees rooted at v_i, for any $v_i \varepsilon T$). See Figure 1.10 for an example of node flipping. For any tree (dendrogram) of n genes, there are 2^{n-1} linear orderings consistent with the structure of the tree. The goal is to find an ordering of the tree leaves that maximizes the sum of the similarities of adjacent leaves in the ordering. This could be stated mathematically in the following way. Denote by Φ the space of the 2^{n-1} possible orderings of the tree leaves. For $\phi \in \Phi$, $D^{\phi}(T)$ is defined as:

$$D^{\phi}(T) = \sum_{i=1}^{n-1} S(z_{\phi_i}, z_{\phi_{i+1}}) \tag{1.8}$$

where, z_{ϕ_i} is the ith leaf when T is ordered according to ϕ and $S(z_{\phi_i}, z_{\phi_{i+1}})$ is the similarity between two leaves of the tree. Thus, the goal is to find an ordering ϕ that maximize $D^{\phi}(T)$. For such an ordering, $D(T) = D^{\phi}(T)$.

Assume that a hierarchical clustering in form of a tree T has been fixed. The basic idea is to create a table M with the following meaning. For any node v of T,

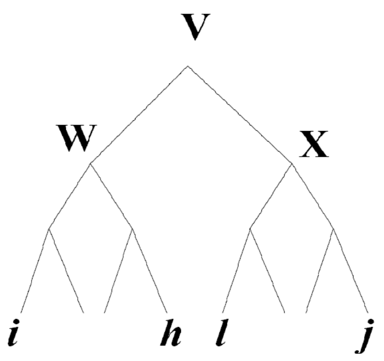


Figure 1.11: A binary tree rooted at node v. For every pair of leaves $i \varepsilon W$ and $j \varepsilon X$ the optimal leaf ordering, M(v, i, j), is computed when the leftmost leaf of W is i and the rightmost leaf of X is j

and any two genes i and j that are at leaves in the subtree defined by v (denoted $T(v)$), define $M(v, i, j)$ to be the cost of the best linear order of the leaves in $T(v)$ that begins with i and ends with j. $M(v, i, j)$ is defined only if node v is the least common ancestor of leaves i and j; otherwise no such ordering is possible. If v is a leaf, then $M(v, i, j) = 0$. Otherwise, $M(v, i, j)$ can be computed as [14]:

$$M(v,i,j) = \max_{h \in T(w), l \in T(x)} M(w,i,h) \ + \ S(h,l) \ + \ M(x,l,j) \tag{1.9}$$

where, w is the left child and x is the right child of v (see Figure 1.11).

Outside the framework of hierarchical clustering, unidirectional Microarray Gene Ordering (MGO) can be performed for identifying highly correlated genes. The ordering acts as an alternative method for clustering and can be formulated with some minor modifications in the Travelling Salesman Problem (TSP), one of the most important test-beds for new combinatorial optimization methods [61]. Beidl et al. [18] formulated the MGO problem as TSP by associating one imaginary city to each gene, and obtaining the distance between any two genes (cities) from the matrix of inter gene distances. Consequently, the MGO problem as TSP is solved using Genetic Algorithm (GA), an effective technique in finding near optimal solutions in short computational time for large combinatorial optimization problems. The importance of GA and TSP stems from the fact there is a plethora of fields in which it finds applications e.g., shop floor control (scheduling), distribution of goods and services (vehicle routing), product design (VLSI layout), protein structure prediction, and DNA fragment assembly. Definition of TSP, its relevance in

MGO problem, and relevant investigations using GA in this regard are available in Chapter 2 and [18, 91, 115]. The application of gene ordering and its utility in partitive clustering is investigated in Chapter 3. In brief, gene ordering helps to determine which of the gene groups are unique and which groups are only a part of a bigger group by means of visual inspection of the ordered gene expression data.

1.6.5 Methods for Combining Multi-Source Information with Microarray Data

One of the important goals of biological investigation is to predict the function and pathway of unclassified gene. Even in a model organism like Yeast, there are more than 1000 genes with unknown biological function defined in Munich Information for Protein Sequences (MIPS) [38] and Saccharomyces Genome Database (SGD) [30]. An approach in predicting function or pathway of unclassified gene involves identifying the group of its closest classified genes and assigning the common biological function or pathway of the group to the unclassified gene, using different sources of information, such as microarray gene expressions [31, 109], protein sequences [73], protein-protein interaction data [95, 97], and phenotypic profiles [20]. Among these heterogeneous functional data sources, gene expressions or phenotypic profiles are relatively new sources for gene function prediction, but they alone often lack the degree of specificity needed for accurate prediction. This lack of specificity is due to the drawbacks of microarray technology (as mentioned in Section 1.5), and it can be overcomed through the incorporation of heterogeneous functional data in an integrated analysis [111]. The working hypothesis is that each set of functional genomics data has an intrinsic error rate and a limited coverage but informs us to some extent about the tendency for genes to operate in the same cellular functions and pathways in the cell. Therefore a more accurate and extensive functional predictions can be achieved by integrating the information from multiple functional genomics datasets, and in this manner the overall functional coupling between genes across a broad set of experiments can be estimated [62].

The value of combining multi-source information with microarray data, for gene function prediction, is first illustrated in [73]. Consequently, related methods and algorithms are developed in [74], [111], [106], [126], and [62]. All the methods have a unified scoring scheme, based on gene annotation, for testing the heterogeneous data sets, even when the data sets are accompanied by their own intrinsic scoring method (such as Pearson correlation for gene expression). This re-scoring by a single criterion allows one to directly measure the relative merit of each data set, and then to integrate the data sets. A brief description of different methods of integration are available in Chapter 5. Here we will describe the method of Lee et al. [62], as we have used it for comparison with our method in Chapter 5.

In [62], different sets of microarray data are analyzed for significant co-expression of pairs of genes, and pairs from the assortment of different microarray data sets are collected to generate a single set. These expression-based pairs are then integrated with other protein-protein interaction experiments, literature mining pairs, and gene context pairs to produce the initial integrated network. A Bayesian statistics approach is used for scoring different data sets, where, each data set adds some degree of evidence that a pair of genes is functionally linked. More specifically, the odds ratio, representing the likelihood that a pair of genes is functionally linked, is calculated. If $P(L|E)$ represents the probability of linkage between a pair of genes conditioned on the given evidence (and $\sim P(L|E)$ represents the probability that these genes are not functionally linked), and $P(L)$ is the unconditional probability of linkage between a pair of genes, the odds ratio (OR) that the given pair of genes is linked is given as [62]:

$$OR(L,\ E) = \frac{P(L|E)/\sim P(L|E)}{P(L)/\sim P(L)} \tag{1.10}$$

In Bayesian terms, the ratio $P(L)/\sim P(L)$ represents the prior odds ratio, which is the ratio of the probability of the linkage and its negation before the evidence is seen. This term is estimated by counting the number of gene pairs with any shared functional annotation (using only a single source of functional annotation, for example, the Kyoto Encyclopedia of Genes and Genomes (KEGG) [55] pathway annotation) and those without any shared functional annotation among all possible gene pairs chosen from the set of annotated yeast genes. The ratio $P(L|E)/\sim P(L|E)$ represents the posterior odds ratio, which is the ratio of the probability of the linkage and its negation conditioned on the given evidence. For estimating these conditional odds, the number of gene pairs that share or do not share functional annotation and that are also supported by the given evidence are counted. The $OR(L,\ E)$ can therefore be interpreted as the ‘likelihood of the linkage conditioned on the given evidence and corrected for background expectations of pairs. The methodology for calculating $OR(L,\ E)$ for each data set, is as follows:

S1) Sort all possible gene pairs by its own intrinsic scoring method and make bins, each containing 20,000 gene pairs.

S2) For each bin, measure frequencies of genes sharing ($P(L|E)$) or not sharing ($\sim P(L|E)$) pathways/processes, based on KEGG or Gene Ontology (GO) process 8th level annotation.

S3) Plot curves for 'average intrinsic score of bin (20,000 gene pairs)' vs. $P(L|E)$ and 'average intrinsic score of bin' vs. $\sim P(L|E)$.

S4) Compute $OR(L,\ E)$ for each bin (i.e., for all gene pairs) from these curves and the odds ratio of prior expectations, $P(L)$ and $\sim P(L)$.

Now calculate the natural logarithm of $OR(L,\ E)$, the log likelihood ratio, in order to create an additive score from different $ln(OR(L,\ E))$, available from different data sets. The ultimate score for each gene pair is based upon a weighted sum of $ln(OR(L,\ E))$ scores. For data sets that provide only binary evidence (e.g., observed to interact or not observed to interact), all linkages (observed to interact) derived from the same data set are scored with an identical $ln(OR(L,\ E))$ value calculated as the log of the odds ratio described above. Other data sets provide gene pairs with associated parametric scores, such as the correlation coefficients indicating the degree of mRNA co-expression, the probability scores of genes being linked by gene fusions or co-citation, and the mutual information score indicating the degree of coinheritance of genes in phylogenetic profiling [80].

1.6.6 Statistical methods

Statistical tests are necessary to infer the significance of biological findings or advantages of a computational method over related methods. Some of the widely used and popular statistical methods for microarray analysis are Students t-test, Mann-Whitney-Wilcoxon rank test, analysis of variance (ANOVA) and significance analysis of gene clusters with p-values. Students t-test is a hypothesis test for answering questions about the mean between two sets of data where the data are a random sample of independent observations from an underlying normal distribution [46].

In Chapter 4 t-test is used to test the alternative hypothesis, that the average of 'percentages of improvement over the lowest biological score' for the proposed (*Maxrange-M*) distance is better than the related one (Pearson or Euclidean) [89]. ANOVA compares group variations to the overall variation observed by using Fishers F-distribution as part of the test of statistical significance [57]. Variations of ANOVA analysis include one-way and factorial or non-parametric ANOVA, that are used depending on experimental design or hypothesis testing [24, 79]. Mann-Whitney-Wilcoxon rank test is a non-parametric statistical test that compares for each gene the difference between measurements in two groups. The biological significance of any clusters generated by a clustering methods can be evaluated with a *p-value*, using biological functional categories of the genes. The *p-value* gives the probability of observing at least m genes from a functional category within a cluster of size n when, the total number of genes within that functional category and the total number of genes within the genome are available. If a *p-value* is found statistically significant then the related functional category is assigned to that cluster.

1.6.7 Gene Annotations

Microarray data analysis is a very demanding task involving multiple steps of data pre-processing, missing value prediction, normalization, filtering, data mining, and interpretation of results. Many steps of the process can be enhanced by additional biological knowledge about the genes being analyzed. Gene annotation [7, 30, 38] information is one of the most important biological knowledge that is currently used for validation of clustering and classification results or expression patterns recovered by data analysis. Most of the gene annotation informations are in the form of genes grouped by the biological function, process, component or sub-cellular location [7, 30, 38]. Result interpretation steps can also be enhanced by a broad variety of annotation information about genes under study, including pathways, transcription factors, chromosomal location, etc.

Gene filtering is a pre-processing step that removes genes that do not vary over experimental conditions or do not have specific expression profile determined by the nature of the experiment. In this step annotations of the genes are used to identify and remove them. In some normalization techniques housekeeping or other constitutively expressed genes are first identified with gene annotation information and then instead of using all genes on the array for normalization of data, the smaller subset of housekeeping genes are used, which are believed to have constant expression across a variety of conditions [127]. In a missing value prediction method [117], genes, whose annotations are related to same biological processes, are first grouped together and then the missing expression values are predicted from the existing expression values of the group. For supervised algorithms like support vector machines (SVM) [21] and neural networks [58], gene annotations can be used to form training sets from genes of specific functional class. In the validation process, the algorithms take advantage of already known annotation informations. Comprehensive and structured annotations for all genes are also essential for pathway prediction and function prediction of unclassified genes from biological datasets like phenotypic profiles [20], protein sequences [73], Kyoto Encyclopedia of Genes and Genomes (KEGG) pathway database [55], protein-protein interactions [95,97], phylogenetic profiles [80] and Rosetta Stone sequences [72].

1.7 Scope of the Thesis

A central step in the analysis of gene expression data is the identification of groups of genes that exhibit similar expression patterns. Clustering and ordering the genes, using gene expression data, into homogeneous groups was shown to be useful in functional annotation, tissue classification, regulatory motif identification, and other applications. Again, based on the surveys [77,85] on bioinformatics, genetic algorithm (GAs) [77,87] and artificial neural networks (ANNs) are found to be promising tools for optimization and classification/categorization tasks in the current state of art. In order to classify microarray data, an appropriate measure is also needed to find the similarity between genes in terms of their expression values.

The present thesis deals with the development of novel computational methods in bioinformatics for ordering and clustering of genes from microarray data, defining new distance measures for genes using an microarray experiment specific normalization factor, and developing new scoring framework for predicting the function of a few unclassified Yeast genes by integrating microarray gene expressions with other informative data sources. Both GAs and deterministic computational methods are used as tools. Superiority of the methods is established on the accuracy of gene ordering, grouping functionally similar genes, and predicting the biological function of genes. The results of these investigations are summarized below on the basis of different chapter headings.

1.7.1 Gene Ordering in Genetic Algorithm (GA) Framework [88, 91]

Some new operators of genetic algorithms and their effectiveness to the travelling salesman problem (TSP) and microarray gene ordering is demonstrated in Chapter 2. The new operators developed are nearest fragment operator based on the concept of nearest neighbor heuristic, and a modified version of order crossover operator. While these result in faster convergence of GA in finding the optimal order of genes in microarray and cities in TSP, the nearest fragment operator can augment the search space quickly and thus obtain much better results compared to other heuristics. The genetic algorithm with the new operators is referred as FRAG_GALK. Appropriate number of fragments for the nearest fragment operator and appropriate substring length in terms of the number of cities/genes for the modified order crossover operators are determined systematically. Gene order provided by the proposed method is seen to be superior to other related methods based on GAs, neural networks and clustering in terms of biological scores computed using categorization of the genes.

1.7.2 Integrating Gene Ordering with Partitive Clustering [86, 90]

Chapter 3 deals with a new hybrid method for ordering genes in each of the clusters obtained from partitive clustering solution, using microarray gene expressions. Two existing algorithms for optimally ordering cities in TSP, namely, FRAG_GALK and Concorde, are hybridized individually with Self Organizing MAP to show the importance of gene ordering in partitive clustering framework. Our hybrid approach is validated using Yeast and Fibroblast data and it is shown that our approach improves the result quality of partitive clustering solution, by identifying subclusters within big clusters, grouping functionally correlated genes within clusters, minimization of summation of gene expression distances, and the maximization of biological gene ordering using MIPS categorization. Moreover, the new hybrid approach, finds comparable or sometimes superior biological gene order in less computation time than those obtained by optimal leaf ordering in hierarchical clustering solution.

1.7.3 New Distance Measure for Microarray Gene Expressions using Linear Dynamic Range of Photo Multiplier Tube [89, 92]

Chapter 4 deals with a new distance measure for genes using their microarray expressions. The distance measure is called, "*Maxrange* distance", where an experiment specific normalization factor is incorporated in the computation of the distance. The normalization factor is dependent on the linear dynamic range of the photo multiplier tube (PMT) for scanning fluorescence intensities of the gene expression values. Superiority of this distance measure in the microarray gene ordering problem has been extensively established on widely studied microarray data sets by performing statistical tests.

1.7.4 Adaptively Combining Multi-Source Information with Microarray Data [93, 94]

A new scoring framework for predicting the function of a few unclassified Yeast genes is developed in Chapter 5. The score, called *Biological Score* (BS), is computed by first evaluating the similarities between genes, arising from different data sources, in a common framework, and then integrating them in a linear combination style through weights. We use microarray gene expressions, phenotypic profiles, KEGG pathway information, protein similarity through transitive homologues, and protein-protein interactions as data sources. The relative weight of each data source, in the score, is determined adaptively by utilizing the information on functional annotations (super GO-Slim) of classified genes, available from Saccharomyces Genome Database (SGD). Genes are grouped by a method, called *K-BS*, where, for each gene, a cluster comprising that gene and its K nearest neighbors is computed using the proposed score (BS) and evaluated with MIPS annotation. We predict the functional categories of 417 classified genes from 417 clusters with 98.20% accuracy. The method is then used to predict the functional categories of 12 unclassified Yeast genes.

1.8 Conclusions and Scope for Future Research

The concluding remarks with further scope for research are presented in Chapter 6.

Chapter 2

Genetic Operators for Combinatorial Optimization in TSP and Microarray Gene Ordering

2.1 Introduction

In Section 1.6.4 it is mentioned that outside the framework of hierarchical clustering, unidirectional Microarray Gene Ordering (MGO) can be performed for identifying highly correlated genes. First, a notion of distance needs to be defined in order to measure similarity among genes using gene expression values and then the genes can be ordered using it. Expression similarity between genes can be measured in terms of Euclidean distance, Pearson correlation, absolute correlation, Spearman rank correlation, etc. In order to determine the functional relationships between groups of genes that are often co-regulated and involved in the same cellular process, gene ordering is necessary. Gene ordering provides a sequence of genes such that those that are functionally related are closer to each other in the ordering [18]. The utility of gene ordering is discussed in detail in Section 1.6.4. A good solution of the microarray gene ordering (MGO) problem (i.e., finding optimal order of large DNA microarray gene expression data) has similar genes grouped together. The ordering problem can be formulated with some minor modifications in the Travelling Salesman Problem (TSP), one of the most important test-beds for new combinatorial optimization methods [61] which has been addressed extensively by mathematicians and computer scientists. Beidl et al. [18] formulated the MGO problem as TSP by associating one imaginary city to each gene, and obtaining the distance between any two genes (cities) from the matrix of inter gene distances. Consequently, the MGO problem as TSP is solved using Genetic Algorithm (GA), an effective technique in finding near optimal solutions in short computational time for large combinatorial optimization problems. Since the TSP has proved to belong to the class of NP-hard problems [41], heuristics and metaheuristics occupy an important place in the methods so far developed to provide practical solutions for large instances and any problem belonging to the NP-class can be formulated with TSP. The classical formulation is stated as: Given a finite set of cities and the cost of traveling from city i to city j, if a traveling salesman were to visit each city exactly once and then return to the home city, find the tour that would incur the minimum cost.

Over decades, researchers have suggested a multitude of heuristic algorithms, such as genetic algorithms (GAs) [45, 52, 87, 112], tabu search [36, 128], neural networks [9, 82], and ant colonies [108] for solving TSP. Of particular interest are the GAs, due to the effectiveness achieved by this class of techniques in finding near optimal solutions in short computational time for large combinatorial optimization problems. The state-of-the-art techniques for solving TSP with GA incorporates various local search heuristics including modified versions of Lin-Kernighan (LK) heuristic [6,40,48,65]. It has been found that, hybridization of local search heuristics with GA for solving TSP leads to better performance, in general. Some important considerations in integrating GAs and Lin-Kernighan heuristic, selection of a proper representation strategy, creation of the initial population, and designing of various genetic operators are discussed in [114]. A comprehensive discussion regarding different representation strategies for TSP is provided in [61].

For creating the initial population, random population based approach and nearest neighbor tour construction heuristic (NN) approach are commonly used. Some considerations regarding the random population based approach are available in [52] and [87]. A GA with immunity (IGA) is developed in [52]. It is based on the theory of immunity in biology, which mainly constructs an immune operator accomplished in two steps: a) a vaccination and b) an immune selection. Strategies and methods of selecting vaccines and constructing an immune operator are also mentioned in [52]. IGA can improve the searching ability and adaptability of TSP. Two operators of GA, namely, knowledge based multiple inversion (KBMI) and knowledge based neighborhood swapping (KBNS) are reported in [87]. KBMI helps in exploring the search space efficiently and prevents the GA from getting stuck in the local optima, whereas KBNS, a deterministic operator, helps the stochastic environment of the working of the GA to derive an extra boost in the positive direction. The corresponding GA for solving TSP is referred to as SWAP_GATSP [87].

Nearest neighbor (NN) tour construction heuristic is a common choice to construct the initial population of chromosome for solving TSP with GAs. Investigations in this line include [17, 53, 96, 112]. In [112] a modified multiple-searching genetic algorithm (MMGA) is used with two kinds of chromosomes namely, conservative and explorer. These two chromosomes operate under different crossover and mutation rates for tour improvement and to avoid the possibility of being trapped at local optima in TSP. Since the NN heuristic takes a locally greedy decision at each step, it is found that several cities that were neglected earlier, may need to be inserted at high costs in the end. This leads to severe mistakes in path construction.

Crossover operators of GAs are seen to rectify the mistakes in path construction by NN or any other approach. Various crossover operators such as order crossover [28], cycle crossover [76], partially matched crossover [45], edge-recombination crossover [107, 122], and matrix crossover [50] have been suggested for the TSP. Order crossover has been observed to be one of the best in terms of quality and speed, and yet is simple to implement for solving TSP using GA [45, 61, 87]. However, the random length of a substring, chosen from the parent string for performing crossover, may increase the computational time to some extent.

The present investigation has three parts. First, we define a new nearest fragment operator (NF) and a modified version of order crossover operator (viz., modified order crossover, MOC). The NF reduces the limitation of NN heuristic in path construction. This reduction is achieved by determining optimum number of fragments in terms of the number of cities and then greedily reconnecting them. The nearest fragment operator also takes care of the neighbor genes, not the distant ones, for MGO and provides good results without ignoring any long distances between genes for fitness evaluation. The modified version of order crossover operator

(MOC) handles the indefinite computational time due to random length of substring and its random insertion in order crossover. This is done by systematically determining an appropriate substring length from the parent chromosome for performing crossover. While the position of the substring in the parent chromosome is chosen randomly, the length of the substring is predetermined. In the second part of the investigation, the effectiveness of the new operators for solving TSP is established. Finally, in the third part the microarray gene ordering problem is considered. Comparison of the proposed genetic operators is carried out with other techniques based on GAs, neural networks and clustering in terms of a biological score.

In Section 2.2 we provide, in brief, a formal definition of TSP and relevance of TSP in microarray gene ordering. The different components of GAs along with their implementation for solving TSP are discussed in Section 2.3. New operators such as NF and MOC, and the algorithm based on them for TSP and gene ordering are described in Section 2.4. In Section 2.5 the results obtained with our algorithm for different TSP instances and microarray data sets are presented. Section 2.6 concludes the investigation. The new operators and some of the results presented in this chapter have been reported in [88, 91].

2.2 TSP Definition and Relevance in Microarray Gene Ordering

Investigations for clustering and then ordering gene expression profiles include hierarchical clustering [14, 18, 31], self-organizing maps (SOM) [31] and evolutionary algorithms [26]. It is mentioned in Section 1.6.4 that, outside the framework of clustering, the TSP, with some minor modifications, can be used to model the microarray gene ordering (MGO) problem. Beidl et al. [18] formulated the MGO problem as TSP using GA and Tsai et al. [115] applied family competition GA (FCGA) for solving it. They associated one imaginary city to each gene, and obtain the distance between any two cities (genes) from the matrix of inter gene distances. For microarray gene ordering it is necessary to minimize the distance between the genes that are in the neighborhood of each other, not the distant genes. However, Tsai et al. tried to minimize the distance between distant genes as well [64, 113]. This problem for TSP formulation in microarray gene ordering using GA is minimized in NNGA [64], where relatively long distances between genes are ignored for fitness evaluation.

Let $\{1, 2, \cdots, n\}$ be the labels of the n cities and $C = [c_{i,j}]$ be an $n \times n$ cost matrix where $c_{i,j}$ denotes the cost of traveling from city i to city j. The Traveling Salesman Problem (TSP) is the problem of finding the shortest closed route among n cities, having as input the complete distance matrix among all cities. The total cost A of a TSP tour is given by

$$A(n) = \sum_{i=1}^{n-1} C_{i,i+1} + C_{n,1} \tag{2.1}$$

The objective is to find a permutation of the n cities, which has minimum cost.

A unidirectional optimal gene order, a minimum sum of distances between pairs of adjacent genes in a linear ordering $1, 2, \cdots, n$, can be formulated as [18]

$$F(n) = \sum_{i=1}^{n-1} C_{i,i+1}, \tag{2.2}$$

where n is the number of genes and $C_{i,i+1}$ is the distance between two genes i and $i+1$. The formula (Eqn. 2.2) for optimal gene ordering is the same as used in TSP, except the distance from the last gene to first gene, which is omitted, as the tour is not a circular one. In this study, the Euclidean distance is used to specify the distance $C_{i,i+1}$.

Let $X = x_1, x_2, \cdots, x_k$ and $Y = y_1, y_2, \cdots, y_k$ be the expression levels of the two genes X and Y in terms of log-transformed microarray gene expression data obtained over a series of k experiments. The Euclidean distance between X and Y is

$$E_{X,Y} = \sqrt{\{x_1 - y_1\}^2 + \{x_2 - y_2\}^2 + \cdots + \{x_k - y_k\}^2}. \tag{2.3}$$

One can thus construct a matrix of inter-gene distances, which serves as a knowledge-base for mining gene order using GA. Using this matrix one can calculate the total distance between adjacent genes and find that permutation of genes for which the total distance is minimized. This is analogous to the traveling salesman problem. One can simply associate one imaginary city to each gene, and obtain the distance between any two cities (genes) from the matrix of inter gene distances. The formula (Eqn. 2.2) for optimal gene ordering is the same as used in TSP, except the distance from the last gene to first gene, which is omitted, as the tour is not a circular one.

2.3 Genetic Algorithms for Solving TSP and MGO

Genetic algorithms (GAs) [45] are randomized search and optimization techniques guided by the principles of evolution and natural genetics, and have a large amount of implicit parallelism. GAs perform multimodal search in complex landscapes and provide near optimal solutions for objective or fitness function of an optimization problem. In GAs, the parameters of the search space are encoded in the form of strings (chromosomes). A collection of such strings is called a population. Initially a random population is created, which represents different points in the search space. Based on the principle of survival of the fittest, a few among them are selected and each is assigned a number of copies that go into the mating pool. Biologically inspired operators like mating, crossover, and mutation are applied on these strings to yield a new generation of strings. The process of selection, crossover and mutation continues for a fixed number of generations or until a termination condition is satisfied. A general description of Genetic Algorithm is presented in this section for solving TSP using elitist model. Roughly, a genetic algorithm works as follows (see Figure 2.1):

2.3.1 Chromosome Representation and Nearest-Neighbor Heuristic

Various representation methods are used to solve the TSP problem using GA. Some of these are binary representation, path representation, matrix representation, adjacency representation, ordinal representation [61]. In order to find the shortest tour for a given set of n cities using GAs, the path representation is more natural for TSP [61]. We have used this representation in our proposed GA. In path representation, the chromosome (or, string) corresponding to a TSP tour is an array of n integers which is a permutation of $(1, 2, \cdots\cdots, n)$, where an entry i in position j indicates that city i is visited in the jth time instant. The objective is to find a string with minimum cost.

```
begin GA
    Create initial population
    while generation_count < $k$ do
        /* $k$ = max. number of generations. */
        begin
            Selection and Elitism
            Produce children by crossover from
                                  -selected parents
            Mutate the individuals
            Increment generation_count
        end
        Output the best individual found
end GA
```

Figure 2.1: The Pseudo-code of Genetic Algorithm (GA)

For solving TSP, the nearest neighbor tour construction heuristic is a common choice to construct the initial population for its $O(n^2)$ time complexity. The salesman starts at some random city and then visits the city nearest to the starting city. From there he visits the nearest city that was not visited so far, until all the cities are visited, and the salesman returns to the starting city. The NN tours have the advantage that they only contain a few severe mistakes, while there are long segments connecting nodes with short edges. Therefore such tours can serve as good starting tours for subsequent refinement using other sophisticated search methods. In NN the main disadvantage is that, several cities are not considered during the course of the algorithm and have to be inserted at high costs in the end. This leads to severe mistakes in path construction. To overcome the disadvantages of the NN heuristics, we propose a new heuristic operator, called the Nearest Fragment (NF) operator (discussed in Section 2.4.1). However, unlike NN heuristic that is used only for constructing the initial population, NF is used in every generation (iteration) of GA with a predefined probability for every chromosome in the population as a subsequent tour improvement method.

2.3.2 Selection and Elitism

A number of different selection implementations have been proposed in the literature [45], such as roulette wheel selection, tournament selection, linear normalization selection. Here linear normalization selection, which has a high selection pressure [45], has been implemented. In linear normalization selection, an individual is ranked according to its fitness, and then it is allowed to generate a number of offspring proportional to its rank position. Using the rank position rather than the actual fitness values avoids problems that occur when fitness values are very close to each other (in which case no individual would be favored) or when an extremely fit individual is present in the population (in such a case it would generate most of the offspring in the next generation). This selection technique pushes the population toward the solution in a reasonably fast manner, avoiding the risk of a single individual dominating the population in the space of one or two generations.

A new population is created at each generation (iteration) and after selection procedure, chromosome with highest fitness (least cost) from the previous generation replaces randomly a chromosome from this new generation provided fitness of the fittest chromosome in the previous generation is higher than the best fitness in this current generation in the elitist model.

2.3.3 Crossover

As the TSP is a permutation problem, it is natural to encode a tour by enumerating the city indices in order. This approach has been dominant in GAs for solving the TSP. In such an encoding, the chromosomal location of a city is not fixed, and only the sequence is meaningful. Some representative crossovers performed on order-based encodings include cycle crossover [76], partially matched crossover [45] and order crossover [28, 45]. Order crossover has been found to be one of the best in terms of quality and speed [61], and yet is simple to implement. Below order crossover is described briefly.

Order Crossover (OC): The order crossover operator [28, 45] selects at random a substring in one of the parent tours, and the order of the cities in the selected positions of this parent is imposed on the other parent to produce one child. The other child is generated in an analogous manner for the other parent. As an example consider two parents A and B, and a substring in A of length 3, selected randomly, as shown [45].

$A = 1\ 2\ 3\ |5\ 6\ 7|\ 4\ 8\ 9\ 0$

and

$B = 8\ 7\ 1\ |2\ 3\ 0|\ 9\ 5\ 4\ 6$

The cities in the selected substring in A (here, 5, 6, and 7) are first replaced by * in the receptor B.

$A = 1\ 2\ 3\ |5\ 6\ 7|\ 4\ 8\ 9\ 0$

and

$B = 8 * 1\ |2\ 3\ 0|\ 9 * 4 *$

Now to preserve the relative order in the receiver, a sliding motion is made to leave the holes in the matching section marked in the receiver. The convention followed in [45] is to start this sliding motion in the second crossover site, so after the rearrangement we have

$A = 1\ 2\ 3\ |5\ 6\ 7|\ 4\ 8\ 9\ 0$

and

$B = 2\ 3\ 0\ | * * * |\ 9\ 4\ 8\ 1$

After that, the stars are replaced with the city names taken from the donor A resulting in the offspring B1

$B1 = 2\ 3\ 0\ |5\ 6\ 7|\ 9\ 4\ 8\ 1$

Similarly the complementary crossover from B to A yields

$A1 = 5\ 6\ 7\ |2\ 3\ 0|\ 4\ 8\ 9\ 1$

In order crossover (OC) the length of the substring for crossover (chosen from the parent string) is random and may often be significantly large; this can have an adverse impact on the computational time. This uncertainty is tackled with a small and predefined length of substring, obtained after extensive empirical studies, for crossover (discussed in Section 2.4.2).

2.3.4 Mutation

For TSP, the simple inversion mutation (SIM) is one of the leading performers [61]. Here simple inversion mutation (SIM) is performed on each string probabilistically as follows: Select randomly two cut points in the string, and reverse the substring between these two cut points. For example consider the tour

(1 2 |3 4 5| 6 7 8)

and suppose that the first cut point is chosen randomly between 2nd city and 3rd city, and the second cut point between the 5th city and the 6th city as shown. Then the resulting string will be

$C = (1\ 2\ |5\ 4\ 3|\ 6\ 7\ 8)$

2.4 New Operators of GAs

In this section, some new operators of GAs for solving TSP and microarray gene ordering are described. These are nearest fragment (NF) and modified order crossover (MOC). The genetic algorithm designed using these operators is referred to as FRAG_GA. The structure of the proposed FRAG_GA is provided in Figure 2.2.

```
begin  FRAG_GA
    Create initial population with Nearest-Neighbor Heuristic
    while generation_count < $k$  do
          /* $k$ = max. number of generations. */
          begin
              Apply NF heuristic or (NF and LK) heuristic
              Elitism
              Linear Normalized Selection
              MOC
              Mutation
              Increment generation_count
          end
          Output the best individual found
end  FRAG_GA
```

Figure 2.2: The Pseudo-code for FRAG_GA

The basic steps of the FRAG_GA are as follows:

S1) Create the string representation (chromosome of GA) for a TSP tour (an array of n integers), which is a permutation of $1, 2, \cdots\cdots, n$, with Nearest-Neighbor heuristic. Repeat this step to form the population of GA.

S2) Apply NF heuristic on each chromosome probabilistically.

S3) Upgrade each chromosome to local optimal solution using chained LK heuristic probabilistically. (If S3 is used in the GA we denote it as FRAG_GALK and otherwise as FRAG_GA.).

S4) Evaluate the fitness of the entire population and use elitism so that the fittest string among the child population and the parent population is passed into the child population.

S5) Apply linear normalized selection procedure by using the evaluated fitness of entire population.

S6) Distribute the chromosomes randomly and apply Modified Order Crossover operator between two consecutive chromosomes probabilistically.

S7) Apply simple inversion mutation (SIM) on each string probabilistically.

S8) Increment the generation count of GA and if it is less than the maximum number of generations (predefined) then repeat the steps from S2 to S6.

Local search heuristics, such as 2-swap, 2-opt [53], 3-opt [53], and Lin-Kernighan (LK) heuristic [6, 48, 65, 114], have been extensively applied in GAs for solving TSPs. These techniques exchange some edges of parents to generate new children. Usually, stronger local search methods correspond to better performing GAs. The mechanisms by which these methods add and preserve edges vary. 2-swap arbitrarily changes two cities at a time, removing four edges at random and adding four edges at random. 2-opt, 3-opt and LK exchange edges if the generated solution is better than the original one. In each iteration, 2-opt and 3-opt exchange two and three edges respectively, while, LK exchanges a variable number of edges. In the present investigation Concorde version of chained-LK [5] is used for fair comparison with [114]. In the following sections, the new operators NF and MOC are described in details.

2.4.1 Nearest Fragment Heuristic (NF)

In this process, each string (chromosome in GA) is randomly sliced in $frag$ fragments. The value of $frag$ is determined by FRAG_GA in terms of the total no. of cities/genes (n) for a particular TSP instance (or microarray data). The systematic process of determining $frag$ is described later in this section. As an example, let us consider a string P that is sliced into three random fragments (1-8), (9-14) and (15-20) for a 20-city problem.

$P = 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ |9\ 10\ 11\ 12\ 13\ 14|\ 15\ 16\ 17\ 18\ 19\ 20$

For tour construction the first fragment (9-14) is chosen randomly. From the last city of that fragment (14) the nearest city that is either a start or an end point of a not yet visited tour fragment is determined from the cost matrix. In this example, let the nearest city (among 1, 8, 15 and 20) be 20. The fragment containing the nearest city is connected to the selected fragment, with or without inversion depending on whether the nearest city is the last city of a fragment or not respectively. In this example , the fragment 15-20 is inverted and connected to fragment 9-14, resulting in the following partial tour $P1$.

$P1 = 9\ 10\ 11\ 12\ 13\ 14\ |20\ 19\ 18\ 17\ 16\ 15$

The process is repeated until all fragments have been reconnected. From the last city (15) of $P1$ the nearest city from unvisited fragment (1-8) is say 1. From this result the final string $P2$, shown below, is formed.

$P2 = 9\ 10\ 11\ 12\ 13\ 14\ 20\ 19\ 18\ 17\ 16\ 15\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8$

The basic steps of choosing frag value systematically for a TSP instance with n cities are :

S1) Set $frag$ value to $frag_{min}$.

S2) Run FRAG_GA with the selected $frag$ value for x generations and store the number of times the best tour cost is decreased from one generation to the next for that $frag$ value. Denote the stored values by $Decr_{cost}$.

S3) Increase $frag$ by amount $\Delta frag$.

S3) Repeat S2 to S3 until $frag <= frag_{max}$

S4) Find θ consecutive $frag$ values for which the summation of corresponding $Decr_{cost}$ values is maximum.

S5) The best $frag$ value is set to the average of the selected five consecutive $frag$ values.

S6) Repeat S1 to S5 ten times and the average of the best $frag$ values is fixed as the final $frag$ value for NF for a particular TSP instance.

In this study we have used $frag_{min} = \frac{n}{16}$, $frag_{max} = \frac{n}{2}$, $x = \frac{n}{10}$, $\Delta frag = \frac{n}{50}$, and $\theta = 5$, though experiments were conducted for a few other values as well with similar results. The value of $frag_{max}$ is not set to n as this will lead to NN heuristic. Also, the crossover operator was disabled. The motivation for setting the initial $frag$ value to a low one (and consequently fragment lengths are larger) and then increasing it is that, first exploring the distant neighbors reduces the chances of locking at a local optimal tour for the GA. The probabilistic use of NF also helps to come out from local optimal solution by leaving some chromosomes for mutation and crossover operators to explore.

As an example, consider a 100 city problem with $frag = \frac{n}{16}$, and consequently the fragment length is 16 on an average. As the initial population of the FRAG_GA is formed with NN heuristic there is a likelihood that a city at one end of a fragment is close to the 16th neighbor of the similar end of the next/previous fragment and consequently, they may be connected by the NF heuristic. In the later generations of the GA, using $\frac{n}{16}$ fragments in NN heuristic, explores on an average, from any city to the 16th city in the chromosome rather than the 16th neighbor. Due to random slicing of the chromosome, some fragment lengths will be obviously greater than 16 and some less than 16, and consequently different types of neighbors will be considered. The lowest $frag$ value is set to $\frac{n}{16}$ from the studies in [6], where it is mentioned that good/optimal results are obtained for most of the TSP instances in the TSP library [116] with a search space near about 16 neighbors. The more distant neighbors are mostly explored with mutation and crossover operators.

2.4.2 Modified Order Crossover (MOC)

As already mentioned, in order crossover the length of a substring is chosen randomly and can lead to an increase in the computational time, this uncertainty can be minimized if the length of the substring for performing crossover can be fixed to a small value. However, no study has been reported in the literature for determining an appropriate value of the length of a substring for performing order crossover. Such an attempt is made in this article for finding a small substring length for MOC that provides good results for TSP/microarray data with the lowest computational cost.

Unlike order crossover, where the substring length is randomly chosen, in MOC it is determined automatically by the FRAG_GA in a similar way of choosing $frag$ value in Section 2.4.1. In the process of choosing appropriate substring length, NF heuristic is also present in FRAG_GA with its final $frag$ value. As final frag value for NF is determined without any crossover operator, it is preferable to start the process of choosing substring length for MOC initially with a very small value like $\frac{n}{32}$ (very close to no MOC) and then increasing it. For example, for a 10 city problem let the systematically chosen substring length by FRAG_GA is 2. Now for the parents A and B the chromosomes may be as follows

$A = 0\ 9\ 8\ 4\ |5\ 6|\ 7\ 3\ 2\ 1$

and

$B = 9\ 5\ 4\ 1\ |2\ 3|\ 0\ 6\ 8\ 7$

The cities in the selected substring in A (here, 5 and 6) are first replaced by * in the receptor B.

$B = 9 * 4\ 1\ |2\ 3|\ 0 * 8\ 7$

Now to preserve the relative order in the receiver, the convention followed in [45] is to gather the holes in the second crossover site and insert the substring there. But this convention leads to loss of information and increases randomness in the receiver because, after insertion of substring 56 in B neither 5 is nearer to 3, nor 6 is nearer to 0. To reduce this randomness, in MOC the holes are gathered in the position of the last deleted city (here city 6) of the receiver B.

$B = 9\ 4\ 1\ 2\ 3\ 0\ | * *|\ 8\ 7$

So after substring insertion, B is as follows:

$B = 9\ 4\ 1\ 2\ 3\ 0\ 5\ 6\ 8\ 7$

Now, at least one edge of the substring is nearer to the next city (city 6 is nearer to 8 according to chromosome B, and this information is preserved). Same convention is followed for inserting substring in chromosome A.

2.5 Experimental Results

FRAG_GA is implemented in C on Pentium-4 (1.2 GHz) and the results are compared with those obtained using SWAP_GATSP [87], MMGA [112], IGA [52], OX_SIM (standard GA with order crossover and simple inversion mutation [61]), MOC_SIM (Modified order crossover and SIM), and self organizing map (SOM) [9] for solving TSP. For fair comparison with the above mentioned methods Lin-Kernighan (LK) heuristic is not used with FRAG_GA, whereas, for comparison with HeSEA [114] and other LK based methods each chromosome in FRAG_GALK is updated probabilistically with 20 runs of consecutive chained LK and mutation (as recommended in [114]). Several benchmark TSP instances, for which the comparative study with various recently developed pure genetic algorithms (without LK), and SOM are available in the literature, are taken from the TSPLIB [116] without any bias on data sets. These include Grtschels24.tsp, bayg29.tsp, Grtschels48.tsp, eil51.tsp, St70.tsp, eil76.tsp, kroA100.tsp, d198.tsp, ts225.tsp, pcb442.tsp and rat783. tsp. For comparative study between HeSEA, FRAG_GALK, and other LK based methods the available TSP instances are lin318, rat783, pr1002, vm1084, pcb1173, u1432, u2152, pr2392, pcb3038, fnl4461, and usa13509. For biological microarray gene ordering, Cell Cycle cdc15, Cell Cycle and Yeast Complexes datasets are chosen [120]. The three data sets consists of about 782, 803 and 979 genes respectively, which are cell cycle regulated in Saccharomyces cerevisiae, with different number of experiments (24, 59 and 79 respectively) [105]. Each dataset is classified into five groups termed G1, S, S/G2, G2/M, and M/G1 by Spellman et. al. [105]. Results are compared with those obtained using GAs [64,113], different versions of hierarchical clustering [18,31] and self-organizing map (SOM) [109] for solving microarray gene ordering. Throughout the experiments for FRAG_GA, SWAP_GATSP, OX_SIM, and MOC_SIM the population size is set equal to 100. Crossover probability is fixed at 0.6 and mutation probability is fixed at 0.02 across the generations. For FRAG_GA and FRAG_GALK the probability of applying NF heuristic is fixed at 0.3. Table 2.1 shows the various parameters of different genetic algorithms used in this current investigation.

Table 2.1: Different parameters of FRAG_GA, SWAP_GATSP, OX_SIM, and MOC_SIM

population size	NF Probability for FRAG_GA	Crossover Probability	Mutation Probability
100	0.3	0.6	0.02

First, we provide results comparing our method (FRAG_GA) with other methods that do not use LK heuristics and then comparisons of results are provided with our method incorporating LK heuristic (FRAG_GALK) with other LK based methods .

2.5.1 Comparison with Other GA Approaches for TSP

Table 2.2: Comparison of the results over 30 runs obtained using FRAG_GA, SWAP_GATSP, OX_SIM, and MOC_SIM for different TSP instances

Problem		FRAG_GA	SWAP_GATSP	OX_SIM	MOC_SIM
Grtschels24	best	1272 (13)	1272 (50)	1272 (800)	1272 (600)
24	average	1272 (100)	1272 (200)	1322 (1500)	1272 (1500)
1272	error(%)	0.0000	0.0000	3.9308	0.0000
Bayg29	best	1610 (30)	1610 (60)	1620 (1000)	1610 (700)
29	average	1610 (100)	1615 (200)	1690 (1500)	1622 (1500)
1610	error(%)	0.0000	0.3106	4.9689	0.7453
Grtschels48	best	5046 (40)	5046 (200)	5097 (2500)	5057 (1700)
48	average	5054 (150)	5110 (700)	5410 (3000)	5184 (3000)
5046	error(%)	0.1585	1.2683	7.2136	2.7348
eil51	best	426 (45)	439 (220)	493 (2500)	444 (1600)
51	average	432 (150)	442 (700)	540 (3000)	453 (3000)
426	error(%)	1.4085	3.7559	26.7606	6.3380
St70	best	675 (40)	685 (600)	823 (4500)	698 (4500)
70	average	679 (150)	701 (1000)	920 (7500)	748 (7500)
675	error(%)	0.5926	3.8519	36.2963	10.8148
eil76	best	538 (75)	548 (700)	597 (5000)	562 (3800)
76	average	544 (150)	555 (1000)	620 (7500)	580 (7500)
538	error(%)	1.1152	3.1599	15.2416	7.8067
KroA100	best	21282 (80)	21397 (2000)	21746 (10000)	21514 (8200)
100	average	21303 (500)	21740 (3000)	22120 (12000)	21825 (12000)
21282	error(%)	0.0987	2.1521	3.9376	2.5515
d198	best	15780 (850)	15980 (4000)	16542 (10000)	16122 (9000)
198	average	15865 (2000)	16106 (4000)	17987 (16000)	16348 (16000)
15780	error(%)	0.5387	2.0659	13.9861	3.5995
ts225	best	126643 (1000)	127012 (4000)	135265 (10000)	128994 (10000)
225	average	126778 (2000)	128467 (4000)	138192 (16000)	130994 (16000)
126643	error(%)	0.1066	1.4403	9.1193	3.4356
pcb442	best	50778 (1900)	52160 (8000)	53320 (16000)	52852 (13000)
442	average	50950 (4000)	53800 (8000)	56330 (26000)	54173 (26000)
50778	error(%)	0.3387	5.9514	10.9339	6.6860
rat783	best	8850 (7500)	9732 (12000)	10810 (28000)	10155 (20000)
783	average	9030 (16000)	10087 (16000)	11136 (40000)	10528 (40000)
8806	error(%)	2.5437	14.5469	26.4592	19.5548

Table 2.2 summarizes the results obtained over 30 runs by running the FRAG_GA, SWAP_GATSP [87], OX_SIM and MOC_SIM [61] on the aforesaid eleven different TSP instances. For SWAP_GATSP, OX_SIM and MOC_SIM, the overlapping pa-

rameters (Table 2.1) are taken from FRAG_GA. For each problem the total number of cities and the optimal tour cost are mentioned below the problem name in the first column. The total number of generations in which the best result and the average result are obtained is mentioned in columns 3-6 within parentheses. The error percentages are shown in third row for each problem, where the error percentage is defined as

$$E = \frac{average - optimal}{optimal} \times 100. \qquad (2.4)$$

Experimental results (Tables 2.2-2.3) using FRAG_GA are found to be superior in terms of quality of solution (best result, average result and error percentage) with less number of generations when compared with those of other existing GAs [52, 61, 87, 112]. It is evident from the table that for different TSP instances the error percentages are lowest for FRAG_GA and the error percentages for MOC_SIM is much less than OX_SIM . The average of error percentages over all the TSP instances for MOC_SIM is 5.8425, which is also less than 14.4407 of OX_SIM. The error averages clearly indicates that the modification of order crossover improves its performance significantly over the existing order crossover which uses random substring length and its random insertion. The average of error percentages over all the TSP instances for FRAG_GA and SWAP_GATSP are 0.6274 and 3.5003 respectively.

Figure 2.3 shows a comparison of FRAG_GA, SWAP_GATSP and OX_SIM when the fitness value of the fittest string is plotted with iteration. The three programs were run for 12000 iterations for kroa100.tsp with population 100. At any iteration, the FRAG_GA has the lowest tour cost. It took 15.36 seconds, 19.94 seconds and 15.14 seconds by FRAG_GA, SWAP_GATSP and OX_SIM respectively for executing 12000 iterations. Moreover, only FRAG_GA is seen to converge at around 250 iterations at the optimal cost value of 21282 km. On the other hand, the cost is 21397 km for SWAP_GATSP after 3100 iterations and 21990 km for OX_SIM even after 12000 iterations.

Note that FRAG_GA takes almost the same time as OX_SIM using one more operator (NF), but provides better result in less number of paths. It is further to be pointed out that the NF operator creates an overhead, leading to an increase in the computation time for FRAG_GA, as compared to OX_SIM. However, this is compensated by the gain obtained in using the proposed MOC operator. As a

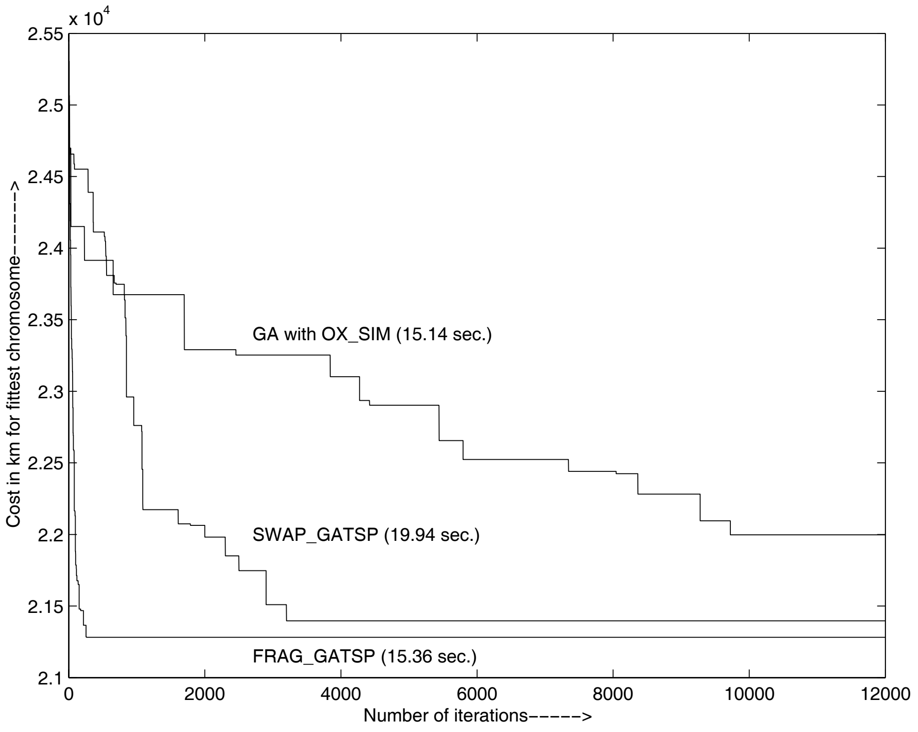


Figure 2.3: Cost of fittest string Vs. Iteration for kroa100.tsp

consequence, the time required to execute one iteration, on an average, becomes almost equal for both FRAG_GA and OX_SIM. Similar observations are also made when the proposed method is compared with other GAs [61,87] and other methods like Self Organizing Map (SOM) [9].

Table 2.3: Average results for various GAs

Problem	Optimal	IGA	MMGA	SOM	FRAG_GA
eil51	426	499	446	432	432
st70	675	–	–	683	679
eil76	538	611	568	556	544
Kroa100	21282	24921	22154	–	21303
d198	15780	17925	16360	–	15865
ts225	126643	135467	129453	–	126778
pcb442	50778	59380	55660	55133	50950

In Table 2.3 average results of FRAG_GA are compared to other GA based approaches viz., IGA and MMGA (whose results are taken from [112]) and Self Organizing Map (SOM) [9]. As can be seen from the table, the proposed approach is again found to consistently outperform IGA, MMGA, and SOM.

2.5.2 Comparison with Other LK Based Approaches for TSP

Table 2.4 summarizes the results obtained over 20 runs by running the FRAG_GALK on different TSP instances mentioned in first column. 20 runs of LK [5] and mutation are applied on randomly chosen 50 chromosomes (among those who are not operated with NF heuristic) in each generation of FRAG_GALK. For HeSEA (with LK) [114], LKH (Multi-trial LK) [48], iterated LK (ILK) [53], and tabu search with LK [128] the results are taken from [114]. While FRAG_GALK, HeSEA, LKH, and

Table 2.4: Average results for various LK based algorithms

Problem		FRAG _GALK	HeSEA +LK	LKH	Concorde	ILK	Tabu +LK
lin318	error(%)	0.0000	0.0000	0.1085	0.0000	–	–
318	generation	2.8	3.2	–	–	–	–
42029	time(sec.)	1.4	2.3	1.4	1.4	–	–
rat783	error(%)	0.0000	0.0000	0.0000	0.1761	–	–
783	generation	8.0	8.4	–	–	–	–
8806	time(sec.)	5.9	39.1	2.2	5.9	–	–
pr1002	error(%)	0.0000	0.0000	0.0000	0.0215	0.1482	0.8794
1002	generation	22.2	12.0	–	–	–	–
259045	time(sec.)	34.6	91.0	7.5	34.6	298.0	1211.4
vm1084	error(%)	0.0000	0.0000	0.0068	0.0172	0.0217	0.3932
1084	generation	23.6	10.2	–	–	–	–
239297	time(sec.)	34.2	80.6	12.6	34.2	377.0	597.0
pcb1173	error(%)	0.0000	0.0000	0.0009	0.0070	0.0088	0.6996
1173	generation	22.5	11.5	–	–	–	–
56892	time(sec.)	38.7	84.5	11.8	39.0	159.0	840.0
u1432	error(%)	0.0000	0.0000	0.0000	0.0153	0.0994	0.4949
1432	generation	22.0	11.0	–	–	–	–
152970	time(sec.)	37.7	107.0	6.9	38.0	224.0	775.0
u2152	error(%)	0.0000	0.0000	0.0495	0.0242	0.1743	0.7517
2152	generation	31.5	17.5	–	–	–	–
64253	time(sec.)	48.3	211.0	135.0	49.0	563.0	1624.0
pr2392	error(%)	0.0000	0.0000	0.0000	0.0294	0.1495	0.6492
2392	generation	25.0	14.5	–	–	–	–
378032	time(sec.)	46.6	208.0	26.2	47.0	452.0	1373.0
pcb3038	error(%)	0.0000	0.0000	0.0068	0.1123	0.1213	0.8708
3038	generation	120.6	29.7	–	–	–	–
137694	time(sec.)	245.0	612.0	226.0	219.0	572.0	1149.0
fnl4461	error(%)	0.0014	0.0005	0.0027	0.0734	0.1358	0.9898
4461	generation	265.0	67.8	–	–	–	–
182566	time(sec.)	519.0	2349.0	528.0	519.0	889.0	1018.0
usa13509	error(%)	0.0061	0.0074	0.0065	0.1201	0.1638	0.8897
13509	generation	1102.5	223.0	–	–	–	–
19982859	time(sec.)	19203.0	34984.0	19573.0	19203.0	10694.0	5852.0

concorde chained LK (concorde) [5] are executed on Pentium-4 (1.2 GHz) personal computer, ILK and tabu search with LK are executed on Silicon Graphics 196 MHz MIPS R1000 and Pentium III 800 MHz respectively in [114]. For fair comparison Concorde chained LK is executed separately for same time as FRAG_GALK, but on average concorde converged to the mentioned solutions (in terms of error) before the allocated time. The total number of cities and the optimal tour cost are mentioned below the problem name in the first column. The error percentages (Equation 2.4) are shown in first row for each problem. The average number of generations over 20 runs for which the error percentages are obtained is mentioned in second row for each TSP instance. The third row for each TSP instance shows the average time in seconds taken by each method. From the table it is clear that FRAG_GALK produces comparable results with HeSEA with same version of LK in less computational time, whereas the quality of solution of FRAG_GALK is better than other algorithms with comparable computational time. So FRAG_GALK seems to be a better TSP solver among the existing ones. The time gain obtained by FRAG_GALK over HeSEA is due to probabilistic single run of computationally effective NF heuristic and MOC over each chromosome in FRAG_GALK, whereas, HeSEA uses 20 runs of edge-assembly crossover between the selected chromosomes and for all possible combinations of chromosomes with probability 1. Generations of LKH, ILK, and tabu with LK are not available.

2.5.3 Results for Microarray Gene Ordering

FRAG_GA is applied for ordering the genes based on their expression levels obtained from microarray datasets. Performance of FRAG_GA for gene ordering is compared with other methods based on GAs, clustering and neural networks. GA based investigations include NNGA [64] and FCGA [113].

As already mentioned, clustering methods can be broadly divided into hierarchical and nonhierarchical clustering approaches. Hierarchical clustering approaches [18, 31] group gene expressions into trees of clusters. They start with singleton sets and keep on merging the closest sets until all the genes form a single cluster. Complete-linkage and average-linkage belong to this category of clustering technique, differing only in the way the distance between clusters is defined. Nonhierarchical clustering approaches separate genes into groups according to the degree of similarity (as quantified by Euclidian distances, Pearson correlation) among genes. The relationships among the genes in a particular cluster generated by nonhierarchical clustering methods are lost. Self-organizing map (SOM) [109], a particular class of neural network, performs nonhierarchical clustering.

Table 2.5: Comparison of the results over 30 runs in terms of sum of gene expression distances for microarray data using various algorithms

Algorithms	Cell cycle cdc15		Cell cycle		Yeast complexes	
	Best	Average	Best	Average	Best	Average
FRAG_GA	1272 (1690)	1278 (4000)	2349 (2320)	2362 (4000)	3382 (3890)	3396 (6000)
Complete-linkage	1419	1419	2534	2534	3634	3634
Average-linkage	1433	1433	2559	2559	3681	3681
SOM	1874 (100000)	1905 (100000)	3018 (100000)	3094 (100000)	4376 (200000)	4449 (200000)

Table 2.5 summarizes the results in terms of the sum of gene expression distances (Eqn. 2.2), by executing the FRAG_GA, complete linkage, average linkage and SOM on the three different microarray datasets, described in the first paragraph of Section 2.5. Results (in terms of sum of gene expression distance) and code for NNGA and FCGA are not available in the literature [64, 113]) and hence not provided in Table 2.5. For FRAG_GA and SOM, best and average results obtained over 30 runs are provided, whereas, for complete and average linkage results remain same for all runs. The genetic parameters for FRAG_GA are the same as used before (see Table 2.1). For FRAG_GA and SOM the total number of generations/iterations, for which the best and average results are obtained are mentioned in columns 2-7 within parentheses. From the table it is clear that FRAG_GA produces superior gene ordering than related methods in terms of sum of the gene expression distances.

A biological score, that is different from the fitness function, is used to evaluate the final gene ordering. The biological score is defined as [113]

$$S(n) = \sum_{i=1}^{n-1} s_{i,,i+1} \text{ where } s_{i,,i+1} = 1, \text{ if gene } i \text{ and } i+1 \text{ are in the same group}$$
$$= 0, \text{ if gene } i \text{ and } i+1 \text{ are not in the same group}$$

Using this, a gene ordering would have a higher score when more genes within the same group are aligned next to each other. So higher values of $S(n)$ indicate better gene ordering. For example consider the genes YML120C, YJR048W, YMR002W and YDR432W belonging to groups G2/M, S/G2, S/G2 and G2/M respectively.

Table 2.6: Comparison of the best results over 30 runs in terms of S(n) values for microarray data

Algorithms	Cell cycle cdc15	Cell cycle	Yeast complexes
FRAG_GA	540	635	384
NNGA	539	634	384
FCGA	521	627	− − −
Complete-linkage	498	598	340
Average-linkage	500	581	331
SOM	461	578	306

In the above-mentioned ordering they will return a biological score of 0+1+0=1, whereas if they are ordered like YJR048W, YMR002W, YDR432W and YML120C then the score will be 1+0+1=2. The scoring function is therefore seen to reflect well the order of genes in biological sense. Note that, although $S(n)$ provides a good quantitative index for gene ordering, using it as the fitness function in GA based ordering is not practical, since the information about gene categories is unknown for most of the genes in the real world .

Table 2.6 shows the best results over 30 runs of the above methods in terms of $S(n)$ value, where larger values are better ($S(n)$ values for NNGA are FCGA are taken from [64]). It is clear that FRAG_GA and NNGA [64] are comparable and they both dominate others. Note that FRAG_GA is a conventional GA, while NNGA (hybrid GA) is a one using LK heuristic [65]. The main reason for the good results obtained by FRAG_GA is that, biological solutions of microarray gene ordering lie in more than one sub optimal point (in terms of gene expression distance) rather than one optimal point and there exists different gene orders with same biological score.

2.6 Discussion and Conclusions

A new "nearest fragment operator" (NF) and a modified version of order crossover operators (MOC) of GAs are described along with demonstrating their suitability for solving both TSP and microarray gene ordering (MGO) problem. A systematic method for determining the appropriate number of fragments in NF and appropriate substring length in terms of the number of cities/genes in MOC are also provided. These newly designed genetic operators showed superior performance on both TSP and gene ordering problem. The said operators are capable of aligning more genes with the same group next to each other compared to other algorithms, thereby producing better gene ordering. In fact, in terms of biological score, FRAG_GA produces comparable and sometimes even superior results than NNGA, a GA which implements Lin-Kernighan local search, for solving MGO problem.

An advantage of FRAG_GALK is that the quality of the solution seems to be more stable than that obtained by LKH and concorde chained LK, when used to solve the benchmark TSP problems. An evolutionary algorithm for solving combinatorial optimization problems should comprise mechanisms for preserving good edges and inserting new edges into offspring, as well as mechanisms for maintaining the population diversity. In the proposed approach, nearest fragment heuristic, modified order crossover, and LinKernighan local search preserve good edges and add new edges. The proposed method can seamlessly integrate NF, MOC, and LK to improve the overall search.

The present investigation indicates that incorporation of the new operators in FRAG_GA and LK in FRAG_GALK yields better results as compared to other pure GAs, Self Organizing Map, and related LK based TSP solvers. With its superior results in reasonable computation time FRAG_GALK can be considered as one of the state-of-the-art TSP solvers.

Chapter 3

Gene Ordering in Partitive Clustering using Microarray Expressions

3.1 Introduction

In Chapter 2, some new operators of genetic algorithm (GA) are developed for unidirectional microarray gene ordering. The GA with its new operators has been denoted as FRAG_GALK [91] and can be used for solving all the problems where, Traveling Salesman Problem (TSP) is the key for problem formulation. The present chapter deals with the tasks of ordering genes, using FRAG_GALK, within clusters obtained from a partitive clustering solution.

A key step in the analysis of microarray gene expression data is the identification of groups of genes that manifest similar expression patterns. This translates to the algorithmic problem of clustering and ordering of gene expression data. A clustering problem usually consists of elements and a characteristic vector for each element. A measure of similarity is defined between pairs of such vectors. (In gene expression, elements are usually genes, the vector of each gene contains its expression levels under each of the monitored conditions or time points, and similarity can be measured, for example, by the correlation coefficient between vectors.) While the goal of clustering in microarray analysis is to partition the genes into subsets, which are labelled clusters, the goal of gene ordering is discussed in detail in Section

1.6.4.

Clustering methods can be broadly divided into hierarchical and partitive clustering approaches. Hierarchical clustering approaches (e.g., single, complete and average linkage) [14, 18, 31, 56] group gene expressions into trees of clusters. Hierarchical clustering, however, has a number of shortcomings. It has been noted by statisticians that, hierarchical clustering suffers from lack of robustness, nonuniqueness, and inversion problems that complicate interpretation of the hierarchy [109]. The deterministic nature of hierarchical clustering can cause points to be grouped based on local decisions, with no opportunity to reevaluate the clustering. It is known that the resulting trees can lock in accidental features, reflecting idiosyncrasies of the agglomeration rule [109]. Partitive clustering approaches, such as K-means [49], self-organizing map (SOM) [109], CAST [16], and CLICK [100], separate genes into groups according to the degree of distance among genes. The shortcoming of partitive clustering is that, relationships among the genes in a particular cluster are lost. Integrating gene ordering with partitive clustering is an approach that is likely to overcome the above problem.

Clustering and ordering the genes into homogeneous groups using gene expression data has been shown to be useful in functional annotation, tissue classification and regulatory motif identification. Although there is a rich literature on gene ordering in hierarchical clustering framework [14, 18, 31], there is no work addressing and evaluating the importance of gene ordering for gene expression analysis in partitive clustering framework, to the best of our knowledge. Partitive clustering methods determine unique clusters but do not order genes within cluster and the relationships among the genes in a particular cluster are generally lost. To obtain this relationship among genes in clusters, we propose a novel hybrid method where the proposed gene ordering algorithm "FRAG_GALK" [91] (described in Chapter 2), is used to order genes in clusters obtained from partitive clustering solutions of CLICK [100], k-means [49] and Self Organizing MAP (SOM) [109]. For the purpose of comparison of the proposed hybrid method using FRAG_GALK with related methods, an existing Traveling Salesman Problem (TSP) solver Concorde_LP [5] using linear programming, and optimal leaf ordering by [14], are also used to order genes in a partitive clustering solution and in hierarchical clustering solution, respectively. The utility of the new hybrid algorithm using FRAG_GALK is shown in improving the quality of the clusters provided by any partitive clustering algorithm by,

- identification of subclusters within big clusters,
- grouping functionally correlated genes within clusters,
- the maximization of biological gene ordering using MIPS categorization, and
- using less computation time than those obtained by optimal leaf ordering in hierarchical clustering solution.

Comparability of the new hybrid algorithms using FRAG_GALK as compared to hybrid algorithms using Concorde_LP and B-Joseph's leaf ordering method are demonstrated by the biological scores (see Section 3.4) of gene order for four different data sets . The new hybrid algorithms using FRAG_GALK also requires less computation time and Random Access Memory (RAM) than original B-Joseph's method. Note that, FRAG_GALK and Concorde_LP can obtain the optimal order of cities to many TSPLIB instances; the largest having 13,509 and 15,112 cities, respectively. While FRAG_GALK is a Genetic Algorithm [77] based TSP solver, Concorde_LP is a linear programming based TSP solver and much slower than FRAG_GALK. The various steps used in FRAG_GALK are available in Chapter 2. The new hybrid algorithms and some of the results presented in this chapter have been reported in [86, 90].

3.2 Existing Approaches for Gene Expression Analysis

3.2.1 Distance Measure

The most popular and probably most simple measures for finding global distance (or similarity) between genes, using gene expression, are the Euclidean distance and Pearson correlation, a statistical measure of linear dependence between random variables.

Let $X = x_1, x_2, \cdots, x_k$ and $Y = y_1, y_2, \cdots, y_k$ be the expression levels of the two genes in terms of log-transformed microarray gene expression data obtained over a series of k experiments. While the Euclidean distance between gene X and gene Y is defined in Section 2.2, using Using Pearson correlation the distance between gene X and Y can be formulated as

$$C_{X,Y} = 1 - P_{X,Y} \tag{3.1}$$

where $P_{X,Y}$ represents the centered Pearson correlation and is defined as

$$P_{X,Y} = \frac{1}{k} \sum_{i=1}^{k} \left(\frac{x_i - \overline{X}}{\sigma_X} \right) \left(\frac{y_i - \overline{Y}}{\sigma_Y} \right) \tag{3.2}$$

where $\overline{X}$ and σ_X are the mean and standard deviation of the gene X, respectively.

The Pearson correlation has value between -1 and 1, with 1 indicating a linear relationship between the two vectors. The Manhattan (or Minkowski) distance [60] between gene X and gene Y is

$$M_{X,Y} = \sum_{i=1}^{k} |x_i - y_i|. \tag{3.3}$$

3.2.2 Gene Ordering Methods

Hierarchical clustering does not determine unique clusters. Thus the user has to determine which of the subtrees are clusters and which subtrees are only a part of a bigger cluster. So in the framework of hierarchical clustering a gene ordering algorithm helps the user to identify clusters by means of visual display and to interpret the data [14]. The importance of gene ordering is discussed in detail in Section 1.6.4. For partitive clustering based approaches, clusters are identified by the algorithm automatically and the solutions are robust and not sensible to noise like hierarchical clustering [109]. However, the relationships among the genes in a particular cluster generated by partitive clustering algorithms are generally lost. This relationship (closer or distant) among genes within clusters can be obtained using gene ordering approaches.

Ideally, one would like to obtain a linear order of all genes that puts similar genes close to each other; such that for any two consecutive genes the distance between them is small. As mentioned in section 2.2, gene ordering problem is similar to the Traveling Salesman Problem [77] where, cities are ordered instead of genes [18, 91, 115]. An optimal gene order can be obtained by minimizing the summation of gene expression distances (or maximizing summation of gene expression similarities) between pairs of adjacent genes in a linear ordering $1, 2, \cdots, n$. This can be formulated as described in Section 2.2.

A hybrid method (first clustering then ordering) for ordering genes for a hierarchical clustering solution is proposed in [14] and described in Section 1.6.4. In [26] the MGO problem is tackled in a memetic algorithm framework where representations and solutions take some ideas from the hierarchical clustering. The impact of different fitness function on the solution is also analyzed in [26].

3.3 Materials and Methods

3.3.1 Description of Data Sets

Table 3.1: Summary for different microarray data sets

Dataset	No. of genes	Category	Total experiments
Cell Cycle	652	MIPS 16	184
Yeast Complex	979	MIPS 16	79
All Yeast	6221	MIPS 18	80
Fibroblast	517	GO 1347	18
Herpes	106	GeneBank 5	21

In the present investigation, data sets like Cell Cycle [32], Yeast Complex [14,31], All Yeast [31, 119], Fibroblast [34] and Herpes [51] are chosen. Table 3.1 shows the name of the data sets, number of genes in each dataset, number of gene categories, and finally the total number of experiments performed for a particular dataset. A detailed description of these datasets are available in Section 4.3.2. The genes in the first three data sets of Saccharomyces cerevisiae are classified according to the top level classification (hierarchical structure) of Munich Information Center for Protein Sequences (MIPS) [38] categorization into 16, 16, and 18 groups respectively. According to the Gene Ontology (GO) annotation, the genes in the Fibroblast dataset are distributed in 1347 categories. Herpes virus genes are broadly assigned to five functional groups that are available in [51], and genes that could not be assigned to any of these five groups are designated unknown. After downloading, for each dataset missing gene expression values are predicted with LSimpute [8] (described in Section 1.6.1) and the order of gene expression vectors is randomized to remove initial gene order bias in the dataset.

3.3.2 Missing Value Prediction

It is mentioned in Section 1.6.1 that microarray experiments often produce multiple missing expression values, normally due to various experimental problems. As gene expression analysis requires a complete data matrix as input, the missing values have to be estimated in order to analyze the available data. Alternatively, genes with missing expression values can be removed until no missing values remain. However, for arrays with only a small number of missing values, it is desirable to estimate those values. For the subsequent analysis to be as informative as possible, it is essential that the estimates for the missing gene expression values are accurate [8]. In this thesis, all the genes with more than 50% missing gene expression values are first eliminated from the dataset. Thereafter, for the remaining genes, the missing gene expression values are estimated using LSimpute [8] software, described briefly in Section 1.6.1.

3.3.3 New Hybrid Algorithm for Ordering Genes in Partitive Clustering

A method of ordering genes for a partitive clustering solution is currently missing. Though gene ordering methods exist (described in Section 1.6.4 and 2.2), the utility and application of these methods to individual clusters of partitive clustering solution is not reported. An optimal gene order for a partitive clustering solution can also be obtained by minimizing the the summation of gene expression distances and we define it as

$$F_1(n) = \sum_{j=1}^{k} \sum_{i=1}^{n_j - 1} C_{i,i+1}^{j}, \tag{3.4}$$

where k is the total number of clusters, n_j is the number of genes in cluster j, and $C_{i,i+1}^{j}$ is the distance/similarity between two genes i and $i+1$ in cluster j obtained from distance/similarity matrix.

Here, we propose new hybrid algorithms using FRAG_GALK to improve the quality of the clusters provided by any partitive clustering algorithm. It is mentioned in Section 3.1 that, FRAG_GALK is applied separately on each of the gene clusters found by partitive clustering methods to identify subclusters within large clusters and to group the functionally correlated genes within clusters. The number of nodes/clusters of SOM and K-means are chosen according to MIPS categories (top level of the hierarchical tree) for Yeast data, and available information in [51] and [100] for Herpes and Fibroblast data respectively. For CLICK, the number of clusters are automatically determined by the algorithm itself.

3.4 Biological Interpretation

According to gene bank records, the Herpes genes have been classified into five groups, namely, Homologs of cellular regulatory or signal transduction genes, Virus gene regulation, DNA replication, DNA repair and nucleotide metabolism, and Virion formation and structure [51]. In this classification one gene can belong to only one category. In the case of Cell Cycle, Yeast Complex, and All Yeast the MIPS (Munich Information for Protein Sequences) [38] categorization is available for all the three Yeast data sets that allows a gene to belong to more than one category.

A biological score, that is different from the similarity/distance measure and defined in Eq. 3.5, is used to evaluate the final gene ordering. Each gene that has undergone MIPS categorization can belong to one or more categories, while there are many unclassified genes also (no category). A vector $V(g) = (v_1, v_2, \cdots, v_j)$ is used to represent the category status of each gene g, where j is the number of categories. The value of v_j is 1 if gene g is in the jth category; otherwise is zero. Based on the information about categorization, the score of a gene order for multiple class genes is defined as [115]

$$S(n) = \sum_{i=1}^{N-1} G\left(g_i, g_{i+1}\right), \tag{3.5}$$

where N is the number of genes, g_i and g_{i+1} are the adjacent genes and $G\left(g_i, g_{i+1}\right)$ is defined as

$$G(g_i, g_{i+1}) = \sum_{k=1}^{j} V(g_i)_k V(g_{i+1})_k, \tag{3.6}$$

where $V(g_i)_k$ represents the k^{th} entry of vector $V(g_i)$. For example consider the gene order

g_1, g_2, g_3

with 15 categories represented by vectors

$V(g_1) = (1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$

$V(g_2) = (1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$ and

$V(g_3) = (0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0)$

Then $G(g_1, g_2) = 3$ and $G(g_2, g_3) = 1$, and

$S(n) = G(g_1, g_2) + G(g_2, g_3) = 3 + 1 = 4$.

Note that, $S(n)$ can be used as scoring function for single class genes also. Using scoring function $S(n)$, a gene ordering would have a higher score when more genes within the same group are aligned next to each other. So higher values of $S(n)$ are better and can be used to evaluate the goodness of a particular gene order.

3.5 Experimental Results

Performance of the proposed FRAG_GALK for gene ordering is compared mainly with Concorde_LP [5] and B-Joseph et al.'s [14] method. Experiments of gene ordering are conducted in Matlab 7 on Sun Fire V 890 (1.2 GHz and 8 GB RAM). The codes for B-Joseph et al.'s [14] leaf ordering in hierarchical clustering solution are downloaded from [118]. SOM and K-means are available in Expander [100] and used with 16, 16, and 18 clusters for clustering Cell Cycle, Yeast Complex, and All Yeast data sets respectively as genes in these datasets are classified according to MIPS [38] categorization into 16, 16, and 18 groups. For Fibroblast data SOM is used with 6 clusters as 6 gene clusters are identified in [100]. Herpes data is clustered using 5 nodes as there are 5 gene groups [51]. Finally FRAG_GALK and Concorde_LP are applied separately on the gene clusters obtained by SOM, and B-Joseph et al.'s method is applied on the average linkage based hierarchical clustering solution for each dataset.

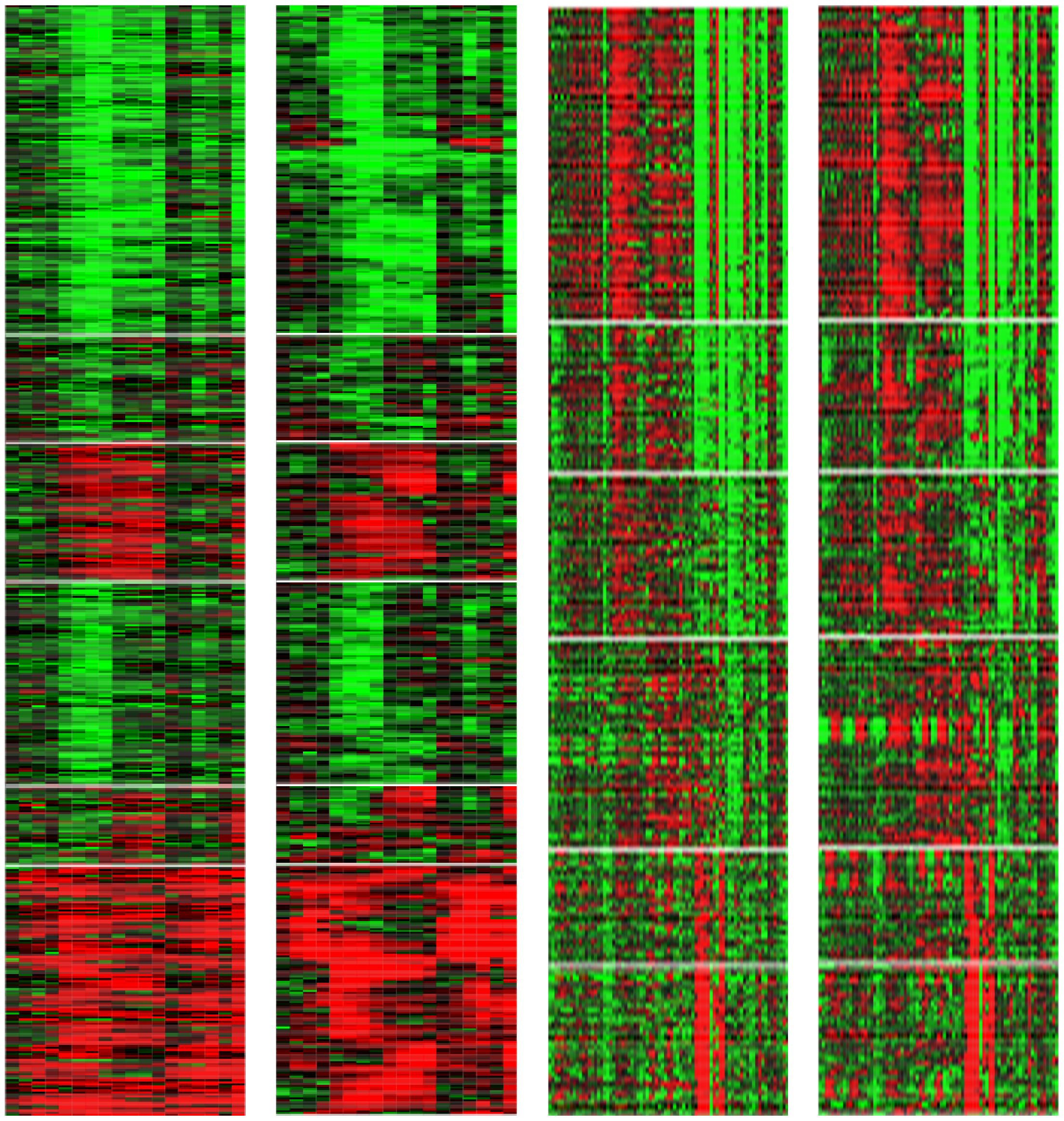


Figure 3.1: Comparing SOM with 'SOM+FRAG_GALK' for Fibroblast data (Fig. a and Fig. b respectively) and Yeast Complex data (Fig. c and Fig. d respectively). The expression profiles are represented as lines of colored boxes using Expander [100], each of which corresponds to a single experiment. Some grouped genes obtained by FRAG_GALK (Fig. b and Fig. d) have similar expression patterns and are coexpressed in each group.

3.5.1 Relevance of Gene Ordering in Partitive Clustering

To show the utility of the hybrid method in identifying different subclusters within big clusters and grouping the functionally correlated genes within clusters, here for illustration, the visual displays are presented for Fibroblast (Fig. 3.1-a and Fig. 3.1-b), Yeast Complex (Fig. 3.1-c and Fig. 3.1-d) and and Herpes data (Fig. 3.2). As stated previously, Fibroblast genes are first clustered in 6 clusters using SOM. Visual display of these 6 clusters is shown in Fig. 3.1-a. Observing this visual pattern no subcluster can be identified in each cluster. After applying FRAG_GALK on each cluster, closely related genes with similar expressions are aligned next to each other as shown in Fig. 3.1-b. Gene ordering here suggests that 2 or more subclusters exists at least in Clusters 1, 4 and 6, and it will be useful to increase the number of nodes of SOM to at least 9 for Fibroblast data. Note that, Iyer et. al. [34] identified 10 clusters of genes for this data using average linkage.

Yeast Complex data is first clustered in 16 groups using SOM. Visual display of first 6 clusters/groups is shown in Fig. 3.1-c. When the genes are ordered in each cluster using FRAG_GALK, 4, 4, 5, and 2 distinct subclusters are identified using visual display in clusters 2, 3, 4, and 5 respectively. Genes names along with their functional categories (indexes) for each subcluster within first 6 clusters are shown in Table 3.2 for the purpose of illustration. Names of the functional categories corresponding to their indexes are shown in Table 3.3. These subclusters of highly coregulated genes cannot be identified if SOM is used alone. For example, all the 9 genes in the 3rd subcluster of cluster 4 (YBR010W, YNL031C, YBL003C, YDR225W, YDR224C, YNL030W, YBR009C, YBL002W and YPL256C) are involved in Cell Cycle and DNA processing, Transcription, and Protein with Binding Function or Cofactor Requirement. While using SOM these 9 genes are distributed in the cluster 4, using CLICK these 9 genes are not assigned to any cluster, and are left as singletons along with 263 other singleton genes, which is undesirable as they belong to the same biological categories. After ordering genes using FRAG_GALK in cluster 4 of SOM and singleton genes of CLICK they (the 9 genes) are tightly grouped and identified easily using visual display.

Using CLICK, Herpes data is clustered in 1 group (Fig. 3.2-a) of 101 genes and 5 singleton genes, and no subcluster can be identified within it by visual display. When the genes are ordered in the cluster using FRAG_GALK, some subclusters are identified using visual display (Fig. 3.2-b), but the display is not conclusive in identifying exact number of sub-clusters. Using K-means, Herpes data is clustered with k=5 (5 clusters). Visual display of the first four clusters are shown in Fig.

Table 3.2: Gene subclusters found by SOM+FRAG_GALK and their functional category indexes in first 6 clusters identified using SOM for Yeast Complex data

Clu-ster	Sub-cluster	Genes	Function-al index
1	Nil	100 genes	5
2	1	YGR162W, YBR079C, YMR309C, YOR361C, YNL062C, YHR021C, YBL092W, YBR143C, YDL136W, YDL191W, YOR167C, YJL191W	
	2	YOR224C, YJR063W, YNL113W, YBR154C, YGL120C, YNL248C, YJL148W, YOR340C, YPR110C	4 and 7
	3	YMR146C, YOL139C, YOR276W, YKL156W, YOR182C, YDR447C, YKR094C, YIL148W, YDR500C, YMR121C, YLR264W, YJL138C, YKR059W, YKL081W, YLR249W	5
	4	YGR159C, YOR207C, YMR239C, YHR089C, YOR310C, YLR197W, YLL008W	4
3	1	YMR260C, YDR429C, YPL237W, YLR406C, YJR007W, YER025W, YPR041W, YDR172W, YDR211W	5
	2	YDR212W, YIL142W, YPL210C, YKL057C, YPL243W	6 and 7
	3	YLR060W, YOR260W, YDL040C, YKR026C, YLR291C, YBR142W, YBL087C, YHL001W, YDR450W, YHL033C, YBR191W, YBR189W, YBR048W, YBR118W	5
	4	YBR142W, YHR062C, YHR065C, YNR003C, YMR043W, YIL021W, YOR210W, YDR194C, YHR069C	4
4	1	YLR093C, YNL121C, YLR170C, YML112W, YBR160W, YBR171W, YLR378C, YML019W, YPL234C, YOR039W	6
	2	YKR068C, YLL050C, YGL200C, YML012W, YPL218W, YKL080W, YDR086C, YNL153C, YKL122C, YLR292C, YGL112C, YLR268W, YLR447C	6 and 9
	3	YBR010W, YNL031C, YBL003C, YDR225W, YDR224C, YNL030W, YBR009C, YBL002W, YPL256C,	3, 4, and 7
	4	YJL025W, YPR101W, YMR061W, YGR195W, YOR244W, YLR105C, YDL043C, YPR056W, YPR057W	4
	5	YGL100W, YNL261W, YKL144C, YNL151C, YJL008C, YER148W	7
5	1	YML051W, YNL103W, YCR035C, YJL006C, YOL135C, YDL165W, YOL102C, YMR270C, YOR085W, YJL002C, YKL211C	1 or 4 or (1 and 4)
	2	YBR087W, YJR043C, YJL173C, YJR076C, YMR080C, YPR029C, YKL045W, YLR103C, YNL262W, YKL113C, YLR212C, YER070W, YIL066C, YDR097C, YOL090W, YDL102W, YOR250C	3 or (3 and 7) or (1, 3 and 7)
6	Nil	49 genes	3

3.2-c. After ordering the genes in each cluster using FRAG_GALK, 3 and 3 distinct patterns are observed in cluster 1 and 2 respectively (Fig. 3.2-d). From Fig. 3.2-e it is found that B-Joseph's method is not very efficient (genes with all red expressions

Table 3.3: Indexes and corresponding functional category

Functional index	Functional Category
1	Metabolism
2	Energy
3	Cell Cycle and DNA Processing
4	Transcription
5	Protein Synthesis
6	Protein Fate (folding, modification, destination)
7	Protein with Binding Function or Cofactor Requirement
8	Protein Activity Regulation
9	Cellular Transport, Transport Facilitation and Transport Routes
10	Cellular Communication/Signal Transduction Mechanism
11	Cell Rescue, Defense and Virulence
12	Interaction with Cellular Environment
13	Cell Fate
14	Development (Systemic)
15	Biogenesis of Cellular component
16	Cell Type Differentiation

are scattered through out the visual display) in identifying clusters as partitive clustering algorithms. On the other hand, hybridizing FRAG_GALK with K-means (Fig. 3.2-c) or SOM (Fig. 3.1-b and 3.1-d) will provide a robust clustering method as well as relations between genes within clusters. With all these ordered and clustered genes one can easily zoom in a useful small subset of genes in a cluster which cannot be done alone with partitive clustering methods. In a similar way, subclusters within big clusters are identified by Concorde_LP for all the data sets.

3.5.2 Comparative Performance of the Algorithms

The ultimate goal of an ordering algorithm is to order the genes in a way that is biologically meaningful. In this regard, Table 3.4 compares the performance of our proposed two hybrid approaches using FRAG_GALK and Concorde_LP with B-Joseph's [14] leaf ordering in hierarchical clustering solution in terms of the $F_1(n)$ value (Eq. 3.4), S value (Eq. 3.5), and computation time. For Fibroblast data, no biological score can be provided as genes in the same biological group for this data are rare. From the biological scores (Table 3.4), it is evident that FRAG_GALK provides biologically comparable gene order with respect to Concorde_LP and some-

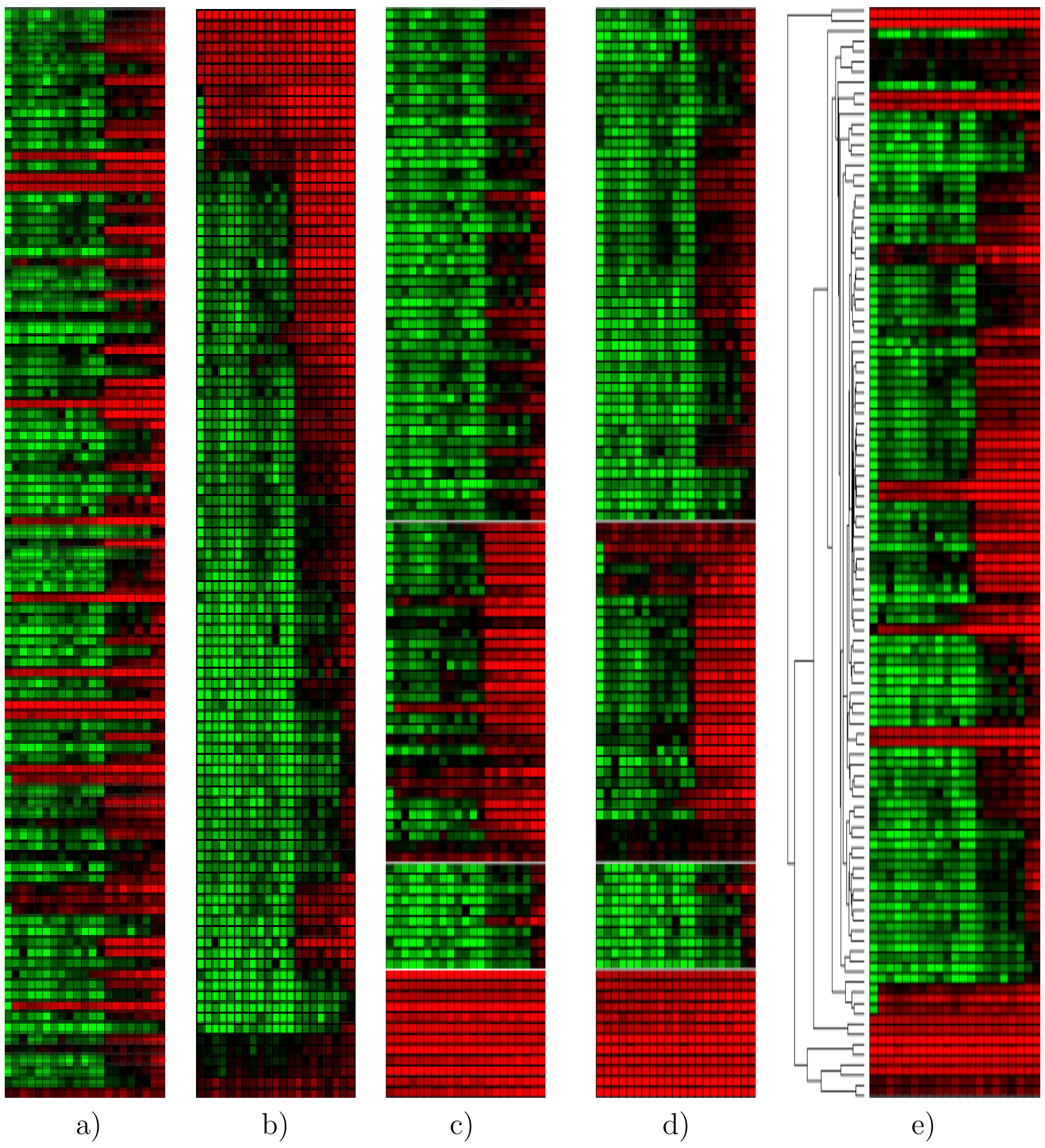


Figure 3.2: Comparing CLICK (Fig. a), 'CLICK + FRAG_GALK' (Fig. b), K-means (Fig. c), 'K-means+FRAG_GALK' (Fig. d), and B-Joseph's method (Fig. e) for Herpes data.

times superior gene order than 'leaf ordering in hierarchical clustering solution' by [14], for all datasets in least computational time. For example, FRAG_GALK took 125 seconds to order All Yeast data (6221 genes) as compared to Concorde_LP and B-Joseph et. al's method which took 2272 and 1989 seconds respectively.

Table 3.4: Summation of gene expression distances ($F_1(n)$), biological Score (S), and computation time of ordering in seconds (within parenthesis) for different hybrid algorithms using SOM

	Data Sets			
Algorithm	Cell cycle	Yeast Complex	All Yeast	Herpes
SOM	442.94 354	547.16 792	3446.60 1730	32.55 23
SOM +FRAG_GALK	301.72 386 (0.7)	330.54 1011 (1.13)	1919.15 2356 (125)	15.52 32 (0.09)
SOM +Concorde_LP	301.72 386 (3.41)	330.54 1011 (15.26)	1919.15 2356 (2272)	15.52 32 (0.44)
B-Joseph	300.51 381 (1.8)	330.17 1024 (3.34)	1920.82 2350 (1989)	15.50 31 (0.10)

3.6 Conclusion

A new method called integrating gene ordering with partitive clustering is proposed in this chapter. Its utility in finding useful subgroups of genes within cluster, grouping functionally correlated genes within clusters, maximization of biological gene ordering using MIPS categorization, and minimization of computation time, are demonstrated for Cell Cycle, Yeast Complex, All Yeast, Fibroblast and Herpes data sets. The hybrid approach not only determines the unique clusters, but also preserves the biologically meaningful relationships among the genes within clusters.

In FRAG_GALK, parallel searching (with large population in genetic algorithm) for optimal gene order in gene clusters (closely related genes) is performed. While this results in reduced searching time for FRAG_GALK as compared to Concorde_LP and B-Joseph's method, in terms of biological score, FRAG_GALK is comparable with Concorde_LP and B-Joseph's method and sometimes superior to B-Joseph's method. It is evident from the experimental results that the combination of partitive clustering and FRAG_GALK is a promising tool for microarray gene expression analysis.

Chapter 4

New Distance Measure for Microarray Gene Expressions using Linear Dynamic Range of Photo Multiplier Tube

4.1 Introduction

Two new operators of genetic algorithms for gene ordering and a hybrid method incorporating gene ordering in partitive clustering solution are described in the previous two chapters, respectively. The present chapter deals with the tasks of developing a distance measure for genes by normalizing their gene expression values with linear dynamic range of photo multiplier tube, a new gene ordering algorithm with less computational complexity then existing ordering algorithms, and evaluating them in gene ordering framework and the hybrid method developed in the previous chapter. While, pure gene ordering is an existing method [18, 91, 115], gene ordering in partitive clustering is proposed in [90] and described in Chapter 3.

The widely used measures for finding the global similarity (where all the gene expression values present in the gene are taken into consideration) between genes are the Pearson correlation and the Euclidean distance [18, 31, 103]. The Pearson correlation is over-sensitive to large three fold changes (peaks) in gene expression profiles, and therefore lead to false interpretation of similarity between genes in certain cases. Moreover, in computing the similarity, all the above mentioned measures do not assign appropriate weights to gene expressions obtained from different types of experiments, where the expressions differ by orders of magnitude from one type to another. Consequently, gene expression values in lower dynamic range do get dominated by those with higher dynamic range. A new similarity measure between genes, called "*Maxrange* distance" is defined in this chapter, where local (for a particular type of experiment) similarities between two genes are first normalized with a factor dependent on the dynamic range of gene expression values of that experiment (type) and then summed to find a global distance. Moreover, Manhattan/Euclidean distance is used in the distance measure to avoid the sensitiveness to large three fold changes in gene expression profiles (discussed in Section 4.3.3). Using any one of these distance measures between gene expressions, gene clustering and ordering can be performed.

Gene ordering is primarily necessary for identifying groups of highly co-regulated genes (discussed in detail in Section 1.6.4). Our proposed methods using FRAG_GA-LK, in Chapter 2 and 3 [91], and existing algorithm Concorde_LP [5] for finding optimal gene order, spend most of the time in repetitive searching for the lowest value (optimal value) of the sum of global similarities within gene groups of the same biological category, and result in the same biological score for all possible permutations of genes within the same group. Although this repetitive searching is obvious and useful for unidirectional optimal gene ordering, to avoid this situation in 'gene ordering in partitive clustering' framework, a fast gene ordering algorithm called "Minimal Neighbor" (MN), using nearest neighbor tour construction heuristic and involving $O(n^2)$ time complexity for computationally effective gene ordering, is described in this chapter.

Superiority of the proposed *Maxrange* distance measure over the related measures is established by using them on three different ordering algorithms, four clustering algorithms, and one hybrid algorithm. Similarly, the comparability of the MN algorithm as compared to two existing algorithms is demonstrated for three different distance measures. First, MN is compared with related algorithms in 'unidirectional gene ordering' framework and then its utility is shown in 'gene ordering in partitive clustering' framework. While the *Maxrange* distance provides the best similarity between genes, MN is seen to provide comparable biological gene ordering with Concorde_LP [5] and FRAG_GALK.

The widely used distance measures, gene clustering algorithms and gene ordering methods are described in Section 3.2.1, 1.6.3, and 1.6.4, respectively. In this chapter, first, the state-of-the-art work in distance measures and gene ordering algorithms are mentioned in Section 4.2. In Section 4.3, preliminary concepts of microarray technology, description of the data sets, their experimental condition and data preprocessing methods are provided. The proposed measure for gene expression distance and the fast algorithm for gene ordering is also explained in this section. The biological interpretation of the output obtained by different ordering algorithms is performed with Eq. 3.5 (described in Section 3.4). Experimental results are reported in Section 4.4. Finally, some conclusions and discussions are presented. The methods, algorithms and some of the results presented in this chapter have been reported in [89, 92].

4.2 Existing Approaches

4.2.1 Distance Measures

Various methods for extracting meaningful information from gene expression data have already been proposed in the literature. The methods for finding gene expression similarity between genes using the Pearson correlation [30], Manhattan (or Minkowski) distance [60] and the Euclidean distance [18, 43, 103] are described in previous chapters. A method for approximating an ideal similarity between genes (using thier expression), by training a neural network, has been suggested in [98]. Spellman et al. [105] used uncentered Pearson correlation as a similarity measure between genes. Qian et al. [83] addressed the problem of identifying local similarities in gene expression data by means of a method that is based on the Smith-Waterman algorithm for (local) sequence alignment. All of the aforementioned indices are numerical measures in the sense that they depend on the concrete numbers in the expression profiles. As opposed to this, Wen et al. (1998) suggest a shape-based similarity measure that compares two profiles on the basis of qualitative changes of expression values. Thus, two sequences are considered as similar if they increase and decrease more or less simultaneously. Filkov et al. [37] proposed a edge detection based similarity measure for periodic datasets with small sequences. This method looks for local regions in pairs of expression profiles where major changes in expression occur (edges). The profiles are regarded as similar if they do have similar edges. Balasubramaniyan et al. [11] used a local shape based similarity measure motivated by BLAST algorithm to cluster time series of gene expression data. By converting a time series into a sequence of events such as increase or decrease, the shape based methods tend to oversimplify the original data. This makes them robust toward noise and outliers, but loss of lot of information contained in the original time series makes the measure unsuitable for gene ordering. So shape based similarity measures are not considered in this investigation for gene ordering. One can thus construct a matrix of inter-gene similarties/distances using any similarity/distance measure, which serves as a knowledge-base for finding gene cluster or gene order or both.

4.2.2 Gene Clustering Methods

Cluster analysis and displays of gene expression patterns are considered to be useful tools to detect genes that are co-expressed or implicated in similar cellular functions [18, 31, 109]. Clustering methods can be broadly divided into hierarchical and nonhierarchical clustering approaches and described in Section 1.6.3. In brief, hierarchical clustering approaches (single, complete and average linkage) [2,14,18,31,56] group gene expressions into trees of clusters. They start with singleton sets and merge all genes until all nodes belong to only one set. Nonhierarchical clustering approaches, such as k-means [49], self-organizing map (SOM) [109], Bayesian clustering [15], CAST [16], and CLICK [101], separate genes into groups according to the degree of distance among genes. The relationships among the genes in a particular cluster generated by nonhierarchical clustering methods are lost.

4.2.3 Gene Ordering Methods

Gene ordering methods in travelling salesman problem framework and hierarchical clustering framework are discussed in Section 2.2 and 1.6.4, respectively. A method for ordering genes for a partitive clustering solution is reported in Chapter 3. The utilities and applications of these methods are also shown in the respective sections and chapters. Here, we will discuss the issues related with the need for developing a new gene ordering algorithm for partitive clustering solution with lower time complexity than existing algorithms. The details of the new ordering algorithm are discussed in Section 4.3.4.

Ray et al. [91] and Tsai et al. [115] applied genetic algorithm for solving optimal gene ordering problem without clustering. Related works are also available in [26, 27]. These algorithms are very useful in identifying groups of highly correlated eight to ten genes for inferring regulatory networks, and manual identification of clusters from the heat map of the ordered genes. However, application of these algorithms in partitive clustering solution results in repetitive searching for lower sum of gene distances. Consequently, repetitive searching leads to significant increase in computational time with respect to all other hierarchical clustering based approaches. For example FRAG_GALK [91], HeSGA [115], and take about 245, 612, and 5042 seconds, respectively, for finding gene/city order for 3038 genes/cities on a pentium IV 1.2 MHz pc with C++, and the time is exponentially growing with higher number of genes/cities, whereas hierarchical clustering and SOM based gene approaches take on average 90-200 seconds for the same problem.

4.3 Materials and Methods

4.3.1 Preliminary Concepts for Measuring Gene Expression with Fluorescence Scanner

In Section 1.4, it is mentioned that, in cDNA (clone DNA) microarray-based investigations, RNA from experimental samples (taken at selected times during the process) is labelled during reverse transcription with the red-fluorescent dye Cy5 and is mixed with a reference sample labelled in parallel with the green-fluorescent dye Cy3 [31]. After hybridization and appropriate washing steps, separate images/spots are acquired for each fluor, and fluorescence intensity ratios are obtained for all target elements. It is also necessary to mention again that, if R (red) and G (green) are the spot-specific, quantitated, fluorescent intensities of the target and reference expression signals respectively, relative gene expression is defined as the log ratio $M = log_2 \frac{R}{G}$.

Fluorescence is currently the predominant method for microarray signal detection [33]. A critical component of a fluorescence scanner is the photomultiplier tube (PMT), in which fluorescent photons produce electrons that are amplified by the

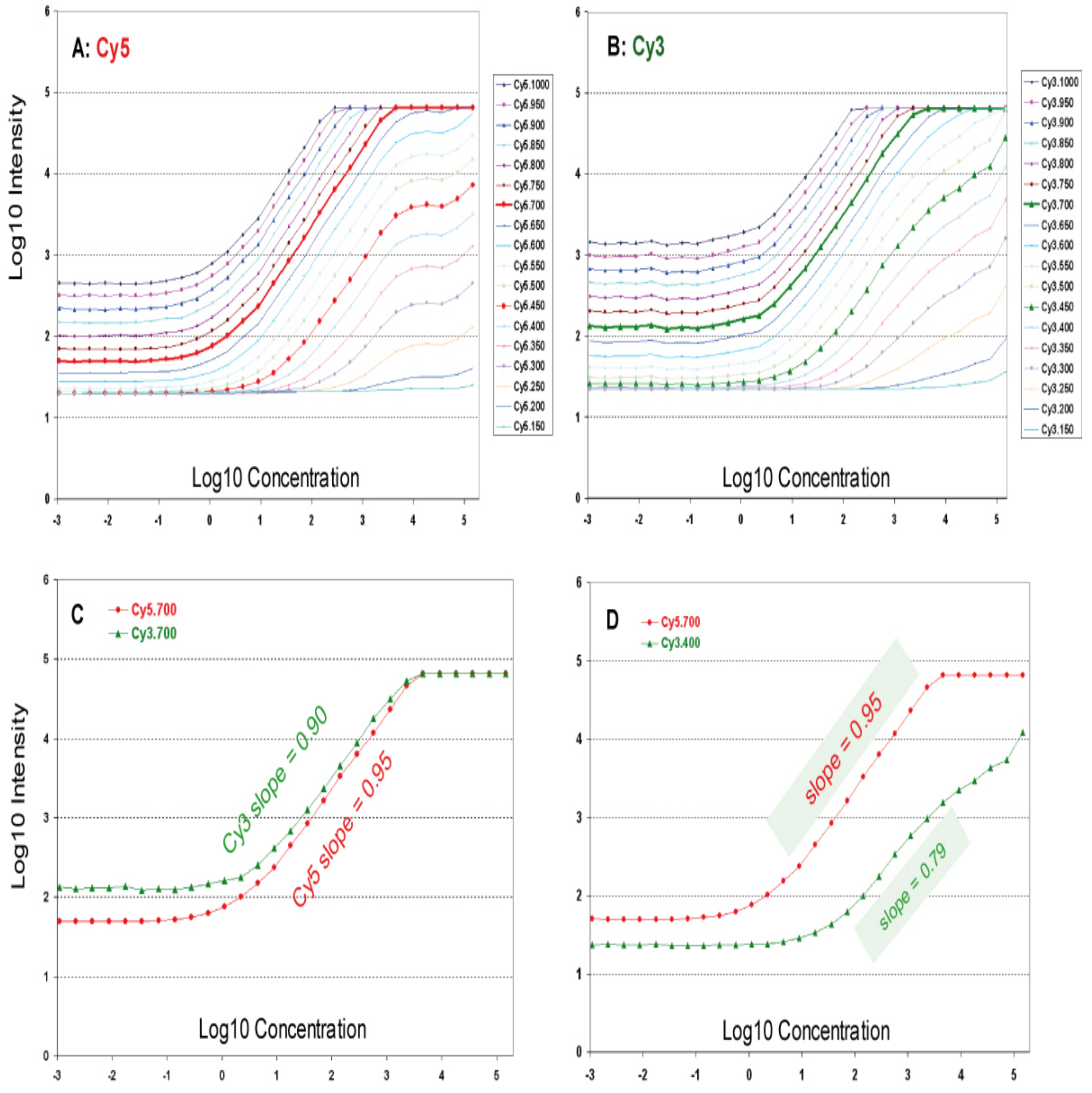


Figure 4.1: Calibration curves of photomultiplier tube under different PMT gains. X-axis: log10 concentration, Y-axis: log10 fluorescence intensity. A: Cy5 dye; B: Cy3 dye. Representative calibration curves are presented in C (Cy5 and Cy3 channels are scanned under the same PMT gain of 700 V) and D (the Cy5 and Cy3 channels are scanned at 700 V and 400 V, respectively). (Figure is taken from [33]).

PMT gain. For many microarray scanners, the calibration curve (i.e., the curve showing the relationship between dye concentration and fluorescence intensity) depends on the PMT gain setting [33]. This PMT gain is also varied for different types of experiments of different biological origin. DNA microarray measurements normally assume a linear relationship between detected fluorescent signal and the concentration of the fluorescent dye that is incorporated into the cDNA (clone DNA) or RNA molecules synthesized from the test sample. Each PMT has its own linear dynamic range within which signal intensity increases linearly with the increase of fluorescent dye concentration [33] (see Fig. 4.1). This linear dynamic range also fixes the dynamic range of the recorded microarray data (log ratio values) [33] within which the data values are most reliable and used as the normalization factor in the proposed distance measure to remove variations of biological origin. For example, in Cell Cycle related experiments, for dye Cy5 PMT gain at 960 volts fixes the intensity range from x1 to x2, and for dye Cy3 PMT gain at 760 volts fixes the intensity range from y1 to y2. So the linear dynamic range of PMT fixes the linear dynamic range of the data from $log_2 \frac{x1}{y1}$ to $log_2 \frac{x2}{y2}$. Note that, this dynamic range is not obtained from the datasets and hence is not sensitive to outliers. Normalization with standard deviation (used in Pearson correlation) for all the expression values for a particular gene also removes variations due to biological origin like Cell Cycle, sporulation etc. However, due to the wide concentration range for genes expressed in a biological sample, the detected fluorescence intensity does not necessarily remain in the linear range for all genes tiled on a microarray (on average 5% of the gene expression values in each dataset used in this investigation are outside the linear range even after normalization). Nonlinearity between fluorescence intensity and dye concentration can occur due to chemical saturation, dye quenching, signal bleaching, optical saturation, and instrument limitations. To remove impact of such nonlinear bias and variations on microarray data, many normalization methods have been proposed (see Section 1.6.2) in the literature.

The proposed dynamic range based normalization (described in Section 4.3.3 in terms of similarity measure) belongs to between-slide or multiple-slide normalization [127]. The two other normalization factors in this category, which aim to allow experiment to experiment comparisons when different types of experiment have substantially different spreads in log ratios, are median absolute deviation and variance regularization (see Section 1.6.2). These two normalization methods, viz., MAD and variance regularization, were implemented for the purpose of comparison. However, the results obtained were not very encouraging.

Table 4.1: Summary for different microarray data sets

Dataset	Genes	Category	Experiments performed				Total
Cell Cycle	652	MIPS 16	Cell Cycle (-1.2 to 1.2) 93	sporulation (-3.0 to 3.0) 9	shock (-1.5 to 1.5) 56	diauxic shift (-2.0 to 2.0) 26	184
Yeast Complex	979	MIPS 16	Cell Cycle (-1.2 to 1.2) 18+14+15	sporulation (-3.0 to 3.0) 7+4	shock (-1.5 to 1.5) 6+4+4	diauxic shift (-2.0 to 2.0) 7	79
All Yeast	6221	MIPS 18	Cell Cycle (-1.2 to 1.2) 60	sporulation (-3.0 to 3.0) 13	diauxic shift (-2.0 to 2.0) 7		80
Fibr-oblast	517	GO 1347	Serum response (-3.0 to 3.0) 12	cycloheximide (-3.0 to 3.0) 6			18
Herpes	106	Gene-Bank 5	No KSHV (-13.0 to 13.0) 1	-TPA (-13.0 to 13.0) 7	TPA (-13.0 to 13.0) 13		21

4.3.2 Description of Data Sets

The datasets used in this investigation are also used in the previous investigation and described in Section 3.3.1. Here, we will mention the details of linear dynamic range of photomultiplier tube, used for each type of experiment, and the issues related with data format and downloading. Table 4.1 shows the name of the data sets, number of genes in each dataset, number of gene categories, name of experiment types and number of experiments performed under each type, and finally the total number of experiments performed for a particular dataset. The dynamic range of expression values of each experiment is shown within parenthesis. The dynamic range of available data represents log ratios of -1.2 to 1.2 for the cell-cycle experiments, -3.0 to 3.0 for sporulation, -1.5 to 1.5 for the shock experiments, -2.0 to 2.0 for the diauxic shift, -3.0 to 3.0 for Fibroblast data, and -13.0 to 13.0 for Herpes data. Herpes data is generated using radioactive probes instead of fluorescent probes and hence higher linear dynamic range is observed than the other data sets. The information for Herpes data is obtained from Dr. Paul Kellam upon request.

The first three data sets of Saccharomyces cerevisiae consists of about 652, 979 and 6221 genes, and 184, 79 and 80 microarray experiments respectively. The genes in the three data sets are classified according to MIPS [38] categorization into 16, 16, and 18 groups respectively. For the Cell Cycle data, first we have downloaded 652 Cell Cycle regulated gene names from the MIPS website. These gene names are then uploaded in Stanford Microarray Database [32] and corresponding gene expression values are downloaded with default parameters by selecting all the cell-cycle, sporulation, heat shock and diauxic shift experiments. The default parameters include intensity-dependent ratio bias correction with Lowess [25] and $log_2\frac{R}{G}$ format. The Yeast Complex and All Yeast datasets are available $log_2\frac{R}{G}$ format and assumed to be corrected from intensity-dependent ratio bias as separate R and G values are not available. The fibroblast dataset consists of 517 genes and 18 time points related to the response of human fibroblasts to serum. According to Gene Omnibus (GO) annotation, 517 fibroblast genes are distributed in 1347 categories. This dataset is available in two formats ($\frac{R}{G}$ and $log_2\frac{R}{G}$). We have downloaded the data in second format. Similarly for herpes data two formats are available and data in $log_2\frac{R}{G}$ format has been downloaded with 106 genes and 21 experiments. Herpes virus genes are broadly assigned to five functional groups and available in [51] and genes that could not be assigned to any of these five groups are designated unknown.

When the data sets are downloaded from the website it is found that about 5% of the gene expression values in each dataset is beyond the above specified dynamic range, so for the Cell Cycle dataset any value beyond the range -1.2 to 1.2 is truncated to the minimum/maximum reliable value. Similarly for other datasets the gene expression values are truncated accordingly. Note that, we have also treated the values outside the linear dynamic range as missing values, and predicted them with LSimpute [8] (statistical package to predict missing values). In this case also the predicted values are very near to one of the boundaries of linear dynamic range and supported our method of truncation. This also provides superior results (in terms of biological score discussed in Section 3.4) than without truncation for all the data sets, algorithms, and similarity measures.

4.3.3 New Distance Measure

A natural basis for organizing gene expression data is to group together genes with similar patterns of expression. The first step to this end is to adopt a mathematical description of distance. A number of measures of distance in the behavior of two genes can be used, such as the Manhattan distance, Euclidean distance, Pearson Correlation. The Pearson correlation is over-sensitive to large three fold changes (peaks) in gene expression profiles due to multiplication of expression vectors in dot product style, and therefore leads to false interpretation of distance between genes in certain cases. Moreover, it is observed that often microarray data consist of different sets of expression values corresponding to different experiment types. For example, for Yeast Complex data, four sets of expression values (each set containing multiple time series gene expression) can be recorded for cell-division cycle, sporulation, shock, and diauxic shift respectively. Existing distance measures usually take the same normalization factor (like standard deviation for Pearson correlation) for a gene. This normalization factor

- is independent of the type of experiment
- performs global normalization to all the expression values for a particular gene; thus loosing useful local information and
- varies from gene to gene

But, a closer look at the gene expression data reveals that the dynamic range of expression values differs with the type of experiment, and remains the same for all the genes in the dataset. So, using the same normalization factor is undesirable for all types of experiments, where expression values differ by orders of magnitude from one kind of experiment to another. Consequently, it may be appropriate and better if the normalization is performed

- separately for the different types of experiment with different normalizing factors; thereby preserving the local information
- keeping the same set of normalization factors for all the genes in the dataset.

Such an attempt is made in this chapter where two new distance measures are developed using Manhattan distance and Euclidean distance respectively (to avoid over sensitivity to three fold changes), in which the normalization is dependent on the type of experiment. This, in turn, results in equal weighting of distance values for different experiment types. The normalization factor is chosen as the linear dynamic range of data values obtained from photo multiplier tube, for a particular type of experiment.

Let
$X = x_1^{e_1}, \cdots, x_{i_1}^{e_1}, x_1^{e_2}, \cdots, x_{i_2}^{e_2}, \cdots, x_1^{e_m}, \cdots, x_{i_m}^{e_m}$ and
$Y = y_1^{e_1}, \cdots, y_{i_1}^{e_1}, y_1^{e_2}, \cdots, y_{i_2}^{e_2}, \cdots, y_1^{e_m}, \cdots, y_{i_m}^{e_m}$

be the expression levels of the two genes in terms of log-transformed microarray gene expression data obtained over a series of m different types of experiment $(e_1, e_2, \cdots e_m)$ consisting of $i_1 + i_2 + \cdots + i_m$ experiments in total. Using Manhattan distance the *Maxrange* distance between X and Y is defined as

$$\textit{Maxrange-}M_{X,Y} = \frac{1}{m}\sum_{r=1}^{m}\frac{1}{i_r} \times \frac{\sum_{j=1}^{i_r}|x_j^{e_r} - y_j^{e_r}|}{Max_{e_r} - Min_{e_r}} \tag{4.1}$$

where, Max_{e_r} and Min_{e_r} are the maximum and minimum $log_2(R/G)$ values obtained from the linear dynamic range of the photo multiplier tube (or radioactive probe) for an experiment of type e_r.

The following can be stated about the measure:

1. $0 \leq \textit{Maxrange-}M_{X,Y} \leq 1$
2. $\textit{Maxrange-}M_{X,Y} = 0$ if and only if $X = Y$
3. $\textit{Maxrange-}M_{X,Y} = \textit{Maxrange-}M_{Y,X}$ (symmetric).

Using the Euclidean distance the *Maxrange* distance between X and Y is defined as

$$\textit{Maxrange-}E_{X,Y} = \frac{1}{m}\sum_{r=1}^{m}\frac{1}{i_r} \times \frac{\sqrt{\sum_{j=1}^{i_r}(x_j^{e_r} - y_j^{e_r})^2}}{Max_{e_r} - Min_{e_r}} \tag{4.2}$$

Throughout the literature we have used *Maxrange-M* and *Maxrange-E* for representing *Maxrange* distance measure using Manhattan and Euclidean distance respectively.

Let three genes X, Y, and Z with four different types of experiments, have the gene expression values

$X = 0.02, \ -0.1, \ 2.9, \ \ 0.1, \ -0.1, \ 0.1, \ -0.15, \ 0.1$

$Y = \ 0.1, \ -0.05, \ 0.15, \ -0.2, \ -0.3, \ 0.64, \ 0.0, \ 0.3.$ and

$z = \ 0.13, \ -0.09, \ 0.1, \ -0.2, \ 1.2, \ 1.2, \ 1.7, \ 1.9$

Assume that the first two expression values for all the genes correspond to cell-cycle experiments with dynamic range between 1.2 to -1.2 *i.e.*, $(Max_{e_1} - Min_{e_1} = 2.4)$, the third and fourth values correspond to sporulation experiments with dynamic range between 3.0 to -3.0 *i.e.*, $(Max_{e_2} - Min_{e_2} = 6)$, the fifth and sixth values correspond to shock experiments with dynamic range between 1.5 to -1.5

i.e., $(Max_{e_3} - Min_{e_3} = 3)$ and the seventh and eighth values correspond to diauxic shift experiments with dynamic range between 2.0 to -2.0 *i.e.*, $(Max_{e_4} - Min_{e_4} = 4)$. So *Maxrange-M* distance and Pearson Correlation distance between gene X and Y is 0.11208 and 0.85202, respectively.

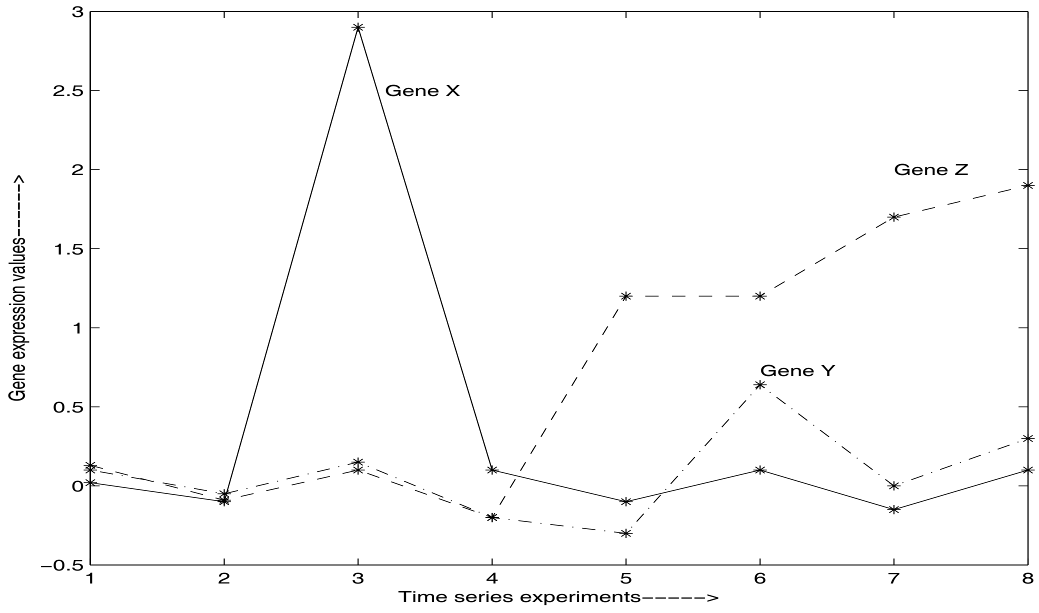


Figure 4.2: Expression profile for three genes. According to *Maxrange-M*, distance between genes X and Y is higher than Z and Y which is in opposition with Pearson correlation and Euclidean distance

To illustrate the difference between the *Maxrange-M* and Pearson correlation, consider Gene X and Gene Y in Fig. 4.2, which shows two profiles (of length 8), which are highly similar according to the *Maxrange-M* but almost dissimilar (uncorrelated) according to the Pearson correlation. This is mainly caused by the comparatively large value of the third fold change in Gene X. As opposed to this, in *Maxrange-M*, sensitivity to three fold change is avoided using Manhattan distance, and normalization with dynamic range of type of experiment correctly reflects the fact that both profiles have similar expressions for three types of experiments namely, cell-cycle, shock and diauxic shift, and differs in only one expression (among two expressions) for sporulation experiments. *Maxrange-E* distance also shows similar performance as *Maxrange*-M. The Euclidean distance between X and

Y is 2.8382, and Y and Z is 2.8317. But X differs with Y in only one expression value of high range experiment type ($Max_{e_r} - Min_{e_r} = 6$), whereas, Z differs with Y in three expression values of relatively small range experiment type. So in the case of Euclidean distances, experiment types with high range dominate the experiment types with small range ones. Pearson correlation also assigns higher distance between X and Y (0.85202) than Y and Z (0.67295), neglecting the fact that Y and Z differ significantly in 5th, 7th and 8th expression values in two types of experiments that have small ranges, while X and Y differ significantly in only one expression (3rd) of high range. As opposed to these, *Maxrange-M* distance between X and Y is 0.11208, which is less than the distance between Y and Z (0.19365). *Maxrange-E* distance between X and Y is also less than the distance between Y and Z.

4.3.4 New Ordering Algorithm

The existing algorithms for finding the optimal gene order spend most of the time in repetitive searching for the lower value of the sum of gene expression distances in gene groups (genes belonging to same category), and result in the same biological score for all possible permutations of genes within the the same group. Under this situation, to avoid repetitive searching, the nearest neighbor (NN) tour construction heuristic can be used to find a near optimal gene order in terms of gene expression distance. The NN tour has the advantage that it commits only a few severe mistakes in tour construction, while there are long segments connecting nodes with short edges. It has a disadvantage that, several genes which are not considered during the course of the algorithm are inserted at high costs in the end. To overcome this to some extent, we propose a new heuristic based Minimal Neighbor (MN) algorithm.

Let $1, 2, .., i, ...n$ represent the indices of n genes in the microarray dataset and the distance between gene i and $i + 1$ be denoted as $C_{i,i+1}$. Given this microarray dataset of n genes to be ordered and pairwise distance/similarity (of each gene with all other genes) kept in an $n \times n$ matrix (after calculating), the different steps of applying MN are explained below.

S1) Find the closest (most similar) pair of genes and merge them into a single array (string), so that there remains $n - 2$ genes.

S2) Consider only the two end genes of the new array and find two closest genes for each of them from the remaining genes. Out of these two selected genes, find the one closer to one of the end genes of the array, and then place it next to that. The other selected gene is not connected and kept with the remaining genes. The index of this gene is stored for use in the next step. (Note that, if both the selected genes are the same in this step then no gene index can be stored and in the next step we have to compute twice for selection of two genes, else, only one closest gene is needed to be computed.)

S3) Repeat S2 until all genes are aligned into a single array of size n.

Computational complexity of Step 1 is $O((n/2)^2)$ as the distance matrix is a symmetric one. This step can be performed during the calculation of $n \times n$ distance matrix also. For Steps 2-3 the worst case complexity is $O(2 * (n-2) * n)$. So the total complexity of the algorithm is $O(n^2)$.

Example: let us consider a microarray data set with five genes represented by 1, 2, 3, 4 and 5, and a 5×5 distance matrix computed using pairwise distance. Let the closest pair of genes be 2 and 4. So they are merged to form a string as

$$P1 = 2\ 4.$$

For these two end genes (2 and 4), the closest gene (from 1, 3 and 5) is determined using the 5×5 distance matrix. In this example, let the closest gene from 2 be 5 and from 4 be 1. Moreover let the gene 5 be closer to 2 than 1 from 4. Then 5 is selected and connected to 2 to form a string like

$$P2 = 5\ 2\ 4.$$

The next closer pairs are then searched for 5 and 4 out of the remaining genes 1 and 3, and let these be found as 3 5 and 4 1. Suppose 4 1 pair is closer than 3 5. Then 4 and 1 are connected to form the string

$$P3 = 5\ 2\ 4\ 1.$$

Finally from the two end genes 5 and 1, let the gene 3 be found to be closer to gene

1. Then the final string P4 is formed as

$$P4 = 5\ 2\ 4\ 1\ 3.$$

4.4 Experimental Results

Algorithms of gene ordering and clustering are implemented using mex files in Matlab 7 on Sun Fire V 890 (1.2 GHz and 8 GB RAM). The codes for single, average and complete linkage and Bar-Joseph et al.'s [14] method are downloaded from [118]. Performance of the proposed *Maxrange-M* and *Maxrange-E* distance are compared with Pearson correlation, Euclidean distance, and Manhattan distance. The MN algorithm for gene ordering is compared mainly with FRAG_GALK [91] and Concorde_LP [5]. Performance comparisons of MN with Bar-Joseph et al.'s gene ordering [14], and partitive clustering methods with "gene ordering with MN in partitive clustering solutions" are also provided. SOM and k-means are used with 16, 16, and 18 nodes (clusters) for clustering Cell Cycle, Yeast Complex, and All Yeast data sets, respectively, as genes in these datasets are classified according to MIPS [38] categorization into 16, 16, and 18 groups. For Fibroblast data SOM is used with 6 nodes as 6 gene clusters are identified in [101]. Herpes data is clustered using 5 nodes of SOM as there are 5 gene groups [51]. Finally MN is applied separately on the gene clusters obtained by partitive clustering methods.

4.4.1 Comparative Performance of Algorithms and Distance Measures

Table 4.2 shows, as an illustration, the summation of gene expression distances in terms of $F(n)$ (computed using Eq. 2.2 for Concorde_LP, MN and B-joseph) and $F_1(n)$ value (computed using Eq. 3.4 for SOM+MN) with *Maxrange-M*, Pearson Correlation, and Euclidean distance for all the datasets and four ordering algorithms. In this comparative study among the ordering algorithms, FRAG_GALK and Concorde_LP provide identical and the lowest sum of gene expression distances in terms of $F(n)$ (Eq. 2.2) value for all the distance measures and data sets, though they have the higher computational complexity ($O(n^5)$ and $O(2^n)$, respectively) than related algorithms. Gene ordering results provided by Concorde_LP are not shown in any table of this result section as they are identical with FRAG_GALK.

Table 4.2: Summation of gene expression distances computed in terms of $F(n)$ (Eq. 2.2 for FRAG_GALK, MN and B-joseph) and $F_1(n)$ (Eq. 3.4 for SOM+MN) value for different ordering algorithms and distance measures

		Data Sets				
Dist.	Algo.	Cell cycle	Yeast comp.	All Yeast	Fibro-blast	Herpes
Max -ran -ge -M	FRAG_GALK	69.00	80.87	463.85	26.21	2.73
	MN	71.50	84.31	481.69	27.87	2.89
	B-joseph	71.67	84.75	493.51	27.87	2.80
	SOM+MN	73.91	88.27	505.12	29.27	3.12
Pea -rs -on	FRAG_GALK	286.51	306.14	1773.15	71.82	11.69
	MN	298.25	327.92	1874.23	81.53	12.37
	B-joseph	300.51	330.17	1920.82	81.71	12.12
	SOM+MN	323.67	361.17	1970.16	99.96	14.34
Euc -lid -ean	FRAG_GALK	3913.90	3244.75	20302.16	851.86	419.48
	MN	4039.40	3386.11	21101.72	902.23	431.73
	B-joseph	4051.43	3388.90	21530.38	897.25	431.36
	SOM+MN	4197.93	3518.30	21525.84	943.27	475.23

Superior performance of FRAG_GALK w.r.t. all other related algorithms for finding optimal gene/city order, in terms of computational time, are shown in Table 2.4 and Section 2.5.2. MN and B-joseph's algorithm provide comparable results in terms of $F(n)$ value. Table 4.3 shows the comparative study in terms of $F(n)$

Table 4.3: Summation of gene expression distances computed in terms of $F(n)$ (Eq. 2.2 for average, complete and single linkage) and $F_1(n)$ (Eq. 3.4 for SOM) value for different clustering algorithms and distance measures

		Data Sets				
Dist.	Algo.	Cell cycle	Yeast comp.	All Yeast	Fibro-blast	Herpes
Maxra -nge-M	AL	76.02	91.31	524.05	32.17	3.420
	CL	76.20	91.28	523.64	31.57	3.522
	SL	97.28	120.62	787.56	49.59	4.486
	SOM	88.24	112.18	712.87	48.54	4.182
Pear -son	AL	332.54	373.80	2187.12	107.22	16.26
	CL	335.17	370.41	2192.12	103.09	14.62
	SL	481.91	584.74	4199.21	222.93	19.83
	SOM	445.34	547.16	3446.60	202.34	18.03
Eucli -dean	AL	4289.53	3633.66	22750.80	1025.54	496.71
	CL	4254.10	3612.30	22610.28	1026.80	496.43
	SL	5355.10	4828.70	33517.56	1537.40	698.71
	SOM	5073.64	4376.70	28624.69	1389.16	644.37

(computed using Eq. 2.2 for average linkage (AL), complete linkage (CL) and single linkage (SL)) and $F_1(n)$ value (computed using Eq. 3.4 for SOM) among four clustering algorithms. It may be noted that clustering and ordering algorithms cannot be compared as clustering algorithms do not optimize $F(n)$ (or $F_1(n)$) value; hence $F(n)$ (or $F_1(n)$) values for clustering algorithms are higher than those for the ordering algorithms (reported in Table 4.2), which is expected. In this case, single linkage (SL) and SOM are found to provide poor performance.

Table 4.4 compares the performance of our proposed approach with those of the other ordering methods in terms of the S value (Eq. 3.5). Three distance measures are considered, namely, *Maxrange-M*, Pearson and Euclidean. The biological scores corresponding to Manhattan Distance are found to be comparable to those for Pearson Correlation, and hence omitted here. The percentages of improvement over the lowest biological score (in terms of S value) in a particular data set are shown within parenthesis, and defined as:

$$PI_{i,j} = \frac{d_{i,j} - min_i(d_{i,j})}{min_i(d_{i,j})} \times 100 \tag{4.3}$$

where, $d_{i,j}$ indicates the biological score (S value) in ith row and jth column of the result matrix in the concerned tables (Tables 4.4, 4.5, and 4.6), and $min_i(d_{i,j})$ indicates the minimum biological score in column j for all i.

Table 4.4: Biological Score and Percentage of Improvement (PI) value (within parenthesis) for different gene ordering algorithms and distance measures

		Data Sets			
Algo. & complexity	Dist.	Cell cycle	Yeast comp.	All Yeast	Herpes
FRAG_GALK	*Maxrange-M*	420 (10.24)	1089 (11.69)	2383 (6.05)	44 (29.41)
	Pearson	400 (4.99)	1039 (6.56)	2350 (4.58)	34 (0.00)
$O(n^5)$	Euclidean	400 (4.99)	1051 (7.79)	2415 (7.48)	40 (17.65)
Min.	*Maxrange-M*	425 (11.55)	1075 (10.26)	2388 (6.28)	43 (26.47)
Neigh.	Pearson	403 (5.77)	1031 (5.74)	2349 (4.54)	37 (8.82)
$O(n^2)$	Euclidean	406 (6.56)	1010 (3.59)	2382 (6.01)	40 (17.65)
Bar-Joseph	*Maxrange-M*	423 (11.02)	1074 (10.15)	2371 (5.52)	43 (26.47)
et. al.	Pearson	381 (0.00)	1024 (5.03)	2350 (4.58)	38 (11.76)
$O(n^4)$	Euclidean	421 (10.50)	1013 (3.90)	2346 (4.41)	40 (17.65)
SOM+Minimal	*Maxrange-M*	409 (7.35)	1039 (6.56)	2335 (3.92)	42 (23.53)
Neighbor	Pearson	386 (1.31)	1002 (2.77)	2302 (2.45)	38 (11.76)
$O(n^2)$	Euclidean	381 (0.00)	975 (0.00)	2247 (0.00)	37 (8.82)

Similarly, the results for clustering algorithms in terms of S value, and percentage of improvement over the lowest biological score (using Eq. 4.3) are shown in Table 4.5. Table 4.6 shows the performance of our proposed approach 'SOM+MN' with respect to SOM alone for the same set of parameters. These PI values in Table 4.4 to 4.6 are used in the next section for conducting t-tests.

Table 4.5: Biological Score and Percentage of Improvement (PI) value (within parenthesis) for different clustering algorithms and distance measures

		Data Sets			
Algo. & complexity	Dist.	Cell cycle	Yeast comp.	All Yeast	Herpes
Average	*Maxrange-M*	415 (15.60)	1040 (22.50)	2341 (21.30)	39 (11.43)
Linkage	Pearson	385 (7.24)	987 (16.25)	2292 (18.76)	38 (8.57)
$O(n^2 \log n)$	Euclidean	403 (12.26)	1011 (19.08)	2431 (25.96)	39 (11.43)
Complete	*Maxrange-M*	407 (13.37)	1043 (22.85)	2305 (19.43)	38 (8.57)
Linkage	Pearson	393 (9.47)	955 (12.49)	2301 (19.22)	36 (2.86)
$O(n^2 \log n)$	Euclidean	403 (12.26)	999 (17.67)	2269 (17.56)	37 (5.71)
Single	*Maxrange-M*	382 (6.41)	903 (6.36)	1970 (2.07)	41 (17.14)
Linkage	Pearson	359 (0.00)	902 (6.24)	1973 (2.23)	39 (11.43)
$O(n^2)$	Euclidean	361 (0.56)	849 (0.00)	1930 (0.00)	36 (2.86)
SOM	*Maxrange-M*	389 (8.36)	973 (14.61)	2100 (8.81)	40 (14.29)
	Pearson	369 (2.79)	944 (11.19)	2073 (7.41)	36 (2.86)
$O(n^2)$	Euclidean	361 (0.56)	902 (6.24)	2041 (5.75)	35 (0.00)

Table 4.6: Biological Score and Percentage of Improvement (PI) value (within parenthesis) for 'SOM+MN' and SOM

		Data Sets			
Algo. & complexity	Dist.	Cell cycle	Yeast complexes	All Yeast	Herpes
SOM+MN	*Maxra -nge-M*	409 (13.30)	1039 (15.19)	2335 (14.40)	42 (20.00)
$O(n^2)$	Pear -son	386 (6.93)	1002 (11.09)	2302 (12.79)	38 (8.57)
	Eucli -dean	381 (5.54)	975 (8.09)	2247 (10.09)	37 (5.71)
SOM	*Maxra -nge-M*	389 (7.76)	973 (7.87)	2100 (2.89)	40 (14.29)
	Pear- -son	369 (2.22)	944 (4.66)	2073 (1.57)	36 (2.86)
$O(n^2)$	Eucli -dean	361 (0.00)	902 (0.00)	2041 (0.00)	35 (0.00)

For Fibroblast data, no biological score can be provided as genes in the same biological group for this data are rare. In order to investigate the effect of normalization using median absolute deviation (MAD) or variance regularization factor, a series of experiments with and without normalization is conducted. For each of the distance measure and any algorithm, the biological scores (in terms of S value) obtained using MAD (or, variance regularization factor) normalization are found to be inferior to the biological scores with *Maxrange* normalization. For the purpose of illustration, the biological scores with *MAD-M* distance (MAD normalization and Manhattan distance) are provided in Table 4.7 for MN algorithm. Though in most of the cases *Maxrange-E* distance is found to be superior to Euclidian distance and inferior to *Maxrange-M*, for All Yeast data, it performs better than *Maxrange-M* for MN and average linkage algorithms (some results are shown in Table 4.7 for illustration). When the microarray data sets contain experiments with data

Table 4.7: Selected Biological Scores for *Maxrange-E* for different algorithms and *Maxrange-E* distance

		Data Sets			
Algo.	Dist.	Cell cycle	Yeast complexes	All Yeast	Herpes
MN	*MAD-M*	406	1048	2349	38
MN	*Maxrange-E*	420	1061	2450	41
AL	*Maxrange-E*	413	1018	2441	39

value of same dynamic range, like Herpes, then *Maxrange-M* provides identical results with Manhattan distance for all widely used ordering algorithms. However the superiority of *Maxrange-M* is evident when different types of experiments are present in a particular microarray data. For example, superior results are obtained with *Maxrange-M* for most of the available algorithms for the Cell Cycle, Yeast complex and All Yeast data sets (shown in first row for each algorithm in Table 4.4). The available measures for gene distance, like Manhattan distance, Euclidean distance and Pearson correlations, are suitable for the same type of experiments in microarray data, but they are unable to assign different weights of distance for different types of experiments. In contrast, the *Maxrange-M* and *Maxrange-E* distance provides this flexibility, and hence better results are obtained for multiple type of experiments.

4.4.2 Statistical Analysis of *Maxrange-M* Distance Measure and Minimal Neighbor Ordering Algorithm

To statistically compare the performance of *Maxrange-M* distance with Pearson Correlation in case of ordering algorithms, t-tests are performed with the PI (Eq. 4.3) values shown within parenthesis in Table 4.4, using the equation

$$t = \frac{\overline{PI_1} - \overline{PI_2}}{\sqrt{\frac{VariancePI_1}{n_1} + \frac{VariancePI_2}{n_2}}}. \tag{4.4}$$

where, $\overline{PI_1}$ and $VariancePI_1$ are the mean and the variance of all the available PI values for *Maxrange-M* distance in Table 4.4. PI_2 is used for Pearson Correlation and $n_1 = n_2 = 16$, as there are 16 PI values available in total from Table 4.4 for each of the distance measures with 4 datasets and 4 algorithm. So, the degrees of freedom for t-test are $16 \times 2 - 2 = 30$. Similarly, t-test is also performed for *Maxrange-M* distance and Euclidean distance. The two t values and related p values are shown in Table 4.8. The alternative hypothesis (H_1), that the average of 'percentages of improvement over the lowest biological score' for the *Maxrange-M* distance is better than the related one (Pearson or Euclidean), is used in the calculation of t-statistics. The final conclusion, once the test has been carried out, is always given in terms of the null hypothesis (H_0), that there is no difference between the averages of 'percentages of improvement over the lowest biological score' for the two distance measures. We either 'reject H_0 in favor of H_1' or 'do not reject H_0'. After finding the p values (from t-table) for corresponding t values, we reject the null hypothesis for both the cases with significance level 0.001 and 0.02 respectively, which suggests that there is strong evidence against the null hypothesis in favor of the alternative. We have also performed t-tests with the PI values for clustering algorithms (shown in Table 4.5), and the results favored the alternative hypothesis, that the average of 'PI values' for the *Maxrange-M* distance is better than the related one (Pearson or Euclidean) with significance level 0.02 and 0.04 respectively.

Similar types of t-tests with the alternative hypothesis, that the average of 'percentages of improvement over the lowest biological score' for the MN algorithm is better than the related algorithm (FRAG_GALK or B-joseph.), are also performed with the percentages of improvement shown in Table 4.4. The results are shown in Table 4.9. For each algorithm there are 12 PI values (for 4 datasets and 3 distance

Table 4.8: Results of t-test for different pairs of distance measures

	Pairs of distance measure	
	Maxrange-M & Pearson	*Maxrange-M* & Euclidean
t	3.4247	2.1563
p	$0.001 > p$	$0.02 > p$

measures) and hence $12 \times 2 - 2 = 22$ degrees of freedom are available for each t-test. From the results of t-test and p values, the null hypothesis, " there is no difference between the averages of 'percentages of improvement over the lowest biological score' for the two algorithms" is accepted for the pairs MN-FRAG_GALK and MN-B-joseph. Alternative hypothesis, that the average of 'percentages of improvement over the lowest biological score' for 'SOM+MN' is better than SOM, is favored in t-test with the PI values shown in Table 4.6.

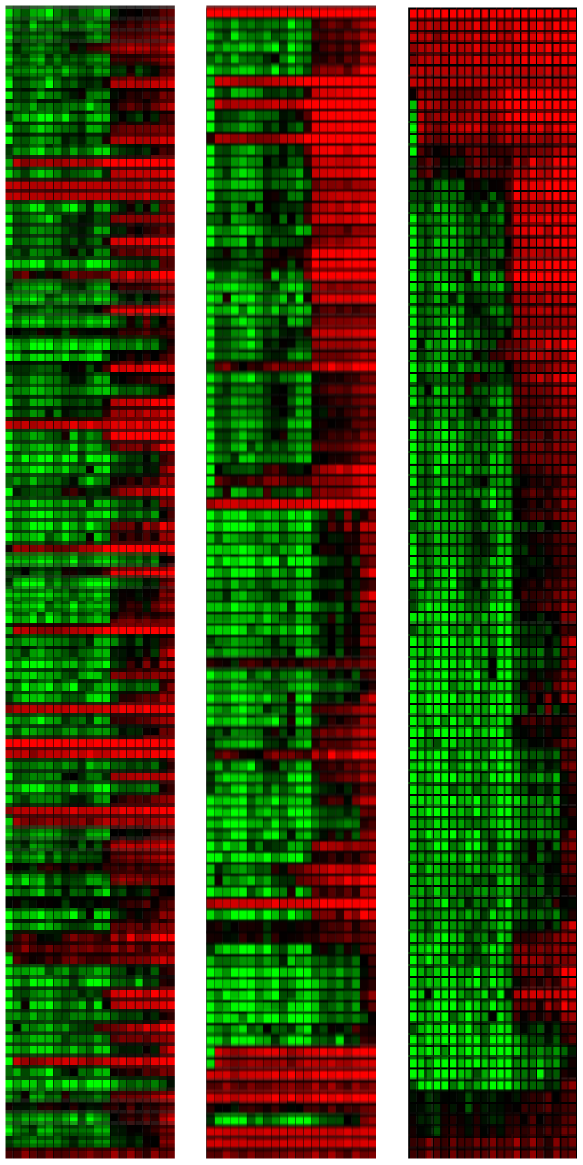
Table 4.9: Results of t-test for different pairs of algorithms

	Algorithm pairs		
	Min. Neigh. & FRAG_GALK	Min. Neigh. & Bar-Joseph et. al.	(SOM + Min. Neigh.) & SOM
t	0.051	0.067	4.103
p	$p > 0.5$	$p > 0.5$	$0.0001 > p$

4.4.3 Subcluster Identification and Grouping of Correlated Genes by Minimal Neighbor

To show how MN helps to identify subclusters within large clusters and groups functionally correlated genes within clusters to improve solution quality of a partitive clustering solution, MN is applied separately on the gene clusters found by SOM. Results/improvements found by combining these two algorithms are shown in Tables 4.2, 4.4, and 4.9. The visual displays 'SOM+MN' are almost identical with 'SOM+FRAG_GALK', shown in Fig. 3.1-b and 3.1-d, for same clustering solutions of SOM using Fibroblast and Yeast Complex data. Moreover, the subclusters identified by MN are similar to FRAG_GALK for Yeast Complex data, which are provided in Table 3.2 and explained in Section 3.5.1.

As mention in Section 3.5.1, Herpes data is clustered with k=5 (5 clusters) using



a) b) c)

Figure 4.3: Comparing CLICK (Fig. a), 'CLICK+MN' (Fig. b), and 'CLICK +FRAG_GALK' (Fig. c) for Herpes data.

K-means. Visual display of the first four clusters are shown in Fig. 3.2-c. After ordering the genes in each cluster with MN, 3 and 3 distinct patterns are observed in cluster 1 and 2, respectively. These patterns are identical with the patterns identified by FRAG_GALK, shown in Fig. 3.2-d.

The only shortcoming of MN is observed in identifying useful patterns for clustering solutions of CLICK, using Herpes data. As mentioned in Section 3.5.1, Herpes data is clustered in 1 group (Fig. 4.3-a) of 101 genes and 5 singleton genes, using CLICK. Fig 4.3-b) and Fig 4.3-c) compares the gene ordering performance of MN and FRAG_GALK in clustering solution of CLICK. Although, some subclusters are identified using visual display (Fig. 4.3-b), but it is clear from the display that the gene ordering is not optimal for MN as compared to FRAG_GALK. The ordering performed by MN is based on local decisions not on global decisions like FRAG_GALK and hence, ordering is not impressive for a large cluster containing 101 genes. However the performance of MN is satisfactory for 5 clusters identified by K-means, containing 48, 32, 10, 12, and 4 genes for Herpes data.

4.5 Conclusion

A new measure called *Maxrange*, for evaluating the distance between genes, and a new Minimal Neighbor gene ordering algorithm are described in this chapter. These are used for efficiently ordering the genes in terms of their expression values for microarray datasets, as well as in individual clusters found by partitive clustering methods for that data sets. The available measures for gene distance, like Manhattan Distance, Euclidean distance, and Pearson correlation, use only one normalization factor (1, 1, and standard deviation respectively) for all types of experiments, although the expression values may differ by orders of magnitude from one kind of experiment to another. As a consequence, the distance between genes may not be properly reflected in these measures for microarray data having different types of experiments. In contrast, normalization is performed separately with different normalizing factors for the different types of experiment in our *Maxrange-M* and *Maxrange-E* distances. This makes it suitable for both single type and multiple type of experiments. As basic distance measure Manhattan/Euclidean distance is used in the *Maxrange* for their insensitiveness to large three fold changes in the gene expression profiles.

The existing ordering algorithms for finding the gene order using genetic algorithms spend most of the time in repetitive search for the lowest value of the sum of gene expression distances in gene groups. So, in MN, the repetitive searching for optimal gene order in partitive clustering solutions, as well as global minimization of distance for distant genes, are avoided. While this results in reduced time complexity ($O(n^2)$) for MN, in terms of biological score it is comparable with FRAG_GALK ($O(n^5)$) and Concorde_LP ($O(2^n)$), the leading TSP solvers currently available. It is also evident from the analysis carried out in Section 4.4 that the user may choose to implement FRAG_GALK to obtain the lowest sum of gene expression distance at a higher computational cost versus MN to obtain equivalent biologically relevant gene order in partitive clustering solutions with much lower computational complexity for clusters containing less than 50 genes.

Huge number of different types of experiment are conducted over genes to find functional correlation between them, by different research groups all over the world. In future more experiments are likely to be appended with the existing microarray data. This demands a distance measure like *Maxrange-M*, and growing number of genes for the same microarray data sets require fast ordering algorithm like MN. It is evident from the experimental results that the *Maxrange-M* with MN performs the best in such situations. As such, this combination seems to be a promising tool for microarray and gene expression related experiments.

Chapter 5

Combining Multi-Source Information through Functional Annotation based Weighting: Gene Function Prediction in Yeast

5.1 Introduction

In the previous chapters, microarray analysis is performed for gene function prediction by analyzing coexpression relationships in a high-throughput fashion. While gene expressions are excellent tools for hypothesis generation, they alone often lack the degree of specificity needed for accurate gene function prediction. This improvement in specificity can be achieved through the incorporation of heterogeneous biological data in an integrated analysis [111]. Increasing quantities of high-throughput biological data have become available in recent years. Many of these, such as phenotypic profiles [20], gene expression microarrays [31], protein sequences [73], KEGG pathway [55], protein-protein interaction data [95, 97], protein phylogenetic profiles [80] and Rosetta Stone sequence [72] assess functional relationships between genes on a large scale. These high-throughput data can be the key to assign accurate functional annotation to a significant number of unclassified genes [111].

The value of combining informations, obtained from different methods, for gene function predictions, has been illustrated by several studies [73, 111]. Marcotte et al. [73] predicted many potential protein functions for Saccharomyces cerevisiae based on a heuristic combination of different types of data, where confidence levels for protein-protein links are defined subjectively on a case-by-case basis. Von Mering et al. [74] first developed quantitative methods to measure functional relationship among genes from three different sources of information (including gene fusion, chromosomal proximity and phylogenetic profiles) and predicted functional modules by using a clustering algorithm. In [111], heterogeneous data sources are integrated in Bayesian network approach and functional modules are predicted by using a clustering algorithm based on the principle of *KNN* algorithm. The network is constructed on some independence assumptions about different data sources and uses conditional probability tables based on information elicited from yeast experts. Lee et al. [62] compared different classes of data (including functional links extracted through literature search) and integrated them by using Bayesian Score. In this approach, all available log likelihood scores, derived from the various data sets and lines of evidences, are added to find a combined similarity. Spirin and Mirny [106] developed algorithms to analyze the structural properties of a predicted interaction network to identify the subsets of genes that are densely connected among themselves, but sparsely connected with others. Yanai and Delisi [126] predicted gene links by combining three different types of links through the union operation. The gene modules predicted by all these studies have shown some level of consistency with the well-established biological concepts as described in MIPS [38], KEGG [55], and other public data-bases.

In spite of the remarkable power and potential to address inferential processes, there are some inherent limitations in Bayesian networks. The problem centers around the quality and extent of the prior beliefs used in Bayesian inference processing. A Bayesian network is only as useful as the reliability of this prior knowledge. An excessively optimistic or pessimistic expectation of the quality of these prior beliefs will distort the entire network and invalidate the result.

While most of the works regarding data source integration are based on Bayesian network, the approach of integrating information from data sources in a linear combination style through functional annotation based weights, is still unexplored. Moreover, all the pre-mentioned works do not incorporate transitive nature of protein homology and KEGG pathway similarity extraction excluding Yeast genes.

In this investigation, we present a new computational framework, using functional annotation based weighting, for the prediction of gene function in yeast through phenotypic profile similarity, gene expression similarity, protein similarity by transitive homology, Kyoto Encyclopedia of Genes and Genomes (KEGG) pathway similarity, interacting protein information and evaluation of these information by MIPS [38] gene annotation. The novelty of our method lies in the way of estimating the weights in a linear combination style, using gene annotations in Eq. 5.7 (described in Section 5.2.3). In a related work, Lee et al. [62] used a single free parameter for estimating weights where they pointed out that a heuristic modification to the strict Bayesian approach performs better for integrating the diverse functional linkage data sets by incorporating the relative weighting of the data (see supplementary material of [62]). In this approach, all available log likelihood scores derived from the various data sets are added with a rank-order dependent weighting scheme. The resulting weighted sum (WS), scoring the functional linkage between a pair of genes, is calculated as:

$$WS = \sum_{i=1}^{n} \frac{L_i}{D^{i-1}}, \tag{5.1}$$

where L represents the log likelihood score for the gene linkage from a single data set, D is a free parameter (weight) roughly representing the relative degree of dependence between the various data sets, and i is the rank index in order of descending magnitude of the n log likelihood scores for the given gene pair. The free parameter D ranges from 1 to α, and is chosen to optimize overall performance (positive predictive value (PPV) and coverage) on the functional benchmark. In this method it is not possible to have equal weights for any two data sources for $D > 1$. Moreover, here the weights follow a strict geometric series, which in most cases will not reflect the relative importance of the data sources. All these limitations are not present in our proposed method. The method and some of the results presented in this chapter have been reported in [93, 94].

5.2 Methods

We mainly focus on integrating phenotypic profiles, microarray gene expression, KEGG pathway related protein database in Protein Information Resource (PIR)

[35], protein sequence similarity by transitive homology, and protein-protein interaction information as data sources. The main steps of our methodology for predicting gene functions can be summarized as:

S1) Extract pairwise similarity of genes, obtained from different data sources (see Section 5.2.1).

S2) Separately re-score the similarities in a common framework of Yeast GO-Slim: Process annotations (see Section 5.2.2).

S3) Integrate the re-scored similarities from different data sources through the proposed scoring framework (see Section 5.2.3) and calculate the combined score.

S4) For each gene g, form a cluster comprising that gene and its K nearest neighbors using the proposed score and predict the function of g by noting the functional enrichment of the cluster using MIPS annotation (see Section 5.2.4).

Each of the above steps are discussed in detail in the following subsections. In brief, in the proposed scoring framework, the weights of the re-scored similarities from different data sources are determined by adaptively maximizing the PPV of the score, using Yeast GO-Slim process annotations [30] of known genes. The weighting scheme enables all possible weighting (including equal and zero weighting) of data sources by first assigning each data source a weight that varies from 0 to α and then optimizing the weights using an objective function, involving the PPV between gene pairs using Yeast GO-Slim process annotation. The aim is to predict gene function from clustering solutions, rather than obtaining detailed interaction relationships among the genes. Our method predicts the functional categories of 12 unclassified Yeast genes from 12 clusters with 98.20 PPV. We also observed that even a small proportion of classified (annotated) genes can provide improvements in predicting true positive gene pairs. Moreover, for evaluating the results further, we merged the available annotations from Yeast GO-Slim process and MIPS for all the genes, and then split the genes into independent training and test sets. The training set is used to determine the weights, while the independent test set is used to compute the PPV and to evaluate the gene pairs and clustering results. The process is repeated 10 times and the cross-validation result is reported.

5.2.1 Data Sources and Similarity Extraction Techniques

Here we describe the different data sources and their respective similarity extraction techniques.

Phenotypic Profile

Recently, Brown et al. [20] presented a method for the global analysis of the function of genes in budding yeast. The method is based on hierarchical clustering of the quantitative sensitivity profiles of the 4756 strains with individual homozygous deletion of all nonessential genes, with each gene replaced by a cassette containing a 20-mer molecular barcode' unique for each deletion mutant. They showed the method to be superior than other global methods for identifying function of genes involved in various DNA repair, damage checkpoint pathways, and other interrogated functions. Analysis of the phenotypic profiles of the 51 diverse treatments places a total of 860 genes of unknown function in clusters with genes of known function. We use this complete phenotypic profile data for quantitative phenotypic profile similarity extraction with Pearson correlation [20].

Let $X = x_1, x_2, \cdots, x_k$ and $Y = y_1, y_2, \cdots, y_k$ be the phenotypic profiles of two genes obtained over a series of k different treatments. Using centered Pearson correlation, the similarity between genes X and Y is defined as

$$Pc_{X,Y} = \frac{1}{k}\sum_{i=1}^{k}\left(\frac{x_i - \overline{X}}{\sigma_X}\right)\left(\frac{y_i - \overline{Y}}{\sigma_Y}\right) \tag{5.2}$$

where $\overline{X}$ and σ_X are the mean and standard deviation of the gene X, respectively. σ_X is defined as

$$\sigma_X = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(x_i - \overline{X})^2}. \tag{5.3}$$

The Pearson correlation has value between -1 and 1, where 1 indicates a linear relationship between the two vectors.

When the phenotypic profile data set is downloaded from the website it is found that out of 51 treatments some treatments are missing for some genes (strains). For the subsequent analysis to be as informative as possible, it is essential that the missing values have to be estimated in order to analyze the available data and the estimates for the missing values are as accurate as possible. Currently, there is no state-of-the-art missing value estimation method for phenotypic profiles. Alternatively, missing values in phenotypic profiles can be estimated using the methods that are used for microarray gene expression. In this phenotypic profile data set, all the genes with more than 50% missing values are first eliminated from the dataset. Thereafter for the remaining genes missing values are estimated using LSimpute [8] software, described in Section 1.6.1.

Gene Expression

We use the All Yeast [31, 119] data for gene expression similarity extraction. Brown et al. [20] have shown that even with 30 distinct biological conditions for gene expression, GO term ribosome biogenesis (GO:0007046) tends to dominate gene pairs implicated by coexpression. As we have already used phenotypic profiles, which implicate gene relationships over a broad range of biological processes, here we only use the widely studied All yeast data. The missing value estimation procedure is similar to that described in Section 3.3.2. Here also we use centered Pearson correlation for extracting gene expression similarity in a similar way as mentioned in the previous section.

KEGG pathway

The pathway information for genes in KEGG [55] can be utilized as a reference for functional reconstruction. All the protein sequences, except Yeast proteins, corresponding to each pathway (121 pathways in the second level) are downloaded from PIR [35]. Profile vector for each protein in Yeast is computed by comparing its sequence across 121 pathway databases, using BLAST [4]. The method is similar to phylogenetic profile [80] construction, where, each pathway database is replaced by all proteins within a species. The pathway profiles of genes, computed using KEGG pathway databases, are denoted as KEGG profiles.

To find the similarity between two genes using KEGG profiles, we used the ratio of dot product value and OR value between two profiles. The similarity matrix has a highest similarity value of 1. Hence, the similarity values, obtained by all pairwise comparison, have a dynamic range from 0 to 1. Note that, the genes, whose protein sequences are not available, are assigned a pathway profile similarity value of 0 w.r.t. all other genes (proteins).

Protein Sequence

Comparing the protein sequences presents an alternative prominent approach for gene annotation and analysis. Sequence-based comparative analysis also proved crucial for deciphering functions of genes and proteins. As the proteins are products of coding regions (open reading frames) of the genes, the integration of expression data similarity with protein sequence similarity for gene analysis could potentially provide new insights into the relation between gene functions. Protein similarity information is one of the major components of biological knowledge, which contains mostly known and validated protein (or gene) relations. Intuitively one can assume that all the protein relations arising from direct protein similarity search is available in the literature and will not help in predicting functions for unclassified genes in a widely studied organism like Yeast. As compared to direct protein similarity search, the field of searching gene/protein similarity through phylogenetic profiles (PP) [80], Rosetta Stone sequence (RS) [72], and transitive homology [78] are relatively new methods and with increasing number of fully sequenced genomes the search space of these methods are increasing rapidly. In this investigation, transitive homologues are used instead of PP and RS, for extracting protein similarity, as its accuracy is reported to be higher than PP and RS in literature [70, 125]. To detect transitive homologues by the third intermediate sequence, 37,66,477 protein sequences are downloaded from UniProt [10].

Transitive homology detection method [70, 78, 125] works by searching the query sequence against the database with a conservative threshold to find the closely homologous sequences and using these homologous sequences as seeds to search the database to find remotely homologous sequences with a less conservative threshold. The method has been shown to be close to the profile [80] based methods and better than a direct pairwise homology search [78]. Our findings are in agreement with [125] that, this homology transitivity can be used as the main source for gene pairing and predicting functions of unknown genes. To find the transitive homologues, homology comparisons are performed among target proteins and 37,66,477 proteins downloaded from UniProt [10], by using BLASTP in BLAST [4]. Before comparison all the yeast proteins are removed from the downloaded database. Let the similarity between two protein sequences A and B be $B_{A,B}$. The value $B_{A,B}$ is replaced by

$B_{A,C} \times B_{C,B}$ if there exists a sequence C such that $B_{A,C} \times B_{C,B}$ is larger than the current value of $B_{A,B}$. This transformation takes advantage of the transitive homology of sequences A and B through the intermediate sequence C, assuming that sequences A and C and sequences B and C are independently homologous [70]. For example, consider the transitive homology between sequence a and sequence b through the third sequence c. The E-values between sequence a and sequence c, sequence c and sequence b, as well as sequence a and sequence b are 0.01, 0.005, and 20 respectively. The protein similarities $B_{a,c}$, $B_{c,b}$, and $B_{a,b}$ are 0.8, 0.9, and 0.2 respectively. The homology between sequence a and sequence b cannot be detected with their direct E-value. However, the value of $B_{a,b}$ is assigned to $0.8 \times 0.9 = 0.72$ because of the transitive sequence homology.

Instead of storing raw BLAST score as the similarity between two protein sequences, we use the metric of ProClust [81] where the metric value scales from 0 to 1. It is the ratio of the raw BLAST score of the sequence alignments to the raw BLAST score of one of those two sequences aligned to itself. Here also the genes, whose protein sequences are not available, are assigned a transitive protein similarity value of 0 w.r.t. all other genes.

Protein-Protein Interaction

Protein-protein maps promise to reveal many aspects of the complex regulatory network underlying cellular function [74]. For this study, manually curated catalogues of known protein-protein interactions are downloaded from BioGRID [95] and binary interactions are used as the common unit of analysis. For a given pair of genes/proteins the similarity value is 1 or 0, indicating a interaction present or absent, respectively. Since the similarity value scales from 0 to 1, its normalization is unnecessary. The BioGRID database/catalogue includes more than 90000 interactions by combining results obtained from synthetic lethality, affinity capture, two-hybrid, epistatic miniarray profile, reconstituted complex, co-crystal structure, co-purification, dosage rescue, phenotypic enhancement, phenotypic suppression, synthetic growth defect, co-fractionation, biochemical activity, synthetic rescue, and protein-peptide based experiments. The related references are available in BioGRID.

5.2.2 Scoring the Similarities in a Common Framework

Our working hypothesis is that each set of data has an intrinsic error rate and a limited coverage but informs us to some extent about the tendency for genes to operate in the same cellular systems and biological processes in the cell. We can therefore construct a more accurate and extensive functional coupling between yeast genes across a broad set of data (experiments). The prerequisite of this strategy is that we have a unified scoring scheme for testing the heterogeneous data sets, even when the data sets are accompanied by their own intrinsic scoring schemes (such as Pearson Correlation for phenotypic profile and gene expression). This re-scoring by a single criterion allows us to directly measure the relative merit of each data set, and then to integrate the data sets with weights that reflect this merit. In this regard, the similarities arising from various heterogeneous data sources are separately re-scored, based on the common framework of Yeast GO-Slim process annotations of genes in the SGD database [30]. Genes/proteins that occur in the same process are presumed to be functionally linked. The proportion of true positive (TP) gene pairs at a particular similarity value (computed from a data source) can be used as a single criterion for re-scoring the similarity values, where TP gene pairs are defined as pairs of genes i and j, such that genes i and j have an overlapping (explicit or implicit) GO (Gene Ontology) term annotation. In [111] proportion of TP pairs (positive predictive value (PPV)) is defined as

$$PPV = \frac{no.\ of\ pairs\ predicted\ by\ method\ that\ share\ common\ GO\ term\ assignment}{total\ no.\ of\ pairs\ predicted\ by\ method}. \tag{5.4}$$

The hierarchical nature of GO and multiple inheritance in the GO structure can lead to evaluation problems if we consider only the particular GO term with which a gene is annotated [111]. To alleviate this problem, we consider the SGD Yeast GO-Slim process annotations, where every gene is annotated in the same level without any tree based structure. For every gene g, that has undergone Yeast GO-Slim process annotation, a vector

$$V(g) = (v_1, v_2, \cdots, v_j) \tag{5.5}$$

is used to represent its category (Yeast GO-slim process) status, where j is the number of categories. The value of v_j is 1 if gene g is in the jth category; otherwise is zero. Based on the information about categorization, the positive predictive value (PPV) at a given similarity value, can be defined as

$$PPV = \frac{\sum_{i=1}^{n} \sum_{m=1}^{j} (V(g_i)_m \times V(g_{ir})_m)}{n}, \tag{5.6}$$

where $\sum_{m=1}^{j}(V(g_i)_m \times V(g_{ir})_m) = 1$ if $\sum_{m=1}^{j}(V(g_i)_m \times V(g_{ir})_m) \geq 1$, g_i and g_{ir} form a gene-pair, n is the number of annotated gene pairs at a given similarity value, and $V(g_i)_m$ represents the m^{th} entry of vector $V(g_i)$.

The PPV can be interpreted as being proportional to the accuracy of the data sources and their ability to predict the cellular/biological processes involved at a given similarity value. In PPV a gene pair is considered as a predicted pair if both the genes in the pair are classified in Yeast GO-Slim process. According to Yeast GO-Slim process and MIPS, there are 6069 and 6131 annotated genes (ORFs) for yeast of which 4387 and 4737 genes, respectively, are classified to some biological or functional process and the remaining genes are unclassified.

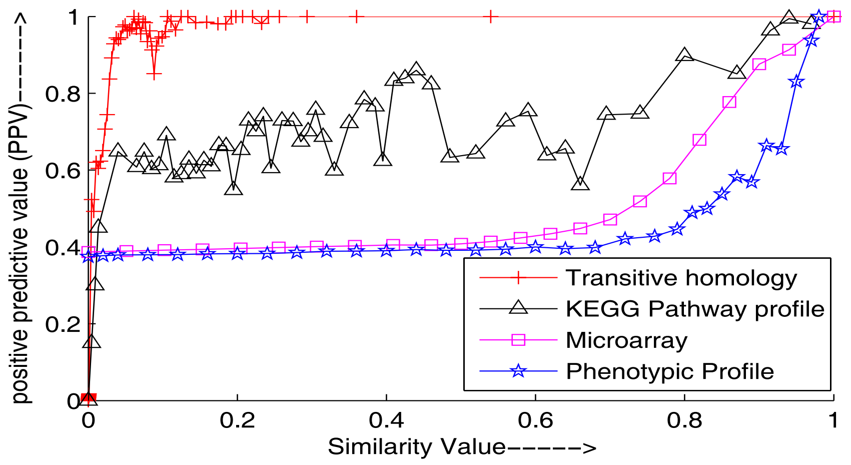


Figure 5.1: Comparing the re-scored similarity values for different types of data sources to obtain equivalency in the common framework of Yeast GO-Slim process annotations. The positive predictive values (PPV) versus the similarity values are plotted for each data source.

Figure 5.1 compares the similarity values obtained from different data sources in terms of their PPV. The PPV for intermediate similarity values, that are not plotted in Fig. 5.1, are calculated from the slopes of the respective curves. The similarities extracted from protein-protein interactions are binary relations in our study. Therefore, PPV for protein-protein interactions has a constant value 0.69 at a similarity value of 1 and hence it is not shown in Fig. 5.1.

5.2.3 New Framework for Data Source Integration

As the similarities computed from different data sources are re-scored (see Section 5.2.2) on a single criterion and common framework of Yeast GO-Slim process annotations, they are directly comparable and can be integrated even when the natures of experiments are distinct (e.g., comparing phenotypic profiles to protein-protein interactions). The PPV reflect the accuracy of similarity values, but do not provide any information about importance/weight of one data source in presence of the other data sources, in predicting gene pairs. Consequently, it will be more appropriate and better if

1. PPV of each data source, in presence of other data sources, is separately weighed by a factor and then integrated;

2. factors are dependent on the PPV of the integrated PPV of different data sources.

Such an attempt is made in this article with a new score where, PPV computed from phenotypic similarity (Pp), gene expression similarity (Pm), KEGG pathway profile similarity (Kp), protein similarity through transitive homologue (B), and protein-protein interaction information (I) between two genes X and Y are integrated through weights a, b, c, d, and e in a linear combination style. This score is referred to as *Biological Score* (BS) and is defined as

$$BS_{X,Y} = \frac{a \times Pp_{X,Y} + b \times Pm_{X,Y} + c \times Kp_{X,Y} + d \times B_{X,Y} + e \times I_{X,Y}}{a+b+c+d+e} \tag{5.7}$$

where a, b, c, d, and e are varied within range 0 to α in steps of 1 to find a combination that maximizes the PPV for a user defined number of top gene pairs. Note that, the weights a, b, c, d, and e are assigned to the complete PPV matrices calculated from individual data sources. The following can be stated about the score:

1. $0 \leq BS_{X,Y} \leq 1$
2. $BS_{X,Y} = BS_{Y,X}$ (symmetric).

The proposed scoring framework for data source integration, in Eq. 5.7, is based on data source weighting where the re-scored similarity spaces, available from different data sources, are adaptively transformed using a set of weighting coefficients. Intuitively, more important similarity spaces should be assigned larger weights than less important ones, while irrelevant ones should be assigned zero weight. Although the proposed framework has some common working principle with feature weighting (FW) [121], it cannot be categorized as FW because what is computed using BS is the pair-wise gene similarities and not the set of features of any individual gene.

Estimation of Weights for Maximization of PPV: We maximize the PPV, using Yeast GO-Slim process annotations, for top gene pairs by varying the weights a, b, c, d, and e in the BS (Eq. 5.7). For each set of values of a, b, and c, the top gene pairs are identified with a gold standard cut-off value. Our gold standard cut-off value and gold standard of top gene pairs are determined from KEGG pathway profiles, which provides 26432 gene pairs with similarity value 1 and constant PPV of .81. These gene pairs are the most accurate of all, whereas the accuracy (PPV) of other data sources, as well as gene pairs below top 26432 for KEGG pathway profiles, vary considerably. We now use the following steps to estimate the weight factors a, b, c, d, and e in the *Biological Score*:

S1) All the factors are assigned an initial value of 1.

S2) BS values are calculated for all the gene pairs and sorted in descending order to identify the cut-off value above which the top 26432 gene pairs are available.

S3) PPV is calculated for the top 26432 gene pairs.

S4) The weight factors are now varied in steps of 0.1 and the steps from S2 to S3 are repeated to find a combination of weights for which the PPV is maximized.

Note that, experiments are also conducted by assigning a value of 0 to the different weight factors and then varying them. Figure 5.2 shows how PPV, using Yeast GO-Slim process, varies for different values of weight factors ranging from 0

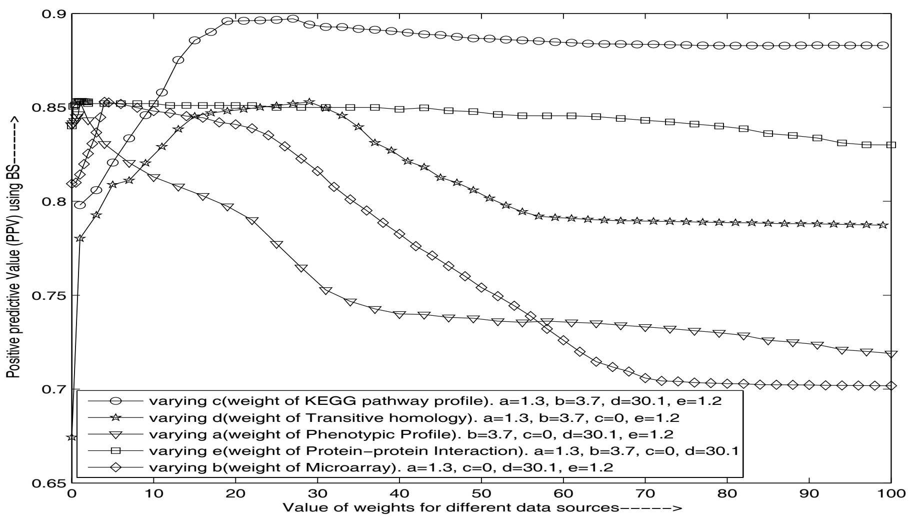


Figure 5.2: Comparing the values of PPV using BS, by varying weights of PPV of different data sources for top 26432 gene pairs. When a particular weight is varied the other weights are kept constant at the values shown in the figure. The curves obtained with c=0 indicate the instances when KEGG pathway profile is not included in the integration process.

to 100, in steps of 1. The curves show instances where one weight factor is varied and the other weight factors are kept constant.

5.2.4 Gene Function Prediction

For biological function prediction of each gene, a cluster comprising that gene and its K nearest neighbors is computed using the proposed score (BS). The function for each gene is predicted from the top K neighbors and selecting a gold standard BS cut-off value obtained from KEGG pathway profiles using MIPS October 2005 classification. The gene clustering method is denoted as *K-BS*, where each gene is considered once for its function prediction and allows its neighbor genes to be a member of multiple gene clusters. This clustering method, based on K nearest neighbors of each gene, is already used in previous related investigations of Marcotte et al. [73] and Troyanskaya et al. [111]. As Yeast GO-Slim process annotations was used for determining the weights of the data sources, 510 different MIPS (October 2005) functional categories are used to evaluate the biological significance of the clusters generated by our *K-BS*. One or several predominant functions are then assigned to each cluster and the target gene (the gene whose K nearest neighbors, using BS as a similarity value, are considered to form the cluster) by calculating the P-values for different functional categories. The probability (P-value) of observing at least m genes from a functional category within a cluster of size n is given by

$$P = 1 - \sum_{i=0}^{m-1} \frac{\begin{pmatrix} f \\ i \end{pmatrix} \cdot \begin{pmatrix} N-f \\ n-i \end{pmatrix}}{\begin{pmatrix} N \\ n \end{pmatrix}} \tag{5.8}$$

where f is the total number of genes within a functional category and N is the total number of genes within the genome (6131).

5.3 Results

As Yeast GO-Slim process was used for determining the weights of the data sources, MIPS annotation is now used to evaluate the performance of *BS*. Genes and their corresponding proteins are denoted by different symbols or identifiers in different data sources. data source integration requires that all genes/proteins are denoted according to a common naming scheme. We mapped genes from different resources to their MIPS identifier. Genes/proteins that could not be mapped to their MIPS identifier are eliminated. Our gold standard PPV of top gene pairs is now changed and determined from KEGG pathway profiles, which provides 26432 gene pairs with constant PPV of .8874, using top level classification of MIPS annotation. In this section, first we present the comparisons of our method with Lee et al.'s [62] probabilistic network and individual data sources in Section 5.3.1. Influence of number of classified genes on the proposed scoring framework is demonstrated in Section 5.3.2. In Section 5.3.3 various paremeters involved in the clustering method and the biological significance of some clusters are addressed. Finally, the performance of *BS* and some comparisons based on independent training (estimating weight factors) and test set with null intersection are presented in Section 5.3.4.

5.3.1 Comparative Performance of Methods and Data Sources

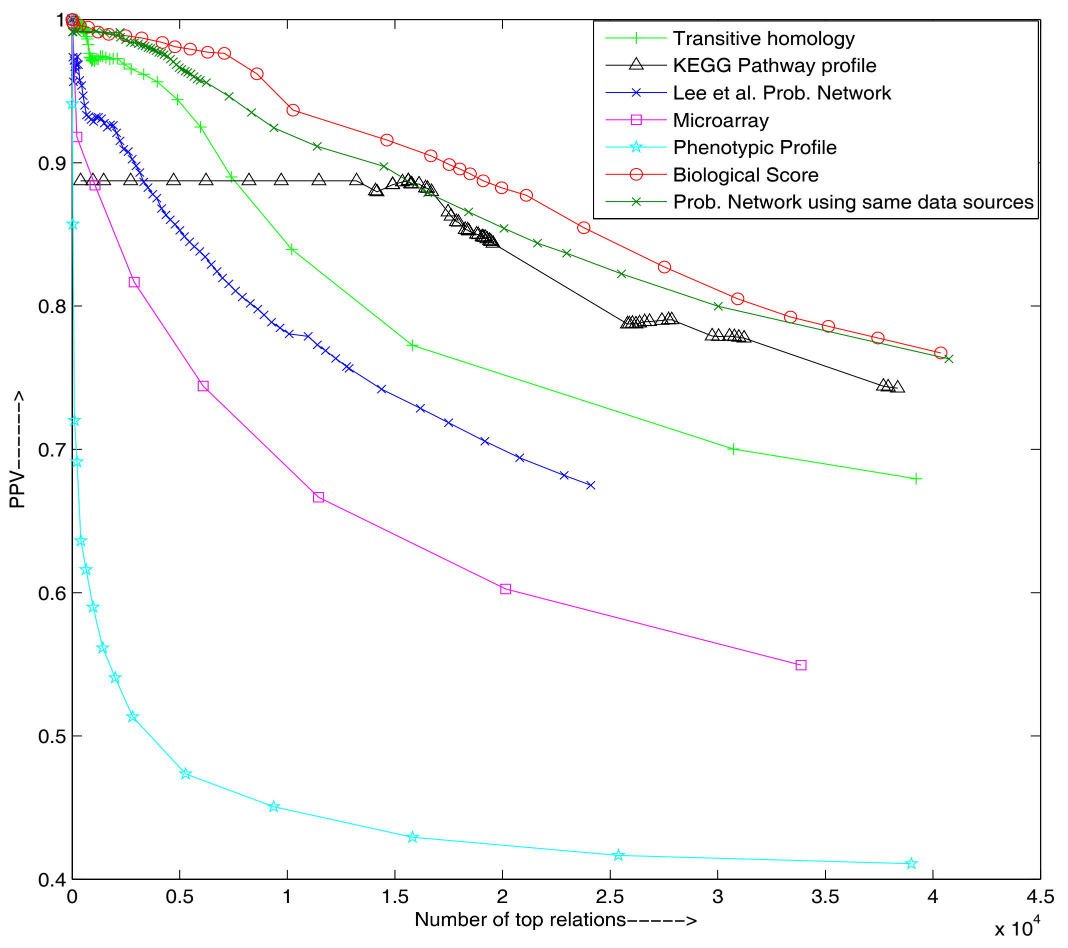


Figure 5.3: Comparison between the *Biological Score* (BS), Lee et al.'s Probabilistic Network, and individual data source in terms of PPV versus the number of top gene pairs. While, the available annotations (using vector $V(g)$ in Eq. 5.5) from Yeast GO-Slim process is used to train the weighting factors in BS and 'Probabilistic Network using same data sources', the available annotations from MIPS are used to evaluate (using PPV) the gene pairs of all the methods and data sources.

In order to demonstrate the power of data source integration, we compare the PPV of gene pairs identified by the BS (Proposed scoring framework for data source integration) with those identified by the individual data sources. Since the proposed method (BS) uses GO annotations for adapting its weights, it is not used for performing the comparisons. Rather, the MIPS annotation of classified genes is used (Fig. 5.3). We sorted the similarity values computed from *Biological Score* (BS), phenotypic profiles, gene expression, KEGG profiles, and protein similarity from transitive homology in descending order, and drew a curve for top gene pairs verses PPV from the sorted data for each form of data source. In contrast, PPV for protein-protein interactions has a constant value of 0.69 and not shown in Fig.5.3. We found that the curve of BS is above the other curves. Moreover, the top 26432 gene pairs has an PPV greater than the gold standard KEGG pathway profiles. The gene pairs are also reasonably distinct from gene pairs of KEGG pathway profiles. It demonstrates that the proposed *Biological Score* achieved higher PPV by combining similarities from multiple sources. Similarities supported by diverse forms of sources are more likely to be correct. This highlights the merit of data source integration. Figure 5.3 also compares the performance of BS and 'final log likelihood scores' of Lee et al.'s probabilistic network (downloaded from the website mentioned in [63]) in terms of PPV with MIPS annotation. The curve of Lee et al.'s probabilistic network is drawn from top 34,000 gene pairs, as mentioned in [62]. For a direct comparison between our method and the probabilistic network, we implemented the probabilistic network as described in Lee et al. using the same datasources as in Biological Score (BS) and plotted the respective curve in Fig. 5.3. From the figure it is clear that the top gene pairs identified in this investigation is better than any other existing network or data sources. The above statement is true not only for gold standard 24632 gene pairs but also for top 80000 gene pairs which can be used further for any gene network or gene function prediction. We found that beyond top 80000 gene pairs the performance of our method is gradually converging to the performance of probabilistic network (with same data sources) but, it does not hampers the superior performance of our method as only a fraction of top gene pairs are generally used [62] for gene function or network prediction. It is also evident from the results that the choice of data sources is a very important factor in data source integration. For example, protein homology and KEGG profile individually performs better than probabilistic network and considered as two important data sources in the proposed BS. The top $1,00,000$ gene pairs predicted by our method with PPV above 0.755 (not shown in the data) are available in http://www.isical.ac.in/~scc/Bioinformatics/AdS/toprelation.txt in tabular (tab delimited) form. The PPV computed from individual data source are also shown in the file.

5.3.2 Influence of Number of Classified Genes on Functional Annotation based Weighting

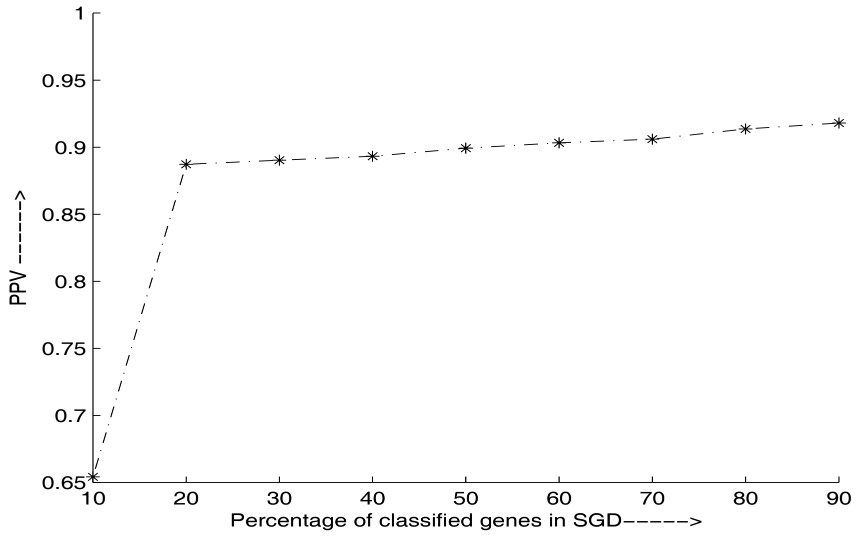


Figure 5.4: Variation of PPV, using *BS*, with nine different percentages of classified genes.

Here we study how the increase in the number of classified genes in Yeast GO-Slim affects the PPV for the classified genes in MIPS for top 26432 gene pairs using *BS*. We found that even with 20% of classified genes the estimated values of a, b, c, d, and e, in maximizing PPV, differs by an amount of 1 than the estimated values with 90% of classified genes. Hence, the value of PPV also varies by an amount of 0.02 to 0.03 with classified genes ranging from 20% to 90%. Fig. 5.4 shows that the percentage of classified genes clearly has a limited contribution to the accuracy (PPV) of the *BS*. Thus *BS* may also be successfully used for organisms where the number of classified genes is as low as 20%.

5.3.3 Gene Function Prediction based on Clustering Results

Genes (open reading frames) are considered to be linked if they are among the 10 closest neighbors within a given distance or similarity cut-off [73]. The biological function for each gene is predicted from the cluster consisting the top 10 neighbors of that gene by selecting K to be at most 10 and BS cut-off value of 0.77. Above this cut-off value the gold standard PPV of 0.8874 is achieved for 36033 gene pairs using the MIPS October 2005 classification. We found several clusters to be significantly enriched with genes of a similar function. Clusters with P-values greater than 10^{-5} are not reported.

To predict a genes function from it's neighbor genes we use the following steps:

S1) 2507 clusters are identified with at-least three or more members by selecting $K = 10$ and with BS gold standard cut-off value 0.77.

S2) Out of these clusters, 1915 clusters are identified with functional enrichment in one or more categories and P-values less than 10^{-5}.

S3) From functionally enriched clusters we predict the functions of 1855 classified and 60 unclassified genes by assigning the function related with the smallest P-value. This ignores the possibility that a gene may be assigned more than one highly significant function, but in practice resulted in more accurate predictions than if multiple functions are allowed per cluster.

The functions of 1855 classified genes are predicted with 95.16 PPV. In general we can say that the possibility of 60 unclassified Yeast genes to match with the predicted functions is 95.16%. The functional enrichment, in one or more categories, for clusters intended for 60 unclassified yeast genes are available in tabular form (tab delimited file) at http://www.isical.ac.in/~scc/Bioinformatics/AdS/unclassifiedprediction.xls. The function with the smallest P-value in the table represents the predicted function for the unclassified gene, and the three values in the parenthesis denote the function related P-value, function related no. of genes in the cluster, and the function related no. of genes in the genome, respectively. The table also includes all the genes within each cluster, the PPV (between target gene and the neighbor gene) arising from various data sources, and the BS values. A table with similar format, containing the predicted functions of 1855 classified yeast genes is available at http://www.isical.ac.in/˜scc/Bioinformatics/AdS/classifiedprediction.xls.

Out of 60 unclassified genes, YEL041W and YDR459C are now (April 2007) classified in MIPS, and our function predictions for these two genes are in agreement with present MIPS classification. YEL041w and its four neighbors YJR049C, YPL188W, YDR226W, and YER170W form a cluster. From the functional enrichment of the cluster we predict that YEL041w is related with the category 'phosphate metabolism' as the four remaining genes belong to this category. The prediction is right according to MIPS (April 2007) classification with p-value 1.42×10^{-6}. We further manually analyze the cluster and predict that the gene YEL041w may be related with category 'metabolism of vitamins, cofactors, and prosthetic groups' and 'homeostasis' since it's two top neighbor genes YJR049C and YPL188W is related with these categories. While the prediction of the category 'metabolism of vitamins, cofactors, and prosthetic groups' is a correct (MIPS 2007) one, the prediction 'homeostasis', may be a novel one for YEL041w. Moreover, according to MIPS, YEL041w has the highest similarity to YJR049C, which is related to homeostasis of metal ions (Na, K, Ca etc.).

The cluster containing gene YDR459C and its ten neighbor genes, YOL003C, YNL326C, YLR246W, YPR193C, YIR042C, YMR127C, YNL035C, YBL052C, YDR126W and YPR051W shows functional enrichment in categories 'protein modification' (8 out of 11, P-value 1.16×10^{-6}), 'modification with fatty acids (e.g. myristylation, palmitylation, farnesylation)' (4 out of 11, P-value 2.3×10^{-7}) and 'modification by acetylation, deacetylation' (4 out of 11, P-value 4.4×10^{-6}). We correctly predict that YDR459C is related to 'modification with fatty acids'. The hierarchical nature of MIPS annotation automatically ensures that YDR459C is related to 'protein modification', which is placed at one level higher than 'modification with fatty acids'. Although the remaining function, 'modification by acetylation, deacetylation', is also significant in terms of P-value, YDR459C is not related with this function and our results are in agreement with our approach of considering the function involving the lowest P-value. The cluster also contains three unclassified (MIPS 2007 classification) genes YDR126W, YIR042C, and YNL035C. Although the cluster is not intended to predict the function of these three genes we can assume that these genes may be related with the function 'protein modification'.

Table 5.1: Top 12 function predictions of unclassified gene at BS cut-off value of 0.77

Unclassified Gene	Functional category	P-value	Genes within cluster	Genes within category
YIL080W	ABC transporters	2.2204e-16	8	28
YLR057W	modification with sugar residues	2.2871e-14	8	67
YHR218W	DNA topology	0	9	52
YHR219W	DNA topology	0	10	52
YIL170W	C-compound and carbohydrate transport	1.3656e-14	8	63
YDR441C	purin nucleotide/nucleoside/ nucleobase metabolism	6.7724e-15	8	58
YCL074W	TRANSPOSABLE ELEMENTS, VIRAL AND PLASMID PROTEINS	3.3307e-16	8	34
YBL112C	DNA topology	0	10	52
YLR464W	DNA topology	2.6645e-15	8	52
YMR010W	modification with sugar residues (e.g. glycosylation, deglycosylation)	0	9	67
YIL067C	vesicle fusion	2.2204e-16	9	32
YHR049W	metabolism of secondary products derived from glycine, L-serine and L-alanine	3.3307e-16	7	19

Our top predictions consist the function of 12 unclassified (MIPS 2007) and 417 classified genes at BS cut-off value 0.77, and P-value cut-off 1×10^{-13}. At these cut-off values, the functions of the classified genes are predicted with 98.20 PPV. Table 5.1 summarizes the top 12 predicted functions for 12 unclassified genes. For each of the predicted functions, the related p-values, no. of related genes in the cluster and the genome, is also shown in the table. Each of the clusters contain 11 genes and they are available in the table representing 60 clusters for function prediction of unclassified genes. since four of the 12 clusters show functional enrichment in a single category of 'DNA topology', we analyze these clusters manually. We observe that 15 classified (YBL113c, YDR545w, YEL077c, YER190w, YGR296w, YHL050c, YIL177c, YJL225c, YLL066c, YLL067c, YLR466w, YLR467w, YNL339c, YPL283c, and YPR204w), 4 unclassified (YHR218W, YHR219W, YBL112C, and YLR464W) and 2 recently deleted (YEL076C and YPR203W) genes are distributed in these clusters with 80% genes in common. We further perform clustering with K-BS by selecting $K = 20$ to find if these four clusters merge to form a single cluster. After clustering, all the 21 genes are found in the same cluster, which shows functional enrichment in categories 'CELL CYCLE AND DNA PROCESSING' (15 out of 19, P-value 1.53×10^{-10}), 'DNA processing' (15 out of 19, P-value 7.02×10^{-15}) and

'DNA topology' (15 out of 19, P-value 2.38×10^{-30}). Our analysis predicts that the four unclassified genes are very likely to be involved in the above mentioned processes. On examination of the literature for 4 unclassified genes, we find that their involvement in DNA processing and DNA topology is likely due to their relation to helicase-proteins [30, 102]. These proteins play important roles in various cellular processes including DNA replication, DNA repair, RNA processing, chromosomal segregation, and maintenance of chromosome stability. It has been well known that the amino acid sequences of these proteins contain several conserved motifs, and that the open reading frames (ORFs) which encode helicase-related proteins make up several gene families [102]. While YHR218W encodes helicase-like protein within the telomeric Y' element, YHR219W encodes protein that is similar to helicases and contains telomeric short Y' element [30]. YBL112C and YLR464W also contain helicase-encoding repetitive sequence and lies within TEL02L (subtelomeric region next to the telomeric repeats) and TEL12R, respectively.

5.3.4 Evaluation Based on Independent Training and Test Sets

The performance of the proposed integration method relies critically on Yeast GO-Slim process annotations in order to determine the weights of the data sources in the training process and its evaluation depends on MIPS annotation in the test process. But to perform a fair evaluation, the training and test set should be independent with null intersection. In this regard, we also experimented with an alternative method based on cross-validation. First, we merged the available annotations (using vector $V(g)$ in Eq. 5.5) from Yeast GO-Slim process and MIPS for all the genes, and then split the genes (with annotation vectors) into independent training and test sets. Because data are integrated using weights derived only from the training set, the performances measured on the remaining test benchmark are expected to be free from circular logic and memorization of the annotation set during the training procedure. Moreover, the KEGG pathway profile dataset is now excluded from the datasource integration procedure as pathway information may be a bit redundant with functional annotations available in MIPS and Yeast GO-Slim process.

We randomly separated the set of 6,072 genes into 2 disjoint training and test subsets of 3,036 genes each. All links among genes within the same training subset are calculated and then used for training the weights. Similarly, all links among genes within the same test subset are calculated, with neither links nor genes shared between the training and test sets. The calculation of weights for data source integration and all other steps prior to the final assessment of *BS*, are performed using only the training set. The final assessment is performed on the independent test set. The cross-validation procedure is repeated 10 times and the performance of *BS* is evaluated.

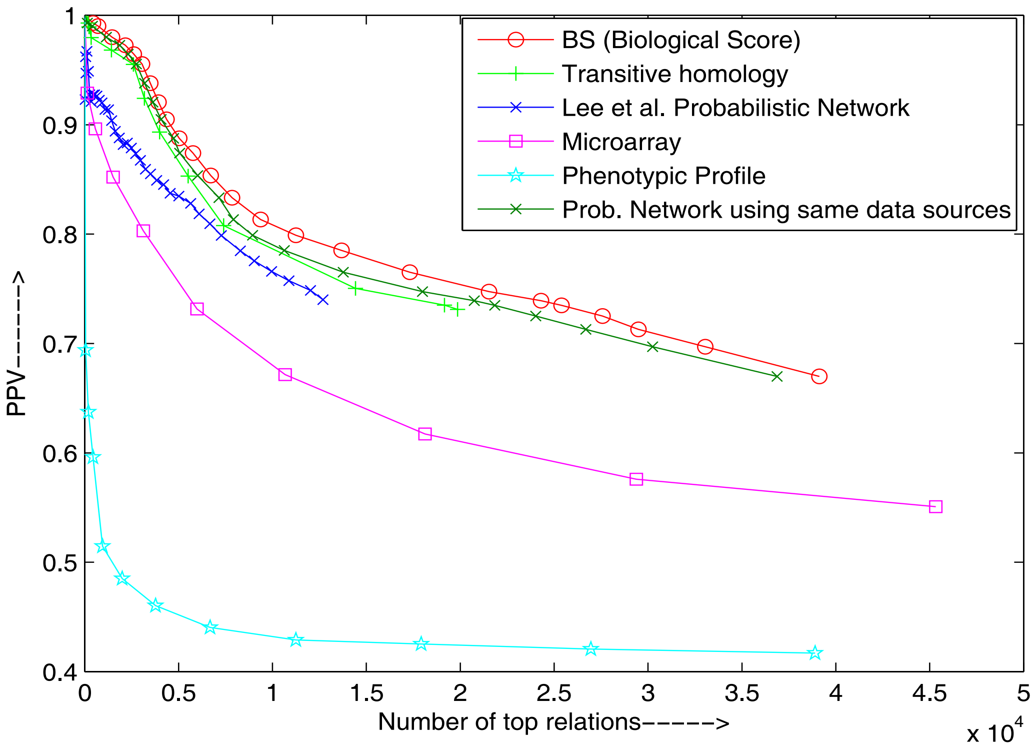


Figure 5.5: Comparison between the *Biological Score* (*BS*), Lee et al.'s Probabilistic Network, and individual data source in terms of PPV versus the number of top gene pairs. The available annotations (using vector $V(g)$ in Eq. 5.5) from Yeast GO-Slim process and MIPS are first merged for all the genes, and then the genes (with annotation vectors) are randomly splited into disjoint training and test sets. While, the training set is used to determine the weighting factors in *BS* and 'Probabilistic Network using same data sources', the test set is used to evaluate (using PPV) the gene pairs of all the methods and data sources.

Fig. 5.5 shows the curves comparing *BS* and individual data sources in terms of PPV for top gene pairs, in one of the cross-validation procedures. Similar curves are obtained when the cross-validation procedure is repeated. The curves show that BS performs better than Lee et al.'s Probabilistic Network and individual data source. In clustering solutions using K-BS, on average 800 clusters are identified with functional enrichment in one or more categories by selecting K to be at most 10, BS cut-off value 0.77, and P-values less than 10^{-5}, by repeating the cross-validation procedure. From functionally enriched clusters, on average we predict the functions of 500 classified genes with 96.2 PPV and 300 unclassified genes by assigning the function related with the smallest P-value. In one of the cross-validation process (out of 10 repetitions), functions of 516 classified yeast genes are predicted with 97.1 PPV from 516 clusters. For the purpose of illustration, the predicted functions of 516 classified yeast genes are uploaded at http://www.isical.ac.in/~scc/Bioinformatics/AdS/classifiedpredictionreview.xls. The PPVs reported here are higher (on average 96.2) than that reported (on average 95.16) in Section 5.3.3 because, a part of Yeast GO-Slim process annotations, which on average have more genes in each functional process than pure MIPS annotations, are now included in the cluster evaluation procedure.

5.4 Conclusion

In the study made in this chapter, we proposed a framework for data source integration that combines information from different sources to predict gene functions. We employed functional annotation based weighting of data sources through annotations of classified genes to predict gene pairs for yeast from five data sources, namely, phenotypic profiles, gene expression data, KEGG profiles, protein-protein interaction and protein sequence similarity through transitive homologues. Functional categories of 60 unclassified (MIPS October 2005) Yeast genes and 1855 classified genes are predicted with 95.16 PPV. Evaluation on the predicted gene pairs confirmed the validity and potential value of the proposed framework for gene function prediction.

Although a neighbor based clustering method needs a user defined neighbor number, from this investigation we find that K-BS is a highly accurate and efficient gene function annotation tool. The system integrates heterogeneous biological information in a functional annotation based weighting framework, leading to more biologically accurate gene groupings, which can be used for gene function prediction. The flexibility of the system allows for easy inclusion of other data sources by first benchmarking them, and then adaptively estimating the individual weights. Furthermore, one can examine the proposed framework on a larger test-bed by including similarities arising from gene-fusion and gene-order conservation based methods.

Chapter 6

Conclusions and Scope for Further Research

The area of life sciences has experienced explosive growth in terms of research and advances in bioinformatics, of which microarray technology is one of the most promising tools. The advances in microarray technologies have resulted in a significant increase in the amount of genomic data. It enables the monitoring of the expression levels of thousands of genes simultaneously. Due to the large quantity of information available from microarray experiments, it is necessary to employ advanced computational methods for classification of the data in order to obtain initial conclusions about the biological functions and pathways related to the genes. The rationale for applying computational methods to facilitate the understanding of various biological processes mainly includes:

- To provide a more global perspective in experimental design
- To capitalize on the emerging technology of database-mining : the process by which testable hypotheses are generated regarding the function, pathway or structure of a gene or protein of interest by identifying similar expression or sequences in well characterized organisms.

While gene ordering in partitive clustering framework and dynamic range based normalization are successfully investigated in Chapter 2 and 3, respectively, to provide a more global perspective in experimental design, multiple heterogeneous data sources are integrated with microarray gene expression data to capitalize on the emerging technology of database-mining through which testable hypotheses are generated regarding the function of a gene or protein.

An overview of conventional GAs and the relevance of TSP to handle microarray gene ordering problem efficiently is provided in Chapter 2. GAs appear to be a very powerful artificial intelligence paradigm to handle the combinatorial optimization problems. There are three general characteristics that might appear to limit the effectiveness of GAs. First, the basic selection, crossover and mutation operators are common to all applications; so researches are now focussed to design problem specific operators to get better results. Second, a GA requires extensive experimentation for the specification of several parameters so that appropriate values can be identified. Third, GAs involve a large degree of randomness and different runs may produce different results; so it is necessary to incorporate the problem specific domain knowledge into GA to reduce randomness and computational time. In this direction, an algorithm called FRAG_GALK is developed in Chapter 2 for solving the TSP and gene ordering problem by incorporating domain specific information in GAs.

The FRAG_GALK, along with its two new operators - 'nearest fragment operator' (NF) and a modified version of order crossover operators (MOC), are described in Chapter 2. The NF reduces the limitation of Nearest Neighbor (NN) heuristic in path construction by determining the optimum number of fragments in terms of the number of cities and then greedily reconnecting them. Although the fragment reconnection is performed on local decisions, the random slicing of chromosomes (in GA) into optimum number of fragments, augments the search space quickly and helps FRAG_GALK in obtaining the global optimal solutions to some of the MGO problems and TSPLIB [116] instances; the largest having 13,509 cities. It will be worthwhile to test the performance of NF and FRAG_GALK on larger TSP instances. For solving larger TSP instances it may be necessary to connect the fragments on global decisions, i.e.,

a) checking the fitness of the connected fragments before accepting them as a solution, and

b) searching for chromosomes with higher fitness by connecting all possible combinations of fragments and not by greedily connecting them on local decisions.

The modified version of order crossover operator (MOC) handles the indefinite computational time due to random length of substring and its random insertion in order crossover. This is done by systematically determining an appropriate substring length from the parent chromosome for performing crossover. While the position of the substring in the parent chromosome is chosen randomly, the insertion of the substring is performed in the position of the last deleted city in another parent chromosome. Thus NF and MOC operators are capable of aligning more genes with the same group next to each other compared to other algorithms, thereby producing better gene ordering for FRAG_GALK.

The representation used in FRAG_GALK is a direct one (integer i= city/gene i) and also used in all other state-of-the-art TSP solvers using genetic algorithm and LK heuristic based approaches. An indirect representation, like offset-based representation, in general takes more computational time in representation, whereas, there is no chance for improving the solution quality over optimal results for most of the TSP instances.

Although there is a rich literature on gene ordering in hierarchical clustering framework for gene expression analysis, outside the framework of hierarchical clustering, previously, different gene ordering algorithms are applied on the whole data set and the domain of partitive clustering remained unexplored with gene ordering approaches. To the best knowledge of the author, the investigation reported in Chapter 3, is the first work on addressing and evaluating the importance of gene ordering in partitive clustering framework. A method for optimally ordering the genes within clusters, obtained from partitive clustering methods, is presented in Chapter 3. The rationale for ordering genes in partitive clustering solution is to regain the relationship among the genes within clusters, which are generally lost for partitive clustering methods. There are several ways in which one can extend the results presented in this investigation. One interesting open problem is the automatic identification of clusters from gene ordering results. Current research is going on this direction. Another interesting (though more theoretical) problem is the extension of the ordering algorithm for a two dimensional clustering solution, where the goal is to order both the rows and the columns of the resulting outcome simultaneously.

Throughout the last decade, visual display of gene expression patterns has proven to be a useful tool for gene expression analysis. The utility of visual display is extended from clustering approaches to gene ordering methods. In Chapter 3, its utility is further extended in identifying useful patterns from ordered genes in partitive clustering solutions. Several examples on microarray gene expression data reveals that the results of integrating gene ordering with partitive clustering are superior to the results obtained using only partitive clustering methods. Moreover, the method requires less computation time and in certain cases provides better results than leaf ordering in a hierarchical clustering framework, proposed by Bar-Joseph et al. It is also evident from our investigation that the method not only regains the relationship among genes, but also helps to identify subclusters of functionally correlated genes by means of visual display (heat map) of gene expression patterns.

Since the available measures for gene distance, like Manhattan Distance, Euclidean distance, and Pearson correlation, use only one normalization factor (1, 1, and standard deviation, respectively) for all types of experiments, gene expression values in lower dynamic range do get dominated by those with higher dynamic range in computation of gene expression distance. To overcome this situation, normalization of gene expression data with experiment specific linear dynamic range of photo multiplier tube is performed in the first part of Chapter 4. The over-sensitivity of Pearson correlation to large three fold changes (peaks) in gene expression profiles is handled with Manhattan distance. The above mentioned normalization process along with Manhattan distance is defined as *Maxrange-M* distance and its effectiveness is demonstrated in finding biologically meaningful gene order. Superiority of *Maxrange-M* to related distance measures and normalization processes are extensively established with widely studied microarray gene expression data.

Relationship between genes often exists by locally similar patterns rather than globally similar patterns in their expression profiles. That is, the complete profiles share similar sub-profiles, which might furthermore be time-shifted. Apart from that, often not all genes respond to all conditions (or time points) in an experiment. In that case, the use of a global similarity measure, i.e., the computation of a similarity degree between the complete expression profiles, will not reveal the relationship between two genes, simply because this degree might be rather small [11]. Extension of *Maxrange-M* distance in finding local and time shifted distance between gene expression profiles will be an interesting application towards further work.

A new algorithm called Minimal Neighbor (MN), for computationally effective gene ordering is developed in the second part of Chapter 4 (Section 4.3.4). As mentioned in Section 2.3.1, the nearest neighbor (NN) tour construction heuristic is a common choice in TSP to construct the initial solutions. It has $O(n^2)$ time complexity and contains only a few severe mistakes in edge construction. Consequently, some concepts of NN heuristic are incorporated in the MN algorithm to achieve biologically meaningful gene ordering in short computation time. Though MN may not be optimal in terms of the summation of gene expression distances like FRAG_GALK (as evident from Table 4.2 and visual display from Fig. 4.3-b), from the biological scores (Table 4.4) and t-test results (Table 4.9), it is evident that MN provides biologically comparable gene order with respect to FRAG_GALK for all datasets and distance measure. More important is that the time complexity of MN is $O(n^2)$, whereas the time complexity of FRAG_GALK is $O(n^5)$, where n is the number of genes. As mentioned in Section 4.4.3, the only shortcoming of MN is observed in identifying useful patterns, through visual display, for a clustering solution of CLICK, containing 101 genes of Herpes data. However the performance of MN, as evaluated by visual display, is satisfactory for 5 clusters identified by K-means, containing 48, 32, 10, 12, and 4 genes for Herpes data. Therefore, it is preferable to use MN algorithm instead of FRAG_GALK for clusters containing less than 50 genes.

The neighbor connecting procedure of MN is based on local decisions. In Step 2 of MN algorithm (described in Section 4.3.4), only the two end genes of the new array are considered and two closest genes for each of them are searched from the remaining genes. An improvement of this step can be performed by considering two groups of gene (from the ordered genes) at the two ends of the new array and the average expression profile of each group can be used for searching closest gene. Obviously an open problem will be how to define the size of the group. A possible solution is to assign a user defined threshold of homogeneity and include all the genes in the group that satisfy the criterion.

One of the important goals of biological investigation is to predict the function of unclassified gene. Since in a model organism like Yeast, there are more than

1000 genes with unknown biological function defined in Munich Information for Protein Sequences (MIPS) and Saccharomyces Genome Database (SGD), Yeast is an automatic choice for gene function prediction. Although the high-throughput data achieved from microarray gene expressions can be the used to assign functional annotations to unclassified genes, the degree of specificity, needed for accurate gene function prediction, can be improved by integrating other data sources with gene expressions. An approach in this direction is investigated in Chapter 5. It involves identification of a group of K closest classified genes of an unclassified gene and assigning the common biological function of the group to that unclassified gene. In order to identify the group, a new methodology, called *K-BS*, is developed by combining different sources from emerging technologies of biological database-mining. The sources of information used in the investigation are microarray gene expressions, transitive homology of protein sequences, phenotypic profiles, protein-protein interaction and KEGG pathway profile, which are the output of technologies developed in the last decade.

The basic requirement before integration of data sources is to measure the gene similarities on a common scoring framework. In this regard, at first, the similarities between genes arising from different data sources are measured in the common scoring framework of 'Yeast GO-Slim: Process' annotations. These are then integrated in a linear combination style through a set of weights that are adaptively computed. The scoring framework reflects the accuracy of similarity values, but does not provide any information about importance/weight of one data-source in presence of the other data-sources in predicting gene-pairs. Consequently, assignment of appropriate weights to data sources before integration is necessary and natural. The new scoring framework, *K-BS*, for predicting the function of some of the unclassified Yeast genes, not only takes care of the above mentioned issues but also predicts the functional categories of 417 classified genes from 417 clusters with 98.20% accuracy. Superiority of *K-BS* in predicting true positive gene pairs gene-pairs, as compared with Lee et al.'s [62] YeastFinalNet in terms of positive predictive value (PPV), is also established in Sections 5.3.1 and 5.3.4.

Since it has been found in Section 5.3.2 that even a small proportion of annotated genes can provide improvements in true positive gene pairs for *K-BS*, it may also be successfully used for organisms where the number of classified genes is as low as 20%. If the ideal of rapid experimental characterization of all gene products in sequenced genomes is to be achieved, future research efforts must be directed toward genes not yet characterized. High-confidence predictions of gene functions will be invaluable in suggesting such directed experimentation. *K-BS* offers a flexible, general framework in which to predict and validate functional predictions for genes and proteins. The system can be upgraded in a modular fashion to incorporate improvements in each data source or in annotation quality or coverage, and new developments in annotation or prediction algorithms. The approach can be used to predict gene function in any organism. Now human data and annotations are widely available and we believe that our approach will be valuable in directing the experimental validation of newly proposed human gene functions. In addition, our approach can use as input groupings of genes derived from other data sources. For example, gene-fusion and gene-order conservation based methods could be incorporated directly into our methodology.

The most convincing demonstration of the efficacy of a predictive method is to confirm specific predictions by de novo experimentation. Some predicted gene functions, shown in Table 5.1, are confirmed or supported by the related literature, while the other functions are yet to be confirmed. An important application of *K-BS* will be in directing new research toward functional classes that appear to be under-studied.

Even though the current approaches in biocomputing are very helpful in identifying patterns and functions of proteins and genes, the output results are still not perfect. The methods are not only time-consuming, requiring Unix workstations to run on, but might also lead to false interpretations and assumptions due to necessary simplifications. It is therefore still mandatory to use biological reasoning and common sense in evaluating the results delivered by a biocomputing program. Also, for evaluation of the trustworthiness of the output of a program it is necessary to understand its mathematical and theoretical background to finally come up with a use- and senseful analysis.

Other potential bioinformatics tasks, where methods developed in this thesis can be used include

- characterization of protein content and metabolic pathways,

- identification of interacting proteins,
- assignment of gene products, and
- mapping expression data to sequence, structural and biochemical data.

Author's Publications

International Journal Papers

1. S. K. Pal, S. Bandyopadhyay, and S. S. Ray. Evolutionary Computation in Bioinformatics: A Review. *IEEE Transactions on Systems, Man, and Cybernetics, Part-C*, 36(5):601–615, 2006.

2. S. S. Ray, S. Bandyopadhyay, P. Mitra, and S. K. Pal. Bioinformatics in Neurocomputing Framework. *IEE Proc. Circuits Devices & Systems*, 152:556–564, 2005.

3. S. S. Ray, S. Bandyopadhyay, and S. K. Pal. Genetic Operators for Combinatorial Optimization in TSP and Microarray Gene Ordering. *Applied Intelligence*, 26(3):1830–195, 2007.

4. S. S. Ray, S. Bandyopadhyay, and S. K. Pal. Gene Ordering in Partitive Clustering using Microarray Expressions. *Journal of Biosciences*, 32(5):1019–1025, 2007.

5. S. S. Ray, S. Bandyopadhyay, and S. K. Pal. Dynamic Range Based Distance Measure for Microarray Expressions and a Fast Gene Ordering Algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part-B*, 37(3):742–749, 2007.

6. S. S. Ray, S. Bandyopadhyay, and S. K. Pal. Combining Multi-Source Information through Functional Annotation Based Weighting: Gene Function Prediction in Yeast. *IEEE Transactions on Biomedical Engineering*, 2008 (under revision).

International Conference Papers

1. S. S. Ray, S. Bandyopadhyay, P. Mitra, and S. K. Pal. Bioinformatics in Neurocomputing Framework. *The International Conference on Computers and Devices for Communication, CODEC-04*, page 94, January 1-3, Kolkata, India, 2004.

2. S. S. Ray, S. Bandyopadhyay, and S. K. Pal. New Operators of Genetic Algorithms for Traveling Salesman Problem. *The 17th International Conference on Pattern Recognition, ICPR-04* volume 2, pages 497–500, Cambridge, UK, 23-26 August 2004.

3. S. S. Ray, S. Bandyopadhyay, and S. K. Pal. New Genetic Operators for Solving TSP: Application to Microarray Gene Ordering. *The First International Conference on Pattern Recognition and Machine Intelligence, PReMI 2005*, pages 617-622, December, Kolkata, India, 2005.

4. S. S. Ray, S. Bandyopadhyay, and S. K. Pal. Gene Ordering in Partitive Clustering using Microarray Expressions. *International Conference on Bioinformatics, INCOB 2006*, page 33, 18-20 December, New Delhi, 2006.

5. S. S. Ray, S. Bandyopadhyay, and S. K. Pal. New Distance Measure for Microarray Gene Expressions using Linear Dynamic Range of Photo Multiplier Tube. *Int. Conf. on Computing: Theory and Applications, Kolkata, India*, pages 337–341, 2007.

6. S. S. Ray, S. Bandyopadhyay, and S. K. Pal. Predicting Gene Function in Yeast through Adaptive Weighting of Multi-Source Information. *The Eighth International Conference on Systems Biology, ICSB 2007*, online proceedings, no. H03, October 1-6, Long Beach, California, USA, 2007.

Bibliography

[1] T. Akutsu, S. Miyano, and S. Kuhara. Identification of Genetic Networks from a Small Number of Gene Expression Patterns under the Boolean Network Model. *Proc. Pacific Symposium on Biocomputing*, 99:17–28, 1999.

[2] A. A. Alizadeh and M. B. Eisen et al. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature*, 403(6769):503–511, 2000.

[3] R. B. Altman, A. Valencia, S. Miyano, and S. Ranganathan. Challenges for intelligent systems in biology. *IEEE Intelligent Systems*, 16(6):14–20, 2001.

[4] S. F. Altschul, T. L. Madden, A. A. Schffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.

[5] D. Applegate, R. Bixby, V. Chvtal, and William Cook. Concorde package. [online]. *www.tsp.gatech.edu/concorde/downloads/codes/src/ co031219.tgz*, 2003.

[6] D. Applegate, W. Cook, and A. Rohe. Chained Lin-Kernighan for large traveling salesman problems. Technical report, Dept. Comput. Appl. Math., Rice Univ., July 2000.

[7] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nat Genet.*, 25(1):25–29, 2000.

[8] T. H. B, B. Dysvik, and I. Jonassen. Lsimpute: accurate estimation of missing values in microarray data with least squares methods. *Nucleic Acids Research*, 32(3: e34):online, 2004.

[9] Y. Bai, W. Zhang, and Z. Jin. An new self-organizing maps strategy for solving the traveling salesman problem. *Chaos, Solitons & Fractals*, 28(4):1082–1089, 2006.

[10] A. Bairoch, R. Apweiler, C. H. Wu, W. C. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, M. J. Martin, D. A. Natale, C. O'Donovan, N. Redaschi, and L. S. Yeh. The universal protein resource (uniprot). *Nucleic Acids Research*, 33(Database issue):D154–159, 2005.

[11] R. Balasubramaniyan, E.Hullermeier, N. Weskamp, and J. Kamper. Clustering of gene expression data using a local shape-based similarity measure. *Bioinformatics*, 21(7):10691077, 2005.

[12] P. Baldi and S. Brunak. *Bioinformatics: The Machine Learning Approach*. MIT Press, Cambridge, MA, 1998.

[13] S. Bandyopadhyay and S. K. Pal. *Classification and Learning Using Genetic Algorithms: Applications in Bioinformatics and Web Intelligence*. Springer-Verlag, Hiedelberg, Germany, 2007.

[14] Z. Bar-Joseph, D. K. Gifford, and T. S. Jaakkola. Fast optimal leaf ordering for hierarchical clustering. *Bioinformatics*, 17:2229, 2001.

[15] Y. Barash and R. Friedman. Context-specific Bayesian clustering for gene expression data. *J. Computational Biology*, 9:169191, 2002.

[16] A. Ben-Dor, R. Shamir, and Z. Yakhin. Clustering gene expression patterns. *J. Computational Biology*, 6:281297, 1999.

[17] J. L. Bentley. Fast Algorithms for Geometric Traveling Salesman Problems. *ORSA Journal on Computing*, 4(4):387–411, 1992.

[18] T. Biedl, B. Brejov, E. D. Demaine, A. M. Hamel, and T. Vinar. Optimal arrangement of leaves in the tree representing hierarchical clustering of gene expression data. Technical Report 2001-14, Dept. Computer Sci., Univ. Waterloo, 2001.

[19] S. Brenner, F. Jacob, and M. Meselson. An unstable intermediate carrying information from genes to ribosomes for protein synthesis. *Nature*, 190:576–581, 1961.

[20] J. A. Brown, G. Sherlock, C. L. Myers, N. M. Burrows, C. Deng, H. I. Wu, K. E. McCann, O. G. Troyanskaya, and J. M. Brown. Global analysis of gene function in yeast by quantitative phenotypic profiling. *Molecular System Biology*, 2(2006.0001):1–9, 2006.

[21] Michael P. S. Brown, W. N. Grundy, D. Lin, N. Cristianini, C. W. Sugnet, T. S. Furey, M. A. Jr., and D. Haussler. Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proc Natl Acad Sci U S A*, 97(1):262–267, 2000.

[22] R. J. Cho, M. J. Campbell, E. A. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, T. G. Wolfsberg, A. E. Gabrielian, D. Landsman, D. J. Lockhart, and R. W. Davis. A genome-wide transcriptional analysis of the mitotic cell cycle. *Molecular Cell*, 2(1):65–73, 1998.

[23] P. Chou and G. Fasmann. Prediction of the secondary structure of proteins from their amino acid sequence. *Advances in Enzymology*, 47:145–148, 1978.

[24] G. A. Churchill. Using anova to analyze microarray data. *Biotechniques*, 37(2):173–175, 2004.

[25] W. S. Cleveland and S. J. Devlin. Locally weighted regression: An approach to regression analysis by local fitting. *Journal of the American Statistical Association*, 83:596–610, 1988.

[26] C. Cotta, A. Mendes, V. Garcia, P. Franca, and P. Moscato. Applying memetic algorithms to the analysis of microarray data. *Evo Workshops*, pages 22–32, 2003.

[27] J. S. d. Sousa, L. d. C. T. Gomes, G. B. Bezerra, L. N. d. Castro, and F. J. V. Zuben. An immune-evolutionary algorithm for multiple rearrangements of gene expression data. *Genetic Programming and Evolvable Machines*, 5(2):157–179, 2004.

[28] L. Davis. Applying adapting algorithms to epistatic domains. *Proc. Int. Joint Conf. Artificial Intelligence*, Quebec, canada, 1985.

[29] P. Dhaeseleer, S. Liang, and R. Somogyi. Genetic network inference: From co-expression clustering to reverse engineering. *Bioinformatics*, 16:707726, 2000.

[30] S. S. Dwight, M. A. Harris, K. Dolinski, C. A. Ball, G. Binkley, K. R. Christie, D. G. Fisk, L. Issel-Tarver, M. Schroeder, G. Sherlock, A. Sethuraman, S. Weng, D. Botstein, and J. M. Cherry. Saccharomyces genome database (sgd) provides secondary gene annotation using the gene ontology (go). *Nucleic Acids Research*, 30(1):69–72, 2002.

[31] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc. National Academy of Sciences*, 95:14863–14867, 1998.

[32] G. Sherlock et al. The stanford microarray database. *Nucleic Acids Research*, 29(1):152–155, 2001.

[33] L. Shi et. al. Microarray scanner calibration curves: characteristics and implications. *BMC Bioinformatics*, 6((Suppl2):S11):1–14, 2005.

[34] V. R. Iyer et al. The transcriptional program in the response of human fibroblasts to serum. *Science*, 283(5398):83–87, 1999.

[35] W. C. Barker et al. The protein information resource (pir). *Nucleic Acids Research*, 28(1):41–44, 2000.

[36] C. N. Fiechter. A parallel tabu search algorithm for large traveling salesman problems. *Discrete Appl. Math. Combin. Oper. Res. Comput. Sci.*, 51:243–267, 1994.

[37] V. Filkov, S. Skiena, and J. Zhi. Analysis techniques for microarray time-series data. *J. Comput. Biol.*, 9:317330, 2002.

[38] Munich Information for Protein Sequences. http://www.mips.com.

[39] N. Gale, W. C. Halperin, and C. Costanzo. Unclassed matrix shading and optimal ordering in hierarchical cluster analysis. *J. Classif.*, 1:7592, 1984.

[40] D. Gamboa, C. Rego, and F. Glover. Implementation analysis of efficient heuristic algorithms for the traveling salesman problem. *Computers & Operations Research*, 33(4):1154–1172, 2006.

[41] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman and Co., San Francisco, 1979.

[42] A. P. Gasch and M. B. Eisen. Exploring the conditional coregulation of yeast gene expression through fuzzy k-means clustering. *Genome Biology*, 3(11):research0059.1–0059.22, 2002.

[43] M. Gerstein and R. Jansen. The current excitement in bioinformatics-analysis of whole-genome expression data: how does it relate to protein structure and function? *Curr. Opin. Struct. Biol.*, 10:574584, 2000.

[44] D. Gillespie and S. Spiegelman. A quantitative assay for dna-rna hybrids with dna immobilized on a membrane. *J. Molecular Biology*, 12(3):829–842, 1965.

[45] D. E. Goldberg. *Genetic Algorithm in Search, Optimization and Machine Learning*. Machine Learning, Addison-Wesley, New York, 1989.

[46] W. S. Gossett. The probable error of a mean. *Biometrika*, 6:1–25, 1908.

[47] G. Gruvaeus and H. Wainer. Two additions to hierarchical cluster analysis. *British J. Math. Statist. Psychol.*, 25:200206, 1972.

[48] K. Helsgaun. An effective implementation of the Lin-Kernighan traveling salesman heuristic. *European Journal of Operational Research*, 1:106–130, 2000.

[49] R. Herwig, A. J. Poustka, C. Muller, C. Bull, H. Lehrach, and J. OBrien. Large-scale clustering of cDNA-fingerprinting data. *Genome Res.*, 9:10931105, 1999.

[50] A. Homaifar, S. Guan, and G. Liepins. A new approach on the traveling salesman problem by genetic algorithms. *5th Int. Conf. Genetic Algorithms*, pages 460–466, 1993.

[51] R. G. Jenner, M. M. Alb, C. Boshoff, and P. Kellam. Kaposi's sarcoma-associated herpesvirus latent and lytic gene expression as revealed by dna arrays. *Journal of Virology*, 75(2):891–902, 2001.

[52] L. Jiao and L. Wang. A novel genetic algorithm based on immunity. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 30(5):552–561, 2000.

[53] D. S. Johnson and L. A. McGeoch. The Traveling Salesman Problem: A Case Study in Local optimization. *Local Search in Combinatorial Optimization, Wiley and Sons, New York*, 1996.

[54] S. C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 2:241–254, 1967.

[55] M. Kanehisa, S. Goto, M. Hattori, K. F. Aoki-Kinoshita, M. Itoh, S. Kawashima, T. Katayama, M. Araki, and M. Hirakawa. From genomics to chemical genomics: new developments in kegg. *Nucleic Acids Res.*, 34:D354–D357, 2006.

[56] S. Kawasaki, C. Borchert, M. Deyholos, H.Wang, S. Brazille, K. Kawai, D. Galbraith, and H. J. Bohnert. Gene expression profiles during the initial phase of salt stress in rice. *Plant Cell*, 13(4):889906, 2001.

[57] M. K. Kerr, M. Martin, and G. A. Churchill. Analysis of variance for gene expression microarray data. *Journal of Computational Biology*, 7(6):819–837, 2000.

[58] J. Khan, J. S. Wei, M. Ringnr, L. H. Saal, M. Ladanyi, F. Westermann, F. Berthold, M. Schwab, C. R. Antonescu, C. Peterson, and P. S. Meltzer. Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature Medicine*, 7(6):673–679, 2001.

[59] T. Kohonen. The self-organizing map. *Proc. IEEE*, 78(9):1464–1480, 1990.

[60] E. F. Krause. Taxicab Geometry: An Adventure in Non-Euclidean Geometry. 1986.

[61] P. Larranaga, C. Kuijpers, R. Murga, I. Inza, and S. Dizdarevic. Genetic algorithms for the traveling salesman problem: A review of representations and operators. *Artificial Intell. Rev.*, 13:129–170, 1999.

[62] I. Lee, S. V. Date, A. T. Adai, and E. M. Marcotte. A probabilistic functional network of yeast genes. *Science*, 306:1555–1558, 2004.

[63] I. Lee, R. Narayanaswamy, and E. M. Marcotte. *Yeast Gene Analysis*, chapter : Bioinformatic prediction of yeast gene function. Elsevier Press, Amsterdam, 2006.

[64] S. K. Lee, Y. H. Kim, and B. R. Moon. Finding the Optimal Gene Order in Displaying Microarray Data. *GECCO*, pages 2215–2226, 2003.

[65] S. Lin and B. W. Kernighan. An effective heuristic for the traveling salesman problem. *Operation Research*, 21(2):498–516, 1973.

[66] D. J. Lockhart and E. A. Winzeler. Genomics, gene expression and dna arrays. *Nature*, 405(6788):827–836, 2000.

[67] J. A. Love. Introduction to microarray technology. *Whitehead Institute Center for Microarray Technology*, pages 1–23.

[68] C. Lu. Improving the scaling normalization for high-density oligonucleotide genechip expression microarrays. *BMC Bioinformatics*, 5(103), 2004.

[69] N. M. Luscombe, D. Greenbaum, and M. Gerstein. What is Bioinformatics? A Proposed Definition and Overview of the Field. *Yearbook of Medical Informatics*, pages 83–100, 2001.

[70] Q. Ma, G. W. Chirn, R. Cai, J. D. Szustakowski, and N. Nirmala. Clustering protein sequences with a novel metric transformed from sequence similarity scores and sequence alignments with neural networks. *BMC Bioinformatics*, 6(242), 2005.

[71] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *In Proc. of the 5th Berkeley Symposium on Mathematical Statistics and Probability (ed. L. M. LeCam and J. Neyman)*, volume 1, pages 281–297, University of California Press, Los Angeles, CA, 1967.

[72] E. M. Marcotte, M. Pellegrini, H. L. Ng, D. W. Rice, T. O. Yeates, and D. Eisenberg. Detecting protein function and protein-protein interactions from genome sequences. *Science*, 285:751–753, 1999.

[73] E. M. Marcotte, M. Pellegrini, M. J. Thompson, T. O. Yeates, and D. Eisenberg. A combined algorithm for genome-wide prediction of protein function. *Nature*, 402:83–86, 1999.

[74] C. V. Mering, R. Krause, B. Snel, M. Cornell, S. G. Oliver, S. Fields, and P. Bork. Comparative assessment of large-scale data sets of protein-protein interactions. *Nature*, 417:399–403, 2002.

[75] B. J. T. Morgan and A. P. G. Ray. Non-uniqueness and inversions in cluster analysis. *Applied Statistics*, 44(1):117–134, 1995.

[76] I. Oliver, D. Smith, and J. Holland. A study of permutation crossover operators on the traveling salesman problem. *Second Int. Conf. Genetic Algorithms*, pages 224–230, 1987.

[77] S. K. Pal, S. Bandyopadhyay, and S. S. Ray. Evolutionary computation in bioinformatics: A review. *IEEE Transactions on Systems, Man, and Cybernetics, Part-C*, 36(5):601–615, 2006.

[78] J. Park, K. Karplus, C. Barrett, R. Hughey, D. Haussler, T. Hubbard, and C. Chothia. Sequence comparisons using multiple sequences detect three times as many remote homologues as pairwise methods. *J Mol Biol*, 284:1201–1210, 1998.

[79] P. Pavlidis. Using anova for gene selection from microarray studies of the nervous system. *Methods*, 31(4):282–289, 2003.

[80] M. Pellegrini, E. M. Marcotte, M. J. Thompson, D. Eisenberg, and T. O. Yeates. Assigning protein functions by comparative genome analysis: protein phylogenetic profiles. *Proc. Natl. Acad. Sci. USA*, 96:4285–4288, 1999.

[81] P. Pipenbacher, A. Schliep, S. Schneckener, A. Schonhuth, D. Schomburg, and R. Schrader. Proclust: improved clustering of protein sequences with an extended graph-based approach. *Bioinformatics*, 18(2):S182S191, 2002.

[82] J. Y. Potvin. The traveling salesman problem: A neural network perspective. *ORSA J. Comput.*, 5:328–348, 1993.

[83] J. Qian, M. Dolled-Filhart, J. Lin, H. Yu, and M. Gerstein. Beyond synexpression relationships: local clustering of time-shifted and inverted gene expression profiles identifies new, biologically relevant interactions. *J. Mol. Biol.*, 314:10531066, 2001.

[84] J. Quackenbush. Microarray data normalization and transformation. *Nature Genetics*, 32:496–501, 2002.

[85] S. S. Ray, S. Bandyopadhyay, P. Mitra, and S. K. Pal. Bioinformatics in neurocomputing framework. *IEE Proc. Circuits Devices & Systems*, 152:556–564, 2005.

[86] S. S. Ray, S. Bandyopadhyay, and S. K. Pal. Gene ordering in partitive clustering using microarray expressions. *International Conference on Bioinformatics, INCOB 2006*, page 33, 18-20 December, New Delhi, 2006.

[87] S. S. Ray, S. Bandyopadhyay, and S. K. Pal. New operators of genetic algorithms for traveling salesman problem. volume 2, pages 497–500, Cambridge, UK, 23-26 August 2004. ICPR-04.

[88] S. S. Ray, S. Bandyopadhyay, and S. K. Pal. New genetic operators for solving tsp: Application to microarray gene ordering. In *First Int. Conf. Pattern Recognition and Machine Intelligence (accepted)*, Kolkata, India, December 2005. Indian Statistical Institute, Springer.

[89] S. S. Ray, S. Bandyopadhyay, and S. K. Pal. Dynamic range based distance measure for microarray expressions and a fast gene ordering algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part-B*, 37(3):742–749, 2007.

[90] S. S. Ray, S. Bandyopadhyay, and S. K. Pal. Gene ordering in partitive clustering using microarray expressions. *Journal of Biosciences*, 32(5):1019–1025, 2007.

[91] S. S. Ray, S. Bandyopadhyay, and S. K. Pal. Genetic operators for combinatorial optimization in tsp and microarray gene ordering. *Applied Intelligence*, 26(3):1830–195, 2007.

[92] S. S. Ray, S. Bandyopadhyay, and S. K. Pal. New distance measure for microarray gene expressions using linear dynamic range of photo multiplier tube. *Int. Conf. on Computing: Theory and Applications, Kolkata, India*, pages 337–341, 2007.

[93] S. S. Ray, S. Bandyopadhyay, and S. K. Pal. Combining multi-source information through functional annotation based weighting: Gene function prediction in yeast. *IEEE Transactions on Biomedical Engineering*, pages 1–7, 2008 (under revision).

[94] S. S. Ray, S. Bandyopadhyay, and S. K. Pal. Predicting gene function in yeast through adaptive weighting of multi-source information. *The Eighth International Conference on Systems Biology, ICSB 2007*, (H03):63, October 1-6, Long Beach, California, USA, 2007.

[95] T. Reguly, A. Breitkreutz, L. Boucher, B. J. Breitkreutz, G. C. Hon, C. L. Myers, A. Parsons, H. Friesen, R. Oughtred, A. Tong, C. Stark, Y. Ho, D. Botstein, B. Andrews, C. Boone, O. G. Troyanskya, T. Ideker, K. Dolinski, N. N. Batada, and M. Tyers. Comprehensive curation and analysis of global interaction networks in saccharomyces cerevisiae. *Journal of Biology*, 5(4):1–28, 2006.

[96] G. Reinelt. The Traveling Salesman: Computational Solutions for TSP Applications. *Lecture Notes in Computer Science, Springer-Verlag*, 840, 1994.

[97] L Salwinski, C. S. Miller, A. J. Smith, F. K. Pettit J. U. Bowie, and D. Eisenberg. The database of interacting proteins. *Neuclic Acid Research*, 32:449451, 2004.

[98] T. Sawa and L. Ohno-Machado. A neural network-based similarity index for clustering DNA microarray data. *Comput. Biol. Med.*, 33:115, 2003.

[99] J. setubal and J. Meidanis. *Introduction to Computational Molecular Biology.* International Thomson Publishing, 20 park plaza, Boston, MA 02116, 1999.

[100] R. Sharan, A. Maron-Katz, and R. Shamir. CLICK and EXPANDER: a system for clustering and visualizing gene expression data. *Bioinformatics*, 19(14):1787–1799, 2003.

[101] R. Sharan and R. Shamir. CLICK: A clustering algorithm with applications to gene expression analysis. *Int. Conf. Intelligent Systems for Molecular Biology*, page 307316, 2000.

[102] A. Shiratori, T. Shibata, M. Arisawa, F. Hanaoka, Y. Murakami, and T. Eki. Systematic identification, classification, and characterization of the open reading frames which encode novel helicase-related proteins in saccharomyces cerevisiae by gene disruption and northern analysis. *Yeast*, 15(3):219–253, 1999.

[103] D. K. Slonim. From patterns to pathways: gene expression data analysis comes of age. *Nat. Genet. Suppl.*, 32:502508, 2002.

[104] E. M. Southern. Detection of specific sequences among dna fragments separated by gel electrophoresis. *J. Molecular Biology*, 98(3):503–517, 1975.

[105] P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisia by microarray hybridization. *Molecular Biology Cell*, 9:3273–3297, 1998.

[106] V. Spirin and L. A. Mirny. Protein complexes and functional modules in molecular networks. *Proc. Natl Acad. Sci. USA*, 100(21):1212312128, 2003.

[107] T. Starkweather, S. McDaniel, K. Mathias, D. Whitley, and C. Whitley. A comparison of genetic sequencing operators. *4th Int. Conf. Genetic Algorithms*, pages 69–76, 1991.

[108] T. Stutzle and M. Dorigo. *ACO algorithms for the traveling salesman problem, Evolutionary Algorithms in Engineering and Computer Science.* John Wiley and Sons, 1999.

[109] P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E. S. Lander, and T. R. Golub. Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation. *Proc. National Academy of Sciences*, pages 2907–2912, 1999.

[110] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman. Missing value estimation methods for dna microarrays. *Bioinformatics*, 17(6):520–525, 2001.

[111] O. G. Troyanskaya, K. Dolinski, A. B. Owen, R. B. Altman, and D. Botstein. A bayesian framework for combining heterogeneous data sources for

gene function prediction (in saccharomyces cerevisiae). *Proc. Natl. Acad. Sci. USA*, 100(14):8348–8353, 2003.

[112] C. F. Tsai, C. W. Tsai, and T. Yang. A modified multiple-searching method to genetic algorithms for solving traveling salesman problem. *IEEE Int. Conf. Systems, Man and Cybernetics*, 3:6–9, 2002.

[113] H. K. Tsai, J. M. Yang, and C. Y. Kao. Applying Genetic Algorithms To Finding The Optimal Gene Order In Displaying The Microarray Data. *GECCO*, pages 610–617, 2002.

[114] H. K. Tsai, J. M. Yang, Y. F. Tsai, and C. Y. Kao. An Evolutionary Algorithm for Large Traveling Salesman Problems. *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cyebernetics*, 34(4):1718–1729, 2004.

[115] H. K. Tsai, J. M. Yang, Y. F. Tsai, and C. Y. Kao. An Evolutionary Approach for Gene Expression Patterns. *IEEE Trans. on Info. Tech. in Biomedicine*, 8(2):69–78, 2004.

[116] TSPLIB. http://www.iwr.uniheidelberg.de/groups/comopt/software/TSPLIB95/.

[117] J. Tuikkala, L. Elo, O. S. Nevalainen, and T. Aittokallio. Improving missing value estimation in microarray data with gene ontology. *Bioinformatics*, 22(5):566–572, 2006.

[118] D. Venet. MatArray: a Matlab toolbox for microarray data. *Bioinformatics*, 19(5):659–660, 2003.

[119] Website. http://rana.lbl.gov/EisenData.htm.

[120] Website. http://www.psrg.lcs.mit.edu/clustering/ismb01/optimal.html.

[121] D. Wettschereck, D. W. Aha, and T. Mohori. A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, 11(1-5):273–314, 1997.

[122] D. Whitley, T. Starkweather, and D. Fuquay. Scheduling problems and traveling salesman: The genetic edge recombination operator. *3rd Int. Conf. Genetic Algorithms*, pages 133–140, 1989.

[123] A. S. Wu and R. K. Lindsay. A Survey of Intron Research in Genetics. *In Proc. 4th Conf. of on Parallel Problem Solving from Nature*, pages 101–110, 1996.

[124] W. Wu, E. P. Xing, C. Myers, I. S. Mian, and M. J. Bissell. Evaluation of normalization methods for cdna microarray data by k-nn classification. *BMC Bioinformatics*, 6(191):1–21, 2005.

[125] H. Xie, A. Wasserman, Z. Levine, A. Novik, V. Grebinskiy, Avi Shoshan, and Liat Mintz. Large-scale protein annotation through gene ontology. *Genome Research*, 12:785–794, 2002.

[126] DeLisi C Yanai I. The society of genes: networks of functional links between genes from comparative genomics. *Genome Biology*, 3(11):1–64, 2002.

[127] Y. H. Yang, S. Dudoit, P. Luu, and T. P. Speed. Normalization of cdna microarray data. *In M. L. Bittner, Y. Chen, A. N. Dorsel, and E. R. Dougherty (eds.), Microarrays: Optical Technologies and Informatics, Proc. of SPIE*, 4266:141–152, 2001.

[128] M. Zachariasen and M. Dam. Tabu search on the geometric traveling salesman problem. *Proc. of Int. Conf. on Metaheuristics*, pages 571–587, 1995.

Annexure (Publications)

Evolutionary Computation in Bioinformatics: A Review

Sankar K. Pal, *Fellow, IEEE*, Sanghamitra Bandyopadhyay, *Senior Member, IEEE*, and Shubhra Sankar Ray

***Abstract*—This paper provides an overview of the application of evolutionary algorithms in certain bioinformatics tasks. Different tasks such as gene sequence analysis, gene mapping, deoxyribonucleic acid (DNA) fragment assembly, gene finding, microarray analysis, gene regulatory network analysis, phylogenetic trees, structure prediction and analysis of DNA, ribonucleic acid and protein, and molecular docking with ligand design are, first of all, described along with their basic features. The relevance of using evolutionary algorithms to these problems is then mentioned. These are followed by different approaches, along with their merits, for addressing some of the aforesaid tasks. Finally, some limitations of the current research activity are provided. An extensive bibliography is included.**

***Index Terms*—Biocomputing, data mining, evolutionary algorithm, molecular biology, soft computing.**

I. Introduction

OVER the past few decades, major advances in the field of molecular biology, coupled with advances in genomic technologies, have led to an explosive growth in the biological information generated by the scientific community. This deluge of genomic information has, in turn, led to an absolute requirement for computerized databases to store, organize, and index the data, and for specialized tools to view and analyze the data.

Bioinformatics can be viewed as *the use of computational methods to make biological discoveries* [1]. It is an interdisciplinary field involving biology, computer science, mathematics, and statistics to analyze biological sequence data, genome content and arrangement, and to predict the function and structure of macromolecules. The ultimate goal of the field is to enable the discovery of new biological insights as well as to create a global perspective from which unifying principles in biology can be derived [2]. There are three important subdisciplines within bioinformatics.

1) Development of new algorithms and models to assess different relationships among the members of a large biological data set in a way that allows researchers to access existing information, and to submit new information as they are produced.
2) Analysis and interpretation of various types of data including nucleotide and amino acid sequences, protein domains; and protein structures.
3) Development and implementation of tools that enable efficient access and management of different types of information.

Recently, evolutionary algorithms (EAs), a class of randomized search and optimization techniques guided by the principles of evolution and natural genetics, have been gaining the attention of researchers for solving bioinformatics problems. Genetic algorithms (GAs) [3]–[9] evolutionary strategies (ES), and genetic programming (GP) are the major components of EAs. Of these, GAs are the most widely used. GAs are efficient, adaptive, and robust search processes, producing near optimal solutions, and have a large amount of implicit parallelism. Data analysis tools used earlier in bioinformatics were mainly based on statistical techniques such as regression and estimation. The role of GAs in bioinformatics gained significance with the need to handle large data sets in biology in a robust and computationally efficient manner.

This paper provides a survey of the various evolutionary-algorithm-based techniques that have been developed over the past few years for different bioinformatics tasks. First, we describe the basic concepts of bioinformatics along with their biological basis. Methodology for applying GAs to bioinformatics tasks is also mentioned in Section II. In Section III, various bioinformatics tasks and different evolutionary algorithms based methods available to address the bioinformatics tasks are explained. Finally, conclusions and some future research directions are presented in Section IV.

II. Basic Concepts in Bioinformatics and Relevance of Evolutionary Algorithms

First, we introduce the basic biological concepts required to understand the various problems in bioinformatics, and then we describe the relevance of EAs in bioinformatics with particular emphasis on their application of GAs.

A. Basic Units of Cell Biology and Bioinformatics Tasks

Deoxyribonucleic acid (DNA) and proteins are biological macromolecules built as long linear chains of chemical components. A DNA strand consists of a large sequence of nucleotides, or bases. For example there are more than three billion bases in human DNA sequences. DNA plays a fundamental role in different biochemical processes of living organisms in two respects. First, it contains the templates for the synthesis of proteins, which are essential molecules for any organism [10]. The second role in which DNA is essential to life is as a medium to transmit hereditary information (namely, the building plans for

Manuscript received May 7, 2004; revised January 7, 2005. This paper was supported by the Council of Scientific and Industrial Research, New Delhi, India, under the project "Knowledge Based Connectionist Data Mining System: Design and Application" under Grant 22(0346)/02/EMR-II. This paper was recommended by Associate Editor M. Last.

The authors are with the Machine Intelligence Unit, Indian Statistical Institute, Kolkata 700108, India (e-mail: sankar@isical.ac.in; sanghami@isical.ac.in; shubhra_r@isical.ac.in).

Digital Object Identifier 10.1109/TSMCC.2005.855515

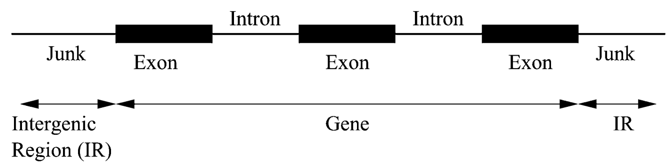


Fig. 1. Various parts of DNA.

proteins) from generation to generation. Proteins are responsible for structural behavior.

The units of DNA are called nucleotides. One nucleotide consists of one nitrogen base, one sugar molecule (deoxyribose), and one phosphate. Four nitrogen bases are denoted by one of the letters A (adenine), C (cytosine), G (guanine), and T (thymine). A linear chain of DNA is paired to a complementary strand. The complementary property stems from the ability of the nucleotides to establish specific pairs (A-T and G-C). The pair of complementary strands then forms the double helix that was first suggested by Watson and Crick in 1953. Each strand, therefore, carries all the information, and the biochemical machinery guarantees that the information can be copied over and over again, even when the "original" molecule has long since vanished.

A gene is primarily made up of a sequence of triplets of the nucleotides (exons). Introns (noncoding sequence) may also be present within the gene. Not all portions of the DNA sequences are coding. A coding zone indicates that it is a template for a protein. As an example, for the human genome, only 3%–5% of the sequence are coding; i.e., they constitute the gene. The promoter is a region before each gene in the DNA that serves as an indication to the cellular mechanism that a gene is ahead. For example, the codon AUG is a protein which codes for methionine and signals the start of a gene. Promoters are key regulatory sequences that are necessary for the initiation of transcription. Transcription is process in which ribonucleic acid (RNA) is formed from a gene, and through translation, aminoacids are formed from RNA. There are sequences of nucleotides within the DNA that are spliced out progressively in the process of transcription and translation. A comprehensive survey of the research done in this field is given in [11]. In brief, the DNA consists of three types of noncoding sequences (see Fig. 1) as follows:

1) Intergenic regions: Regions between genes that are ignored during the process of transcription.
2) Intragenic regions (or Introns): Regions within the genes that are spliced out from the transcribed RNA to yield the building blocks of the genes, referred to as Exons.
3) Pseudogenes: Genes that are transcribed into the RNA and stay there, without being translated, due to the action of a nucleotide sequence.

Proteins are polypeptides, formed within cells as a linear chain of amino acids [10]. Amino acid molecules bond with each other by eleminating water molecules and forming peptides. 20 different amino acids (or "residues") are available, which are denoted by 20 different letters of the alphabet. Each of the 20 amino acids is coded by one or more triplets (or codons) of the nucleotides making up the DNA. Based on the genetic code, the linear string of DNA is translated into a linear string of amino acids; i.e., a protein via mRNA (messenger RNA) [10]. For example, the DNA sequence GAACTACACACGTGTAAC codes for the amino acid sequence ELHTCN (shown in Fig. 2).

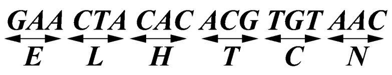


Fig. 2. Coding of amino acid sequence from DNA sequence.

Three-dimensional (3-D) molecular structure is one of the foundations of structure-based drug design. Often, data are available for the shape of a protein and a drug separately, but not for the two together. Docking is the process by which two molecules fit together in 3-D space. Ligands are small molecules such as a candidate drug and are used for docking to their macromolecular targets (usually proteins, sometimes DNA).

Different biological problems considered within the scope of bioinformatics involve the study of genes, proteins, nucleic acid structure prediction, and molecular design with docking. A broad classification of the various bioinformatics tasks is given as follows.

1) alignment and comparison of DNA, RNA, and protein sequences;
2) gene mapping on chromosomes;
3) gene finding and promoter identification from DNA sequences;
4) interpretation of gene expression and microarray data;
5) gene regulatory network identification;
6) construction of phylogenetic trees for studying evolutionary relationship;
7) DNA structure prediction;
8) RNA structure prediction;
9) protein structure prediction and classification;
10) molecular design and molecular docking.

Descriptions of these tasks and their implementation in evolutionary computing (or genetic algorithmic) framework are provided in Section III. Before that, the relevance of GAs in bioinformatics is explained.

B. Relevance of Genetic Algorithms in Bioinformatics

Genetic algorithms [3]–[6], a biologically inspired technology, are randomized search and optimization techniques guided by the principles of evolution and natural genetics. They are efficient adaptive, and robust search processes, producing near optimal solutions, and have a large degree of implicit parallelism. Therefore, the application of GAs for solving certain problems of bioinformatics, which need optimization of computation requirements, and robust, fast and close approximate solutions, appears to be appropriate and natural [4]. Moreover, the errors generated in experiments with bioinformatics data can be handled with the robust characteristics of GAs. To some extent, such errors may be regarded as contributing to genetic diversity, a desirable property. The problem of integrating GAs and bioinformatics constitutes a new research area.

GAs are executed iteratively on a set of coded solutions, called population, with three basic operators: selection/reproduction, crossover, and mutation. They use only the payoff (objective function) information and probabilistic transition rules for moving to the next iteration. They are different from most of the normal optimization and search procedures in four ways:

1) GAs work with the coding of the parameter set, not with the parameters themselves.
2) GAs work simultaneously with multiple points, and not a single point.
3) GAs search via sampling (a blind search) using only the payoff information.
4) GAs search using stochastic operators, not deterministic rules.

A GA typically consists of the following components:

1) a population of binary strings or coded possible solutions (biol)ogically referred to as chromosomes);
2) a mechanism to encode a possible solution (mostly as a binary string);
3) objective function and associated fitness evaluation techniques;
4) selection/reproduction procedure;
5) genetic operators (crossover and mutation);
6) probabilities to perform genetic operations.

Of all the evolutionarily inspired approaches, GAs seem particularly suited to implementation using DNA, protein, and other bioinformatics tasks [12]. This is because GAs are generally based on manipulating populations of bitstrings using both crossover and pointwise mutation.

The main advantages using GAs are as follows.

1) Several tasks in bioinformatics involve optimization of different criteria (such as energy, alignment score, and overlap strength), thereby making the application of GAs more natural and appropriate.
2) Problems of bioinformatics seldom need the exact optimum solution; rather, they require robust, fast, and close approximate solutions, which GAs are known to provide efficiently.
3) GAs can process, in parallel, populations billions times larger than is usual for conventional computation. The usual expectation is that larger populations can sustain larger ranges of genetic variation, and thus can generate high-fitness individuals in fewer generations.
4) Laboratory operations on DNA inherently involve errors. These are more tolerable in executing evolutionary algorithms than in executing deterministic algorithms. (To some extent, errors may be regarded as contributing to genetic diversity—a desirable property.)

C. *Example*

Let us now discuss with an example the relevance of GAs in bioinformatics. Most of the ordering problems in bioinformatics, such as sequence alignment problem, fragment assembly problem (FAP), and gene maping (GM), are quite similar to traveling salesman problem (TSP best-known NP-hard ordering problem) with notable differences. The TSP can be formally defined as follows: Let $1, 2, \ldots, n$ be the labels of the n cities and $C = [c_{i,j}]$ be an $n \times n$ cost matrix where $c_{i,j}$ denotes the cost of traveling from city i to city j. The TSP is the problem of finding the shortest closed route among n cities, having as input the complete distance matrix among all cities. A symmetric TSP (STSP) instance is any instance of the TSP such that $c_{i,j} = c_{j,i}$ for all cities i, j. An asymmetric TSP (ATSP) instance is any instance of the TSP that has at least one pair of cities such that $c_{i,j} \neq c_{j,i}$. The ATSP is a special case of the problem on which we restrict the input to asymmetric instances. The total cost A of a TSP tour is given by

$$A(n) = \sum_{i=1}^{n-1} c_{i,i+1} + c_{n,1}. \tag{1}$$

The objective is to find a permutation of the n cities which has minimum cost.

The FAP deals with the sequencing of DNA. Currently, strands of DNA longer than approximately 500 base pairs cannot routinely be sequenced accurately. Consequently, for sequencing larger strands of DNA, they are first broken into smaller pieces. In the shotgun sequencing method (to which this work applies), DNA is first replicated many times, and then individual strands of the double helix are broken randomly into smaller fragments. The assembly of DNA fragments into a consensus sequence corresponding to the parent sequence constitutes the "fragment assembly problem" [10]. It is a permutation problem, similar to the TSP, but with some important differences (circular tours, noise, and special relationships between entities) [10]. It is NP-complete in nature.

Note that the fragmenting process does not retain either the ordering of the fragments on the parent strand of DNA or the strand of the double helix from which a particular fragment came. The only information available in assembly stage is the base pair sequence for each fragment. Thus, the ordering of the fragments must rely primarily on the similarity of fragments and how they overlap. An important aspect of the general sequencing problem is the precise determination of the relationship and orientation of the fragment. Once the fragments have been ordered, the final consensus sequence is generated from the ordering. Basic steps with four fragments are shown below as an example in Fig. 3. Here, the fragments are aligned in a fashion so that in each column all the bases are the same. As an example, the base in the sixth column is selected, after voting, as G to make the consensus sequence TCACTGGCTTACTAAG.

```
  ACTGGC
    TGGCTTACT
TCACTC
        TTACTAAG
----------------
TCACTGGCTTACTAAG  ⟶ CONSENSUS
                     SEQUENCE
```

Fig. 3. Alignment of DNA fragments.

Formulation of the FAP as a TSP using GA: although the endpoints of the tour of TSP are irrelevant since its solution is a circular tour of the cities, in the case of FAP, the endpoints are relevant as they represent fragments on opposite ends of the parent sequence. Moreover, the cities in the TSP are not assumed to have any relationship other than the distances, and the ordering is the final solution to the problem. In FAP, the ordering referred to as "beads on a string," is only an intermediate step; the layout process uses the overlap data to position the bases within the fragments relative to each other. Here, GAs can be applied. A way of using it in FAP is explained as follows.

Step 1) Let $1, 2, \ldots, j, \ldots, n$ represent the indices of n fragments in the spectrum of fragments. Pairwise relationship (similarity) of a fragment with all other fragments (oligonucleotides) is calculated and kept in an $n \times n$ matrix. Dynamic programming gives best alignment between two sequences (fragments). In this method, each possible orientation is tried for the two fragments, and the overlap, orientation, and alignment are chosen to maximize the similarity between fragments.

Step 2) All the indices of fragments are then ordered randomly with no repetition. Let $f_1, f_2, \ldots, f_i, \ldots, f_n$ be such an ordering of a sequence of n fragments, where $f_i = j$ means that fragment j (in the fragment set) appears in position i of the ordering. The fitness function of this ordered sequence can be computed using

$$F = \sum_{i=1}^{n-1} W_{f_i, f_{i+1}} \tag{2}$$

where $W_{i,j}$ is the pairwise overlap strength (similarity) of fragments i and j in the ordered sequence, as obtained in the $n \times n$ matrix.

Such an ordered sequence provides a genetic representation of an individual chromosome in GA.

Step 3) In this way, P ordered sequences are generated, where P is the size of the population of GA.

Step 4) GA is applied with this population and the following operations.

Selection: Fitness of each sequence is evaluated as in (2), and sequences with higher fitness are selected with roulette wheel.

Crossover: Crossover is performed between two randomly selected sequences for a given crossover rate.

Mutation: For a given mutation rate, only that mutation operator can be applied for which there will be no repetition of fragment indexes in the sequence.

Elitist model: A new population is created at each generation of GA. The sequence with highest fitness from the previous generation replaces randomly a sequence from this new generation, provided the fitness of the fittest sequence in the previous generation is higher than the best fitness in this current generation.

Step 5) The best sequence of indices with maximum F value is obtained from the GA. From this sequence of indices, the corresponding sequence of fragments is obtained using the overlapping information in the $n \times n$ matrix of Step 1).

Step 6) This alignment of fragments is examined to determine the places where insertion or deletion error likely occurred, and gaps or bases are then inserted or deleted into the fragments to obtain their best possible alignment. The resulting sequence is called consensus sequence.

Note: The neighboring fragments in the resulting sequence are assumed to be maximally overlapped—thereby ensuring inclusion in the resulting sequence as many fragments as possible. The fitness function GA evaluating an individual selects the best substring of oligonucleotides, or the chromosome; i.e., the one composed of the most fragments, provided its length is equal to the given length of the reference DNA sequence.

Different GA operators for the assembly of DNA sequence fragments associated with the Human Genome project was studied in [13]. The sorted order representation and the permutation representation are compared on problems ranging from 2–34 K base pairs (KB). It is found that edge-recombination crossover used in conjunction with several specialized operators performs the best. Other relevant investigations for solving FAP using GAs are available in [14] and [15].

III. Bioinformatics Tasks and Application of EAs

We now describe the different problems and associated tasks involved in bioinformatics, their requirements, and the ways in which computational models can be formulated to solve them. The classified tasks (as mentioned in Section II-A) are first explained in this section, followed by a description of how GAs and other evolutionary techniques are applied in solving them.

A. Alignment and Comparison of DNA, RNA, and Protein Sequences

An alignment is a mutual placement of two or more sequences which exhibit where the sequences are similar, and where they differ. These include alignment and prediction of DNA, RNA, protein sequences, and fragment assembly of DNA. An optimal alignment is the one that exhibits the most correspondences and the fewest differences. It is the alignment with the highest score, but which may or may not be biologically meaningful. Basically, there are two types of alignment methods: global alignment and local alignment. Global alignment [16] maximizes the number of matches between the sequences along the entire length of the sequence. Local alignment [17] gives a highest scoring to local match between two sequences. Global alignment includes all the characters in both sequences from one end to the other, and is excellent for sequences that are known to be very similar. If the sequences being compared are not similar over their entire lengths, but have short stretches within them that have high levels of similarity, a global alignment may miss the alignment of these important regions, and local alignment is then used to find these internal regions of high similarity. Pairwise comparison and alignment of protein or nucleic acid sequences is the foundation upon which most other bioinformatics tools are built. Dynamic programming (DP) is an algorithm that allows for efficient and complete comparison of two (or more) biological sequences, and the technique is known as the Smith–Waterman algorithm [17]. It refers to a programmatic technique or algorithm which, when implemented correctly, effectively makes all possible pairwise comparisons between the characters (nucleotide or amino acid residues) in two biological sequences. Spaces may need to be inserted within the sequences for alignment. Consecutive spaces are defined as a gap. The final result is a mathematically, but not necessarily biologically, optimal alignment of the two sequences. A similarity score is also generated to describe how similar the two sequences are, given the specific parameters used.

A multiple alignment arranges a set of sequences in a manner that positions thought to be homologous are placed in a common column. There are different conventions regarding the scoring of a multiple alignment. In one approach, the scores of all the induced pairwise alignments contained in a multiple alignment are simply added. For a linear gap penalty, this amounts to scoring each column of the alignment by the sum of pair (SP-) scores in this column [10]. Although it would be biologically meaningful, the distinctions between global, local, and other forms of alignment are rarely made in a multiple alignment. A full set of optimal pairwise alignments among a given set of sequences will generally overdetermine the multiple alignment. If one wishes to assemble a multiple alignment from pairwise alignments, one has to avoid "closing loops," i.e., one can put together pairwise alignments as long as no new pairwise alignment is included to a set of sequences which is already part of the multiple alignment.

Methods: GAs are used to solve the problem of multiple sequence alignment. Before we describe them, it may be mentioned that other optimization methods, such as simulated annealing [18] and Gibbs sampling [19], are also used in this regard. Simulated annealing can sometimes be very slow, although it works well as an alignment improver. Gibbs sampling is good in finding local multiple alignment block with no gaps, but is not suitable in gapped situations.

It was first described in Sequence Alignment by Genetic Algorithm (SAGA) [20] how to use GA to deal with sequence alignments in a general manner (without DP), shortly before a similar work by Zhang *et al.* [21]. The population is made of alignments, and the mutations are processing programs that shuffle the gaps using complex methods. In SAGA, each individual (chromosome) is a multiple alignment of sequences. The population size is 100 and there is no identical individual in it. To create one of these alignments, a random offset is chosen for all the sequences (the typical range is from 0–50 for sequences 200 residues long) and each sequence is moved to the right, according to its offset. The sequences are then padded with null signs in order to have the same length. The fitness of each individual (alignment) is computed as the score of the corresponding alignment. All the individuals are ranked according to their fitness, and the weakest are replaced by new children. Only a portion (e.g., 50%) of the population are replaced during each generation. Two types of crossover, two types of gap insertion mutation, 16 types of block shuffling mutation, one block searching mutation, and two local optimal rearrangemet mutation operators are used in SAGA. During initialization of the program, all the operators have the same probability of being used, equal to $1/22$. An automatic procedure (dynamic schedules, proposed by Davis [22]) for selecting operator has been implemented in SAGA. In this model, an operator has a probability of being used that is a function of the efficiency it has recently (e.g., ten last generations) displayed at improving alignments. The credit an operator receives when performing an improvement is also shared with the operators that came before, and may have played a role in this improvement. Thus, each time a new individual is generated, if it yields some improvement on its parents, the operator that is directly responsible for its creation gets the largest part of the credit (e.g., 50%). Then the operator(s) responsible for the creation of the parents also get their share of the remaining credit (50% of the remaining credit; i.e., 25% of the original credit), and so on. This report of the credit goes on for some specified number of generations (e.g., 4). After a given number of generations (e.g., 10) these results are summarized for each of the operators. The credit of an operator is equal to its total credit divided by the number of children it generated. This value is taken as usage probability and will remain unchanged until the next assessment, ten generations later. To avoid the early loss of some operators that may become useful later on, all the operators are assigned a minimum probability of being used (the same for all them, typically equal to half their original probability, i.e., $1/44$). The automatically assigned probabilities of usage at different stages in the alignment give a direct measure of usefulness or redundancy for a new operator. SAGA is stopped when the search has been unable to improve for some specified number of generations (typically 100). This condition is the most widely used when working on a population with no duplicates.

Other approaches [23]–[25] are similar to SAGA where, a population of multiple alignment evolves by selection, combination, and mutation. The main difference between SAGA and recent algorithms has been the design of better mutation operators. A simple GA, applied in a straightforward fashion to the alignment problem, was not very successful [20]. The main devices which allow GAs to efficiently reach very high quality solutions are the use of: 1) a large number of mutation and crossover operators, and 2) their automatic scheduling. The GA based methods are not very efficient at handling all types of situations. So it is necessary to invent some new operators designed specifically for the problem, and to slot them into the existing scheme. Most of the investigations using GAs for sequence alignment are on different data sets and results are compared with that of CLUSTAL W [26], so a clear comparison between the GA based methods is not possible. A hybrid approach [27], [28], uses the searching ability of GAs for finding match blocks, and dynamic programming for producing close to optimum alignment of the match blocks. This method is faster and produces better results than pure GA and DP based approaches. Here, the population size is determined as $Q = mn/100$, where m is the average sequence length and n is the number of sequences.

In [29], it was pointed out that the combination of high-performance crossover and mutation operators does not always lead to a high performance GA for sequencing because of the negative combination effect of those two operators. A high-performance GA can be constructed by utilizing the positive combination effect of crossover and mutation.

Other relevant investigations for solving multiple sequence alignment using GAs are available in [30]–[34].

B. Gene Mapping on Chromosomes

Gene mapping is defined as the determination of relative positions of genes on a chromosome, and the distance between them. A gene map helps molecular biologists to explore a genome. A primary goal of the Human Genome Project is to make a series of descriptive diagram maps of each human chromosome at increasingly finer resolutions. Two types of gene maps, *viz.*, cytogenetic map and linkage map are generally used. A cytogenetic map, also known as a physical map, offers a physical picture of the chromosome. In a cytogenetic map, the chromosomes are divided into smaller fragments that can be propagated and characterized, and then the fragments are ordered (mapped) to correspond to their respective locations on the chromosomes. A genetic linkage map shows the relative locations (order) of specific DNA markers along the chromosome.

Since EAs have been used for determining the genetic linkage map, it is described here briefly. The genetic markers in a linkage map are generally small, but precisely defined sequences and can be expressed as DNA regions (genes) or DNA segments that have no known coding function but whose inheritance pattern can be followed. DNA sequence differences are especially useful markers because they are plentiful and easy to characterize precisely [10]. A linkage map is constructed by the following:

1) producing successive generations (chromosomes) of certain organisms through crossover (recombination), and
2) analyzing the observed segregation percentages of certain characteristics in each chromosomal data to find the actual gene order.

A linkage map shows the order and relative distance between genes, but has two drawbacks [10]. First, it does not tell the actual distance of genes, and second, if genes are very close, one can not resolve their order, because the probability of separation is so small that the observed recombinant frequencies are all zero. The closer two genes are, the lower the probability that they will be separated during the DNA repair or replication process, and hence the probability is greater that they will be inherited together. For example, suppose a certain stretch of DNA has been completely sequenced, giving us a sequence S. If we know which chromosome S came from, and if we have a physical map of this chromosome, we could try to find one of the map's markers in S. If the process succeeds, we can locate the position of S in the chromosome. The best criterion to quantify how well a map explains the data set is the multipoint maximum likelihood (exploiting the data on all markers simultaneously) of the map. Given a probabilistic model of recombination for a given family structure, a genetic map of a linkage group, and the set of available observations on markers of the linkage group, we can define the probability that the observations may have occurred given the map. This is termed the likelihood of the map. The likelihood is only meaningful when compared to the likelihood of other maps.

The problem of finding a maximum likelihood genetic map can be described as a double optimization problem. For a given gene order, there is the problem of finding recombination probabilities (crossover probabilities) that yield a maximum multipoint likelihood; then, one must find an order that maximizes this maximum likelihood. The first problem is solved by using the expectation maximization (EM) algorithm. The second problem is more difficult, because the number of possible orders to consider for N markers is $N!/2$. This type of combinatorial problem can be handled efficiently by evolutionary algorithms. The problem of finding an order of genes that maximizes the maximum multipoint likelihood is equivalent to the symmetric TSP. One can simply associate one imaginary city to each marker, and define as the distance between two cities the inverse of the elementary contribution to the log-likelihood defined by the corresponding pair of markers.

Methods: The method of genetic mapping described in [35] is embodied in a hybrid framework that relies on the statistical optimization algorithms (e.g., expectation maximization) to handle the continuous variables (recombination probabilities), while GAs handle the ordering problem of genes. The efficiency of the approach lies critically in the introduction of greedy local search in the fitness evaluation of the GA, using a neighborhood structure inspired by the TSP. A population size ranging from 25–250 has been used for number of markers between 10–29.

In gene mapping problem, Gunnels *et al.* [36] compared GAs with simulated annealing (SA), and found that the GA-based method always converges to a good solution faster since its population-based nature allows it to take advantage of the extra information to construct good local maps that can then be used to construct good global maps.

In canonical GAs with the fixed map it is difficult to design the map without *a priori* knowledge of the solution space. This is overcome in [37], where GAs using a coevolutionary approach are utilized for exploring not only within a part of the solution space defined by the genotype-phenotype map, but also with the map itself. Here, the genotype-phenotype map is improved adaptively during the searching process for solution candidates. The algorithm is applied to three-bit deceptive problems as a kind of typical combinatorial optimization problem. The difficulty with canonical GAs can be controlled by the genotype-phenotype map, and the output shows fairly good performance.

Relevant investigation for gene mapping using GAs is also available in [38].

C. Gene Finding and Promoter Identification From DNA Sequences

Automatic identification of the genes from the large DNA sequences is an important problem in bioinformatics [39]. A cell mechanism recognizes the beginning of a gene or gene cluster with the help of a promoter and is necessary for the initiation of transcription. The promoter is a region before each gene in the DNA that serves as an indication to the cellular mechanism that a gene is ahead. For example, the codon AUG (which codes for methionine) also signals the start of a gene. Recognition of regulatory sites in DNA fragments has become particularly popular because of the increasing number of completely sequenced genomes and mass application of DNA chips. Experimental analyses have identified fewer than 10% of the potential promoter regions, assuming that there are at least 30 000 promoters in the human genome, one for each gene.

Methods: Using GA, Kel *et al.* [40] designed sets of appropriate oligonucleotide probes capable of identifying new genes belonging to a defined gene family within a cDNA or genomic library. One of the major advantages of this approach is the low homology requirement to identify functional families of sequences with little homology.

Levitsky *et al.* [41] described a method for recognizing promoter regions of eukaryotic genes with an application on Drosophila melanogaster. Its novelty lies in realizing the GA to search for an optimal partition of a promoter region into local nonoverlapping fragments, and selection of the most significant dinucleotide frequencies for the fragments.

The method of prediction of eukaryotic Pol II promoters from DNA sequence [42] takes advantage of a combination of elements similar to neural networks and GAs to recognize a set of discrete subpatterns with variable separation as one pattern: a promoter. The neural networks use, as input, a small window of DNA sequence, as well as the output of other neural networks. Through the use of GAs, the weights in the neural networks are optimized to discriminate maximally between promoters and nonpromoters.

D. Interpretation of Gene Expression and Microarray Data

Gene expression is the process by which a gene's coded information is converted into the structures present and operating in the cell. Expressed genes include those that are transcribed into mRNA and then translated into protein, and those that are transcribed into RNA but not translated into protein (e.g., transfer and ribosomal RNAs). Not all genes are expressed, and gene expression involves the study of the expression level of genes in the cells under different conditions. Conventional wisdom is that gene products which interact with each other are more likely to have similar expression profiles than if they do not [43].

Microarray technology [44] allows expression levels of thousands of genes to be measured at the same time. A microarray is typically a glass (or some other material) slide, on to which DNA molecules are attached at fixed locations (spots). There may be tens of thousands of spots on an array, each containing a huge number of identical DNA molecules (or fragments of identical molecules), of lengths from twenty to hundreds of nucleotides. Each of these molecules ideally should identify one gene or one exon in the genome. The spots are either printed on the microarrays by a robot, or synthesized by photolithography (as in computer chip productions), or by ink-jet printing.

Many unanswered and important questions could potentially be answered by correctly selecting, assembling, analyzing, and interpreting microarray data. Clustering is commonly used in microarray experiments to identify groups of genes that share similar expressions. Genes that are similarly expressed are often coregulated and are involved in the same cellular processes. Therefore, clustering suggests functional relationships between groups of genes. It may also help in identifying promoter sequence elements that are shared among genes. In addition, clustering can be used to analyze the effects of specific changes in experimental conditions, and may reveal the full cellular responses triggered by those conditions.

A good solution of the gene ordering problem (i.e., finding optimal order of DNA microarray data) will have similar genes grouped together, in clusters. A notion of distance must thus be defined in order to measure similarity among genes. A simple measure is the Euclidean distance (other options are possible using Pearson correlation, absolute correlation, Spearman rank correlation, etc.). One can thus construct a matrix of intergene distances. Using this matrix one can calculate the total distance between adjacent genes and find that permutation of genes for which the total distance is minimized [similar to what is done in the TSP using GA (Section II-B)].

Methods: Finding the optimal order of microarray data is known to be NP complete. Tsai *et al.* [45] formulated this as the traveling salesman problem and the applied family competition GA (FCGA), to solve it. The edge assembly crossover (EAX) is combined with the family competition concept and neighbor join mutation (NJ). In [46], a modified EAX and NJ are used in EA for efficiently optimizing the clustering and ordering of genes, ranging in size from 147 to 6221. Chromosomes in EAs are represented as a permutation of genes. The size of the population is assumed to equal to the number of genes in problems that involved fewer than 1000 genes, and half of the number of gens in larger problems. Fitness of chromosomes are evaluated from (1) and distance matrix is formed using pearson correlation. Crossover and mutation rates are set to one. Microarray data analysis is a competitive field, and no decisive measure of the performance of methods is available, so methods using EAs for microarray are compared in the TSP framework [46].

Garibay *et al.* [47] introduced a proportional GA (PGA) that relies on the existence or nonexistence of genes to determine the information that is expressed. The information represented by a PGA individual depends only on what is present in the individual, and not on the order in which it is present. As a result, the order of the encoded information is free to evolve in response to factors other than the value of the solution.

E. Gene Regulatory Network Identification

Inferring a gene regulatory network from gene expression data obtained by DNA microarray is considered one of the most challenging problems in the field of bioinfomatics [48]. An important and interesting question in biology, regarding the variation of gene expression levels, is how genes are regulated. Since almost all cells in a particular organism have an identical genome, differences in gene expression, and not the genome content, are responsible for cell differentiation during the life of the organism.

For gene regulation, an important role is played by a type of proteins called transcription factors [10]. The transcription factors bind to specific parts of the DNA, called transcription factor binding sites (i.e., specific, relatively short combinations of A, T, C or G), which are located in promoter regions. Specific promoters are associated with particular genes and are generally not too far from the respective genes, although some regulatory effects can be located as far as 30 000 bases away, which makes the definition of the promoter difficult.

Transcription factors control gene expression by binding to the gene's promoter and either activating (switching on) the gene or repressing it (switching it off). Transcription factors are gene products themselves, and therefore, in turn, can be controlled by other transcription factors. Transcription factors can control many genes, and some (probably most) genes are controlled by combinations of transcription factors. Feedback loops are possible. Therefore, we can talk about gene regulation networks. Microarrays and computational methods are playing a major role in attempts to reverse engineer gene networks from various observations.

Methods: In gene network inference problem the objective is to predict a regulating network structure of the interacting genes from the observed data; i.e., expression pattern. The gene expressions are regulated in discrete state transitions such that the expression levels of all genes are updated simultaneously. In [49], each real valued chromosomes (in GAs) represents the expression level of all the genes. Each gene has a specific expression level for another gene; so, for N genes there are N^2 expression levels. Fitness of the chromosomes are evaluated by absolute error with generated expression pattern (The sum of all expressions) from the target expression pattern. A population size of 2500, 5000, and 7000 are taken for 5, 7, and 10 genes, respectively. The GA run for 150 generations with a crossover and mutation rate of 0.99 and 0.01, respectively. Relevant investigations using GAs are also available in [50]–[53].

F. Construction of Phylogenetic Trees for Studying Evolutionary Relationship

All species on earth undergo a slow transformation process called evolution. To explain the evolutionary history of today's species and how species relate to one another in terms of common ancestors, trees are constructed whose leaves represent the present day species, and interior nodes which represent the hypothesized ancestors. These kind of labeled binary trees are called phylogenetic trees [10]. Phylogenetic analysis is used to study the evolutionary relationship.

Phylogenies are reconstructed based on comparisons between present-day objects. The term object is used to denote the units for which one wants to reconstruct the phylogeny. Input data required for constructing phylogeny are classified into two main categories [10]. 1) Discrete character, such as beak shape, number of fingers of presence or absence of a molecular restriction site. Each character can have a finite number of states. The data relative to these characters are placed in an objects character matrix called character state matrix. 2) Comparative numerical data, called distances between objects. The resulting matrix is called a distance matrix.

Given data (character state matrix or distance matrix) for n taxa (objects), the phylogenetic tree reconstruction problem is to find the particular permutation of taxa that optimize the criteria (distance). The problem is equivalent to the problem of TSP. One can simply associate one imaginary city to each taxa, and define as the distance between two cities the data obtained from the data matrix for the corresponding pair of taxas.

Methods: Exhaustive search of the space of phylogenetic trees is generally not possible for more than 11 taxa, and so algorithms for efficiently searching the space of trees must be developed. Phylogeny reconstruction is a difficult computational problem, because the number of possible solutions (permutations) increases with the number of included taxa (objects) [54]. Branch-and-bound methods can reasonably be applied for up to about 20 taxa, so scientists generally rely on heuristic algorithms, such as stepwise-addition and star-decomposition methods. However, such algorithms generally involve a prohibitive amount of computation time for large problems and often find trees that are only locally optimal. Heuristic search strategies using GAs [54]–[57] can overcome the aforementioned problems by faster reconstruction of the optimal trees with less computing power.

In [57], each chromosome in GA is encoded as a permutation of 15 taxas (the same as TSP); and selection, crossover, and mutation operations are performed to minimize the distance among the taxas. Here, each taxa is an amino acid sequence taken from the GenBank, and distance between them is computed as an alignment score using CLUSTAL W [26]. The GA population consisted of 20 trial trees. A crossover probability of 0.5 and mutation probability of 0.2 has been used. Optimal trees are obtained after 138 generations. The only difference with TSP is that the end points of the chromosome GA are relevant in phylogenetic trees as they represent the starting and the end points of evolutionary relationship. GAs has also been used [58] for automatic self-adjustment of the parameters of the optimization algorithm of phylogenetic trees.

G. DNA Structure Prediction

DNA structure plays an important role in a variety of biological processes. Different dinucleotide and trinucleotide scales have been described to capture various aspects of DNA structure including base stacking energy, propeller twist angle, protein deformability, bendability, and position preference [59]. three-dimension DNA structure and its organization into chromatin fibres is essential for its functions, and is applied in protein binding sites, gene regulation, triplet repeat expansion diseases, etc. DNA structure depends on the exact sequence of nucleotides and largely on interactions between neighboring base pairs. Different sequences can have different intrinsic structures. Periodic repetitions of bent DNA in phase with the helical pitch will cause DNA to assume a macroscopically curved structure. Flexible or intrinsically curved DNA is energetically more favorable to wrap around histones than rigid and unbent DNA.

The curvature of a space line is defined as the derivative, dt/dl, of the tangent vector t, along the line l. Its modulus is the inverse of the curvature radius, and its direction is that of the main normal to the curve [61]. In the case of DNA, the line corresponds to the helical axis and the curvature is a vectorial function of the sequence. The curvature represents the angular deviation ($|C(n)|$) between the local helical axes of the nth and $(n+1)$th base pairs (Fig. 4). Under similar external conditions, the intrinsic curvature function represents the differential behavior of different DNA tracts and corresponds to the most stable superstructure. The physical origin of curvature is still a matter of debate [60]; it is, however, a result of the chemical and, consequently, stereochemical, inhomogeneity of the sequence, which gives rise to different macroscopic manifestations. These manifestations change with the thermodynamic conditions such as pH, the ionic force, the kind of counterions, and obviously the temperature as a result of perturbations on the intrinsic curvature depending on the sequence-dependent bendability. Therefore, it is generally useful to characterize a DNA superstructure with the so-called intrinsic curvature function [60].

Methods: The 3-D spatial structure of a methylene-acetal-linked thymine dimer present in a 10 basepair (bp) sense–antisense DNA duplex was studied in [62] with a GA designed to interpret nuclear Overhauser effect (NOE) inter-proton distance restraints. Trial solutions (chromosomes in GAs) are encoded on bit strings which represents torsion angles between atoms. From these torsion angles, atomic coordinates, needed for the fitness function are calculated using the DENISE program. The problem is to find a permutation of torsion angles (eight torsion angles for each nucleotide in DNA) that minimizes the atomic distance between protons of neucleotides. The GA minimizes the difference between distances in the trial structures and distance restraints for a set of 63 proton–proton distance restraints defining the methylene-acetal-linked thymine dimer. The torsion angles were encoded by Gray coding and the GA population consisted of 100 trial structures. Uniform crossover with a probability of 0.9 and mutation rate of 0.04 was used. It was demonstrated that the bond angle geometry around the methylene-acetal linkage plays an important role in the optimization.

A hybrid technique involving artificial neural networks (ANN) and GA is described in [63] for optimization of DNA curvature characterized in terms of the reliability (RL) value. In this approach, first an ANN approximates (models) the nonlinear relationship(s) existing between its input and output example data sets. Next, the GA searches the input space of the ANN with a view to optimize the ANN output. Using this methodology, a number of sequences possessing high RL values have been obtained and analyzed to verify the existence of features known to be responsible for the occurrence of curvature.

H. RNA Structure Prediction

An RNA molecule is considered as a string of n characters $R = r_1 r_2 \cdots r_n$ such that $r_i \varepsilon A, C, G, U$. Typically n is in the hundreds, but could also be in thousands. The secondary structure of the molecule is a collection S of a set of stems and each stem consisting of a set of consecutive base pairs $(r_i r_j)$ (e.g., GU, GC, AU). Here, $1 \leq i \leq j \leq n$ and (r_i and r_j) are connected through hydrogen bonds. If $(r_i, r_j) \varepsilon S$, in principle we should require that r_i be a complement to r_j and that $j - i > t$, for a certain threshold t (because it is known that an RNA molecule does not fold too sharply on itself). With such an assumption [10], the total free energy E of a structure S is given by

$$E(s) = \sum_{(r_i, r_j) \in S} \alpha(r_i, r_j) \qquad (3)$$

where $\alpha(r_i, r_j)$ gives the free energy of base pair (r_i, r_j). Generally, the adopted convention is $\alpha(r_i, r_j) < 0$, if $i \neq j$, and $\alpha(r_i, r_j) = 0$, if $i = j$.

Attempts to predict automatically the RNA secondary structure can be divided in essentially two general approaches. The first involves the overall free energy minimization by adding contributions from each base pair, bulged base, loop, and other elements [64]. EAs are found to be suitable for this purpose. Chromosomes in EAs are encoded to represent the RNA structure and fitness of each chromosome is evaluated in terms of free energy (3). The second type of approach [65] is more empirical and it involves searching for the combination of nonexclusive helices with a maximum number of base pairings, satisfying the condition of a tree like structure for the bio-molecule. Within the latter, methods using dynamic programming (DP) are the most common [65], [66]. While DP can accurately compute the minimum energy within a given thermodynamic model, the natural fold of RNA is often in a suboptimal energy state and requires soft computing EAs rather than hard computing DP.

RNA may enter intermediate conformational states that are key to its functionality. These states may have a significant impact on gene expression. The biologically functional states of RNA molecules may not correspond to their minimum energy state, and kinetic barriers may exist that trap the molecule in a local minimum. In addition, folding often occurs during transcription, and cases exist in which a molecule will undergo transitions between one or more functional conformations before reaching its native state. Thus, methods for simulating the folding pathway of an RNA molecule and locating significant intermediate states are important for the prediction of RNA structure and its associated function.

Methods: The possibilities of using GAs for the prediction of RNA secondary structure were investigated in [67] and [68]. The implementations used a binary representation for the solutions (chromosomes in GAs). The algorithm, using the procedure of stepwise selection of the most fit structures (similarly to natural evolution), allows different models of fitness for determining RNA structures. The analysis of free energies for intermediate foldings suggests that in some RNAs, the selective evolutionary pressure suppresses the possibilities for alternative structures that could form in the course of transcription. The algorithm had inherent incompatibilities of stems due to the binary representation of the solutions.

Wiese *et al.* [69] used GAs to predict the secondary structure of RNA molecules, where the secondary structure is encoded as a permutation similar to path representation in TSP (each helix is associated to one imaginary city) to overcome the inherent incompatibilities of binary representation for RNA molecule structure prediction. They showed that the problem can be decomposed into a combinatorial problem of finding the subset of helices from a set of feasible helices leading to a minimum energy [using (3)] in the molecule. More specifically, the algorithm predicts the specific canonical base pairs that will form hydrogen bonds and build helices. Different combinations of crossover and mutation probabilities ranging from 0.0 to 1.0 in increments of 0.01 and 0.1 were tested for 400 generations with a population size of 700 (maximum). Results on RNA sequences of lengths 76, 210, 681, and 785 nucleotides were provided. It was shown that the keep-best reproduction operator has similar benefits as in the traveling salesman problem domain. A comparison of several crossover operators was also provided.

A massively parallel GA for the RNA folding problem has been used in [70]–[72]. The authors demonstrated that the

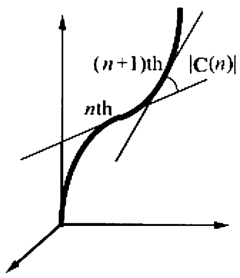


Fig. 4. Representation of the DNA curvature in terms of angular deviation between the local helical axes of the turn centered on the nth and $(n+1)$th basepairs [60].

GA with an improved mutation operator predicts more correct (true-positive) stems and more correct base pairs than could have been a predicted with DP algorithm.

I. Protein Structure Prediction and Classification

Identical protein sequences result in identical 3-D structures. So it follows that similar sequences may result in similar structures, and this is usually the case. The converse, however, is not true: identical 3-D structures do not necessarily indicate identical sequences. It is because of this that there is a distinction between "homology" and "similarity." There are examples of proteins in the databases that have nearly identical 3-D structures, and are therefore homologous, but do not exhibit significant (or detectable) sequence similarity. Pairwise comparisons do not readily show positions that are conserved among a whole set of sequences and tend to miss subtle similarities that become visible when observed simultaneously among many sequences. Thus, one wants to simultaneously compare several sequences.

Structural genomics is the prediction of the 3-D structure of a protein from the primary amino acid sequence [73]. This is one of the most challenging tasks in bioinformatics. The five levels of protein structure are described below. Three of them are illustrated in Fig. 5.

1) Primary structure is the sequence of amino acids that compose the protein.
2) The secondary structure of a protein is the spatial arrangement of the atoms constituting the main protein backbone. Linus Pauling was the first to develop a hypothesis for different potential protein secondary structures. He developed the α-helix structure and later the β-sheet structure for different proteins. An α-helix is a spiral arrangement of the protein backbone in the form of a helix with hydrogen bonding between side-chains. The β-sheets consist of parallel or antiparallel strands of amino acids linked to adjacent strands by hydrogen bonding. Collagen is an example of a protein with β-sheets serving as its secondary structure.
3) The super-secondary structure (or motif) is the local folding pattern built up from particular secondary structures. For example, the EF-hand motif consists of an α-helix, followed by a turn, followed by another α-helix.
4) Tertiary structure is formed by packing secondary structural elements linked by loops and turns into one or several compact globular units called domains; i.e., the folding of the entire protein chain.
5) A final protein may contain several protein subunits arranged in a quaternary structure.

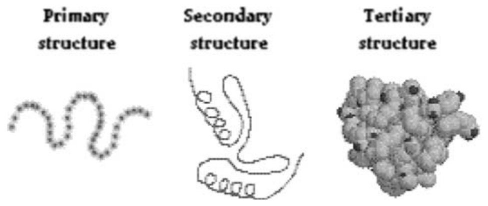


Fig. 5. Three levels of protein structure.

Protein sequences almost always fold into the same structure in the same environment. Hydrophobic interaction, hydrogen bonding, electrostatic, and other Van der Waals-type interactions also contribute to determine the structure of the protein. Many efforts are underway to predict the structure of a protein, given its primary sequence. A typical computation of protein folding would require computing all the spatial coordinates of atoms in a protein molecule, starting with an initial configuration and working up to a final minimum-energy folding configuration [10]. Sequence similarity methods can predict the secondary and tertiary structures based on homology to known proteins. Secondary structure predictions methods include Chou–Fasman [73], neural network [74], [75], nearest neighbor methods [76], [77], and Garnier–Osguthorpe–Robson [78]. Tertiary structure prediction methods are based on energy minimization, molecular dynamics, and stochastic searches EAs of conformational space.

Proteins clustered together into families are clearly evolutionarily related. Generally, this means that pairwise residue identities between the proteins are 30% and greater. Proteins that have low sequence identities, but whose structural and functional features suggest that a common evolutionary origin is probable, are placed together in superfamilies.

Methods: The work of Unger *et al.* [79]–[81] is one of the earlier investigations that discussed the reduced 3-D lattice protein folding problem for determining tertiary structure of protein in a GA framework. In this model, the energy function of protein chains is optimized. The encoding proposed by Unger *et al.* is a direct encoding of the direction of each peptide from the preceding peptide (five degrees of freedom, disallowing back move). Peptides are represented as single point units without side chains. Each peptide is represented by three bits to encode five degrees of freedom. The evaluation function solely evaluates nonsequential hydrophobe to hydrophobe contacts and is stated as a negative value (-1 per contact) with larger negative values indicating better energy conformations (thus stating the problem in terms of minimization). The algorithm begins with a population of identical unfolded configurations. Each generation begins with a series of K mutations being applied to each individual in the population, where K is equal to the length of the encoding. These mutations are filtered using a Monte Carlo acceptance algorithm which disallows lethal configurations (those with back move), always accepts mutations resulting in better energy, and accepts increased energy mutations based upon a threshold on the energy gain which becomes stricter over time. One-point crossover with an additional random mutation at the crossover point follows, producing a single offspring for each selected pair of parents; however, lethal configurations are rejected. In this situation, the crossover operation is retried for a given pair of parents until a nonlethal offspring can be located. Offspring are accepted using a second Monte Carlo filter which accepts all reduced energy confirmations and randomly accepts increased energy offspring again using a cooling threshold on the energy gain. The algorithm uses 100% replacement of all individuals in a generation through crossover except the single best, elitist, individual. Test data consisted of a series of ten randomly produced 27 length sequences and ten randomly produced 64 length sequences. The algorithm operated on each of the 27 and 64 length sequence for roughly 1.2 million and 2.2 million function evaluations, respectively, using a population size of 200. Performance comparisons were given between the above algorithm and a pure Monte Carlo approach which greatly favored the former. While the encoding and evaluation function proposed by Unger and Moult are fairly straightforward, the algorithm differs from a standard GA approach in several aspects. Most notable are the nonrandom initialization, the high level of mutation, and the Monte Carlo filtering of both the mutation and crossover results, which resembles typical simulated annealing approaches.

Patton *et al.* [82] determined tertiary structures of proteins based on the concept of Unger *et al.* [36], [40]. They enlarged the representation from three to seven bits per peptide in order to encode one of the 120 permutations of the five allowable directions for each. It was shown that the GA indeed appears to be effective for determining the tertiary structure with far fewer computational steps than that reported by Unger *et al.*

Natalio *et al.* [83], [84] investigated the impact of several algorithmic factors for a simple protein structure prediction problem: the conformational representation, the energy formulation, and the way in which infeasible conformations are penalized. Their analysis leads to specific recommendations for both GAs and other heuristic methods for solving PSP on the HP model. A detailed comparison between the work of Unger *et al.* and Patton *et al.* and an algorithm using GAs to overcome their limitations has also been presented [84].

A hill-climbing GA for simulation of protein folding has been described in [85]. The program builds a set of Cartesian points to represent an unfolded polypeptide's backbone. The dihedral angles determining the chain's configuration are stored in an array of chromosome structures that is copied and then mutated. The fitness of the mutated chain's configuration is determined by its radius of gyration. A four-helix bundle was used to optimize the simulation conditions. The program ran 50% faster than the other GA programs, and tests on 100 nonredundant structures produced results comparable to that of other GAs.

In [86], features are extracted from protein sequences using a position specific weight matrix. Thereafter, a genetic algorithm based fuzzy clustering scheme [87] is used for generating prototypes of the different superfamilies. Finally, superfamily classification of new sequences is performed by using the nearest neighbor rule.

Other investigations on protein structure prediction are available in [88]–[100]. An overview and state-of-the-art of the applications of EAs only for the protein folding problem is described in [101], whereas the relevance of GAs in several bioinformatics tasks is discussed in the present article.

J. *Molecular Design and Molecular Docking*

When two molecules are in close proximity, it can be energetically favorable for them to bind together tightly. The molecular docking problem is the prediction of energy and physical configuration of binding between two molecules. A typical application is in drug design, in which one might dock a small molecule that is a described drug to an enzyme one wishes to target. For example, HIV protease is an enzyme in the AIDS virus that is essential to its replication. The chemical action of the protease takes place at a localized active site on its surface. HIV protease inhibitor drugs are small molecules that bind to the active site in HIV protease and stay there, so that the normal functioning of the enzyme is prevented. Docking software allows us to evaluate a drug design by predicting whether it will be successful in binding tightly to the active site in the enzyme. Based on the success of docking, and the resulting docked configuration, designers can refine the drug molecule [102].

Molecular design and docking is a difficult optimization problem, requiring efficient sampling across the entire range of positional, orientational, and conformational possibilities [103]. The major problem in molecular binding is that the search space is very large and the computational cost increases tremendously with the growth of the degrees of freedom. A docking algorithm must deal with two distinct issues: a sampling of the conformational degrees of freedom of molecules involved in the complex, and an objective function (OF) to assess its quality.

For molecular design, the structure of a flexible molecule is encoded by an integer-valued or real-valued chromosome in GA, the ith element of which contains the torsion angle for the ith rotable bond. The energy for the specified structure (conformation) can be calculated using standard molecular modeling package, and this energy is used as the fitness function for the GA. GAs try to identify a set of torsion angle values that minimize the calculated energy. GA is becoming a popular choice for the heuristic search method in molecular design and docking applications [104]. Both canonical GAs and evolutionary programming methods are found to be successful in drug design and docking. Some of them are described below.

Methods: A novel and robust automated docking method that predicts the bound conformations (structures) of flexible ligands to macromolecular targets has been developed [105]. The method combines GAs with a scoring function that estimates the free energy change upon binding. This method applies a Lamarckian model of genetics, in which environmental adaptations of an individual's phenotype are reverse transcribed into its genotype and become inheritable traits. Three search methods, *viz.*, Monte Carlo simulated annealing, a traditional GA, and the Lamarckian GA were considered, and their performance was compared in dockings of seven protein-ligand test systems having known three-dimensional structure. The chromosome is composed of a string of realvalued genes: three Cartesian coordinates for the ligand translation; four variables defining a quaternion specifying the ligand orientation; and one real-value for each ligand torsion, in that order. The order of the genes that encode the torsion angles is defined by the torsion tree created by AUTOTORS (a preparatory program used to select rotatable bonds in the ligand). Thus, there is a one-to-one mapping from the ligand's state variables to the genes of the individuals chromosome. An Individual's fitness is the sum of the intermolecular interaction energy between the ligand and the protein, and the intramolecular interaction energy of the ligand. In the GA and LGA dockings, an initial population of 50 random individuals, a maximum number of 1.5×10^6 energy evaluations, a maximum number of 27 000 generations, a mutation rate of 0.02, and a crossover rate of 0.80 have been used. Proportional selection was used, where the average of the worst energy results was calculated over a window of the previous 10 generations.

Bagchi *et al.* [106], [107] presented an evolutionary approach for designing a ligand molecule that can bind to the active site of a target protein. A two-dimensional (2-D) model was considered. A variable string length genetic algorithm (VGA) was used for evolving an appropriate arrangement of the basic functional units of the molecule to be designed. The method is superior to fixed string length GA for designing a ligand molecule to target the human rhinovirus strain 14 (causative agent for AIDS).

Chen *et al.* [108] derived a population based annealing genetic algorithm (PAG) using GAs and simulated annealing (SA). They applied it to find binding structures for three drug protein molecular pairs, including the anti-cancer drug methotrexate (MTX). All of the binding results keep the energy at low levels, and have a promising binding geometrical structure in terms of number of hydrogen bonds formed. One of the design methods of PAG, which incorporates an annealing scheme with the normal probability density function as the neighbor generation method, was described in [109]. The algorithm was used for computer-aided drug design. Using a dihydrofolate reductase enzyme with the anti-cancer drug methotrexate and two analogs of the antibacterial drug trimethoprim, PAGs can find a drug structure within several hours. A similar work is available in [110].

Christopher *et al.* [111] evaluated the use of GAs with local search in molecular docking. They investigated several GA-local search hybrids and compared results with those obtained from simulated annealing in terms of optimization success, and absolute success in finding the true physical docked configuration.

Other relevant investigations are available in [104], [112]–[120]. A survey on the application of GAs for molecular modeling, docking of flexible ligands into protein active sites, and for *de novo* ligand design is described in [121]. Advantages and limitations of GAs are mentioned for the aforementioned tasks. In contrast, the present article provides a broader overview and state-of-the-art of the applications of EAs for several bioinformatics tasks.

IV. Conclusion

The increasing availability of annotated genomic sequences has resulted in the introduction of computational genomics and proteomics, large-scale analysis of complete genomes, and the proteins that they encode for relating specific genes to diseases. The rationale for applying computational approaches to facilitate the understanding of various biological processes mainly includes the following:

1) to provide a more global perspective in experimental design;
2) to capitalize on the emerging technology of database-mining: the process by which testable hypotheses are generated regarding the function or structure of a gene or protein of interest by identifying similar sequences in better characterized organisms.

GAs appear to be a very powerful artificial intelligence paradigm to handle these issues. This article provides an overview of different bioinformatics tasks and the relevance of GAs to handle them efficiently.

Even though the current approaches in biocomputing using EAs are very helpful in identifying patterns and functions of proteins and genes, the output results are still far from perfect. There are three general characteristics that might appear to limit the effectiveness of GAs. First, the basic selection, crossover, and mutation operators are common to all applications. Therefore, research is now focussed on designing problem specific operators to get better results. Second, a GA requires extensive experimentation for the specification of several parameters so that appropriate values can be identified. Third, GAs involve a large degree of randomness and different runs may produce different results, so it is necessary to incorporate problem specific domain knowledge into GA to reduce randomness and computational time and current research is going on in this direction also. The methods are not only time-consuming, requiring UNIX workstations to run on, but might also lead to false interpretations and assumptions due to necessary simplifications. It is therefore still mandatory to use biological reasoning and common sense in evaluating the results delivered by a biocomputing program. Also, for evaluation of the trustworthiness of the output of a program, it is necessary to understand its mathematical/theoretical background to finally come up with a useful and sense-full analysis.

Other potential bioinformatics tasks for which EA can be used include the following:

1) characterization of protein content and metabolic pathways between different genomes;
2) identification of interacting proteins;
3) assignment and prediction of gene products;
4) large-scale analysis of gene expression levels;
5) mapping expression data to sequence, structural and biochemical data.

References

[1] P. Baldi and S. Brunak, *Bioinformatics: The Machine Learning Approach*. Cambridge, MA: MIT Press, 1998.

[2] R. B. Altman, A. Valencia, S. Miyano, and S. Ranganathan, "Challenges for intelligent systems in biology," *IEEE Intell. Syst.*, vol. 16, no. 6, pp. 14–20, Nov./Dec. 2001.

[3] D. Goldberg, *Genetic Algorithms in Optimization, Search, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[4] D. Bhandari, C. A. Murthy, and S. K. Pal, "Genetic algorithm with elitist model and its convergence," *Int. J. Pattern Rcognit. Artif. Intell.*, vol. 10, no. 6, pp. 731–747, 1996.

[5] L. B. Booker, D. E. Goldberg, and J. H. Holland, "Classifier systems and genetic algorithms," *Artif. Intell.*, vol. 40, no. 1–3, pp. 235–282, 1989.

[6] M. Mitchell, S. Forrest, and J. H. Holland, "The royal road for genetic algorithms: Fitness landscapes and GA performance," in *Proc. 1st Eur. Conf. Artificial Life*, 1992.

[7] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Mach. Learn.*, vol. 3, pp. 95–100, 1988.

[8] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Cambridge, MA: MIT Press, 1992.

[9] L. Davis, *Handbook of Genetic Algorithm*. New York: Van Nostrand Reinhold, 1991.

[10] J. Setubal and J. Meidanis, *Introduction to Computational Molecular Biology*. Boston, MA: Thomson, 1999.

[11] A. S. Wu and R. K. Lindsay, "A survey of intron research in genetics," in *Proc. 4th Conf. Parallel Problem Solving from Nature*, pp. 101–110, 1996.

[12] J. Chen, E. Antipov, B. Lemieux, W. Cedeno, and D. H. Wood, "DNA computing implementing genetic algorithms," in *Evolution as Computation*, New York: Springer-Verlag, pp. 39-49, 1999.

[13] R. J. Parsons, S. Forrest, and C. Burks, "Genetic algorithms, operators, and DNA fragment assembly," *Mach. Learn.*, vol. 21, no. 1–2, pp. 11–33, 1995.

[14] ——, "Genetic algorithms for DNA sequence assembly," in *Proc. 1st Int. Conf. Intelligent Systems in Molecular Biology*, pp. 310–318, 1993.

[15] R. J. Parsons and M. E. Johnson, "DNA fragment assembly and genetic algorithms. New results and puzzling insights," in *Int. Conf. Intelligent Systems in Molecular Biology*, 1995, pp. 277–284.

[16] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *J. Mol. Biol.*, vol. 48, pp. 443–453, 1970.

[17] T. F. Smith and M. S. Waterman, "Identification of common molecular sequences," *J. Mol. Biol.*, vol. 147, pp. 195–197, 1981.

[18] E. Aart and V. P. Laarhoven, *Simulated Annealing: A Review of Theory and Applications*. Norwell, MA: Kluwer, 1987.

[19] C. Lawrence, S. Altschul, M. Boguski, J. Liu, A. Neuwald, and J. Wootton, "Detecting subtle sequence signals: A Gibbs sampling strategy for multiple alignment," *Science*, vol. 262, pp. 208–214, 1993.

[20] C. Notredame and D. G. Higgins, "SAGA: Sequence alignment by genetic algorithm," *Nucleic Acids Res.*, vol. 24, no 8, pp. 1515–1524, 1996.

[21] C. Zhang and A. K. C. Wong, "A genetic algorithm for multiple molecular sequence alignment," *Bioinformatics*, vol. 13, pp. 565–581, 1997.

[22] L. Davis, "Adapting operator probabilities in genetic algorithms," in *Proc. 3rd Int. Conf. Genetic Algorithms*, J. D. Schaffer, Ed., 1989, pp. 61–69.

[23] T. Yokoyama, T. Watanabe, A. Taneda, and T. Shimizu, "A web server for multiple sequence alignment using genetic algorithm," *Genome Inf.*, vol. 12, pp. 382–383, 2001.

[24] O. O'Sullivan, K. Suhre, C. Abergel, D. G. Higgins, and C. Notredame, "3DCoffee: Combining protein sequences and structures within multiple sequence alignments," *J. Mol. Biol.*, vol. 340, no. 2, pp. 385–395, 2004.

[25] C. Notredame, E. A. O'Brien, and D. G. Higgins, "RAGA: RNA sequence alignment by genetic algorithm," *Nucleic Acids Res.*, vol. 25, no. 22, pp. 4570–4580, 1997.

[26] J. D. Thompson, D. G. Higgins, and T. J. Gibson, "CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice," *Nucleic Acids Res.*, vol. 22, pp. 4673–4680, 1994.

[27] C. Zhang and A. K. C. Wong, "A technique of genetic algorithm and sequence synthesis for multiple molecular sequence alignment," in *Proc. IEEE Int. Conf. Syst. Man, and Cybernetics*, vol. 3, 1998, pp. 2442–2447.

[28] ——, "Toward efficient multiple molecular sequence alignment: A system of genetic algorithm and dynamic programming," *IEEE Trans. Syst. Man, Cybern. B*, vol. 27, no. 6, pp. 918–932, Dec. 1997.

[29] T. Murata and H. Ishibuchi, "Positive and negative combination effects of crossover and mutation operators in sequencing problems," *Evol. Comput.*, vol. 20–22, pp. 170–175, 1996.

[30] C. Zhang, "A genetic algorithm for molecular sequence comparison," in *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics*, vol. 2, 1994, pp. 1926–1931.

[31] J. D. Szustakowski and Z. Weng, "Protein structure alignment using a genetic algorithm," *Proteins*, vol. 38, no. 4, pp. 428–440, 2000.

[32] K. Hanada, T. Yokoyama, and T. Shimizu, "Multiple sequence alignment by genetic algorithm," *Genome Inf.*, vol. 11, pp. 317–318, 2000.

[33] L. A. Anbarasu, P. Narayanasamy, and V. Sundararajan, "Multiple molecular sequence alignment by island parallel genetic algorithm," *Current Sci.*, vol. 78, no. 7, pp. 858–863, 2000.

[34] H. D. Nguyen, I. Yoshihara, K. Yamamori, and M. Yasunaga, "A parallel hybrid genetic algorithm for multiple protein sequence alignment," in *Proc. Congress Evolutionary Computation*, vol. 1, 2002, pp. 309–314.

[35] C. Gaspin and T. Schiex, "Genetic algorithms for genetic mapping," in *Proc. 3rd Eur. Conf. Artificial Evolution*, 1997, pp. 145–156.

[36] J. Gunnels, P. Cull, and J. L. Holloway, "Genetic algorithms and simulated annealing for gene mapping," in *Proc. 1st IEEE Conf. Evolutionary Computation*, 1994, pp. 385–390.

[37] H. Murao, H. Tamaki, and S. Kitamura, "A coevolutionary approach to adapt the genotype-phenotype map in genetic algorithms," in *Proc. Congress Evolutionary Computation*, vol. 2, 2002, pp. 1612–1617.

[38] J. Fickett and M. Cinkosky, "A genetic algorithm for assembling chromosome physical maps," in *Proc. 2nd Int. Conf. Bioinformatics, Supercomputing, and Complex Genome Analysis*, 1993, pp. 272–285.

[39] J. W. Fickett, "Finding genes by computer: The state of the art," *Trends Genetics*, vol. 12, no. 8, pp. 316–320, 1996.

[40] A. Kel, A. Ptitsyn, V. Babenko, S. Meier-Ewert, and H. Lehrach, "A genetic algorithm for designing gene family-specific oligonucleotide sets used for hybridization: The G protein-coupled receptor protein superfamily," *Bioinformatics*, vol. 14, no. 3, pp. 259–270, 1998.

[41] V. G. Levitsky and A. V. Katokhin, "Recognition of eukaryotic promoters using a genetic algorithm based on iterative discriminant analysis," *In Silico Biol.*, vol. 3, no. 1–2, pp. 81–87, 2003.

[42] S. Knudsen, "Promoter2.0: For the recognition of PolII promoter sequences," *Bioinformatics*, vol. 15, pp. 356–361, 1999.

[43] N. M. Luscombe, D. Greenbaum, and M. Gerstein, "What is bioinformatics? A proposed definition and overview of the field," in *Yearbook Medical Informatics*: Edmonton, AB, Canada: IMIA, 2001, pp. 83–100.

[44] J. Quackenbush, "Computational analysis of microarray data," *Nat. Rev. Genetics*, vol. 2, pp. 418–427, 2001.

[45] H. K. Tsai, J. M. Yang, and C. Y. Kao, "Applying genetic algorithms to finding the optimal order in displaying the microarray data," in *Proc. GECCO*, 2002, pp. 610–617.

[46] H. K. Tsai, J. M. Yang, Y. F. Tsai, and C. Y. Kao, "An evolutionary approach for gene expression patterns," *IEEE Trans. Inf. Technol. Biomed.*, vol. 8, no. 2, pp. 69–78, Jun. 2004.

[47] A. S. Wu and I. Garibay, "The proportional genetic algorithm: Gene expression in a genetic algorithm," *Genetic Programm. Evol. Hardware*, vol. 3, no. 2, pp. 157–192, 2002.

[48] T. Akutsu, S. Miyano, and S. Kuhara, "Identification of genetic networks from a small number of gene expression patterns under the boolean network model," in *Proc. Pacific Symp. Biocomputing*, vol. 99, 1999, pp. 17–28.

[49] S. Ando and H. Iba, "Inference of gene regulatory model by genetic algorithms," in *Proc. Congress Evolutionary Computation*, vol. 1, 2001, pp. 712–719.

[50] N. Behera and V. Nanjundiah, "Trans gene regulation in adaptive evolution: A genetic algorithm model," *J. Theore. Biol.*, vol. 188, pp. 153–162, 1997.

[51] S. Ando and H. Iba, "Quantitative modeling of gene regulatory network-identifying the network by means of genetic algorithms," presented at the 11th Genome Informatics Workshop, 2000.

[52] ——, "The matrix modeling of gene regulatory networks-reverse engineering by genetic algorithms-," in *presented at the Atlantic Symp. Computational Biology and Genome Information Systems and Technology*, 2001.

[53] D. Tominaga, M. Okamoto, Y. Maki, S. Watanabe, and Y. Eguchi, "Nonlinear numerical optimization technique based on a genetic algorithm for inverse problems: Towards the inference of genetic networks," *Comput. Science and Biology (Proc. German Conf. Bioinformatics)*, 1999, pp. 127–140.

[54] P. O. Lewis, "A genetic algorithm for maximum likelihood phylogeny inference using nucleotide sequence data," *Mol. Biol. Evol.*, vol. 15, no. 3, pp. 277–283, 1998.

[55] A. R. Lemmon and M. C. Milinkovitch, "The metapopulation genetic algorithm: An efficient solution for the problem of large phylogeny estimation," *Proc. Nat. Acad. Sci.*, vol. 99, no. 16, pp. 10516–10521, 2002.

[56] K. Katoh, K. Kuma, and T. Miyata, "Genetic algorithm-based maximum-likelihood analysis for molecular phylogeny," *J. Mol. Evol.*, vol. 53, no. 4-5, pp. 477–484, 2001.

[57] H. Matsuda, "Protein phylogenetic inference using maximum likelihood with a genetic algorithm," *Pacific Symp. Biocomputing*, 1996, pp. 512–523.

[58] A. Skourikhine, "Phylogenetic tree reconstruction using self-adaptive genetic algorithm," in *IEEE Int. Symp. Bio-Informatics and Biomedical Engineering*, 2000, pp. 129–134.

[59] P. Baldi and P. F. Baisnee, "Sequence analysis by additive scales: DNA structure for sequences and repeats of all lengths," *Bioinformatics*, vol. 16, pp. 865–889, 2000.

[60] C. Anselmi, G. Bocchinfuso, P. De Santis, M. Savino, and A. Scipioni, "A theoretical model for the prediction of sequence-dependent nucleosome thermodynamic stability," *J. Biophys.*, vol. 79, no. 2, pp. 601–613, 2000.

[61] L. D. Landau and E. M. Lifshitz, *Theory of Elasticity*. New York: Pergamon, 1970.

[62] M. L. Beckers, L. M. Buydens, J. A. Pikkemaat, and C. Altona, "Application of a genetic algorithm in the conformational analysis of methylene-acetal-linked thymine dimers in DNA: Comparison with distance geometry calculations," *J. Biomol. NMR*, vol. 9, no. 1, pp. 25–34, 1997.

[63] R. V. Parbhane, S. Unniraman, S. S. Tambe, V. Nagaraja, and B. D. Kulkarni, "Optimum DNA curvature using a hybrid approach involving an artificial neural network and genetic algorithm," *J. Biomol. Struct. Dyn.*, vol. 17, no. 4, pp. 665–672, 2000.

[64] J. P. Adrahams and M. Breg, "Prediction of RNA secondary structure including pseudoknotting by computer simulation," *Nucleic Acids Res.*, vol. 18, pp. 3035–3044, 1990.

[65] M. Waterman, "RNA structure prediction," in *Methods in Enzymology*. San Diego, CA: Academic, vol. 164, 1988.

[66] M. Zuker and P. Stiegler, "Optimal computer folding of large RNA sequences using thermo-dynamics and auxiliary information," *Nucleic Acids Res.*, vol. 9, pp. 133–148, 1981.

[67] V. Batenburg, A. P. Gultyaev, and C. W. A. Pleij, "An APL-programmed genetic algorithm for the prediction of RNA secondary structure," *J. Theoret. Biol.*, vol. 174, no. 3, pp. 269–280, 1995.

[68] A. P. Gultyaev, V. Batenburg, and C. W. A. Pleij, "The computer simulation of RNA folding pathways using an genetic algorithm," *J. Mol. Biol.*, vol. 250, pp. 37–51, 1995.

[69] K. C. Wiese and E. Glen, "A permutation-based genetic algorithm for the RNA folding problem: A critical look at selection strategies, crossover operators, and representation issues," *Biosystems*, vol. 72, no. 1–2, pp. 29–41, 2003.

[70] B. A. Shapiro and J. Navetta, "A massively parallel genetic algorithm for RNA secondary structure prediction," *J. Supercomput.*, vol. 8, pp. 195–207, 1994.

[71] B. A. Shapiro and J. C. Wu, "An annealing mutation operator in the genetic algorithms for RNA folding," *Comput. Appl. Biosci.*, vol. 12, pp. 171–180, 1996.

[72] B. A. Shapiro, J. C. Wu, D. Bengali, and M. J. Potts, "The massively parallel genetic algorithm for RNA folding: MIMD implementation and population variation," *Bioinformatics*, vol. 17, no. 2, pp. 137–148, 2001.

[73] P. Chou and G. Fasmann, "Prediction of the secondary structure of proteins from their amino acid sequence," *Adv. Enzymol.*, vol. 47, pp. 145–148, 1978.

[74] S. K. Riis and A. Krogh, "Improving prediction of protein secondary structure using structured neural networks and multiple sequence alignments," *J. Comput. Biol.*, vol. 3, pp. 163–183, 1996.

[75] N. Qian and T. J. Sejnowski, "Predicting the secondary structure of globular proteins using neural network models," *J. Mol. Biol.*, vol. 202, no. 4, pp. 865–884, 1988.

[76] A. Salamov and V. Solovyev, "Prediction of protein secondary structure by combining nearest-neighbor algorithms and multiple sequence alignments," *J. Mol. Biol.*, vol. 247, pp. 11–15, 1995.

[77] S. Salzberg and S. Cost, "Predicting protein secondary structure with a nearest-neighbor algorithm," *J. Mol. Biol.*, vol. 227, pp. 371–374, 1992.

[78] J. Garnier, J. F. Gibrat, and B. Robson, "GOR method for predicting protein secondary structure from amino acid sequence," in *Methods in Enzymology*, vol. 266, 1996, pp. 540–553.

[79] R. Unger and J. Moult, "On the applicability of genetic algorithms to protein folding," in *Proc. Hawaii Int. Conf. System Sciences*, vol. 1, 1993, pp. 715–725.

[80] ——, "Genetic algorithms for protein folding simulations," *J. Mol. Biol.*, vol. 231, no. 1, pp. 75–81, 1993.

[81] ——, "A genetic algorithms for three dimensional protein folding simulations," in *Int. Conf. Genetic Algorithms*, 1993, pp. 581–588.

[82] A. Patton, W. P., III, and E. Goldman, "A standard GA approach to native protein conformation prediction," in *Proc. Int. Conf. Genetic Algorithms*, 1995, pp. 574–581.

[83] N. Krasnogor, W. E. Hart, J. Smith, and D. A. Pelta, "Protein structure prediction with evolutionary algorithms," in *Proc. Genetic and Evolutionary Computation*, vol. 2, 1999, pp. 1596–1601.

[84] N. Krasnogor, D. Pelta, P. M. Lopez, P. Mocciola, and E. Canal, "Genetic algorithms for the protein folding problem: A critical view," in *Proc. Engineering Intelligent Systems*, 1998

[85] L. Cooper, D. Corne, and M. Crabbe, "Use of a novel hill-climbing genetic algorithm in protein folding simulations," *Comput. Biol. Chem.*, vol. 27, no. 6, pp. 575–580, 2003.

[86] S. Bandyopadhyay, "An efficient technique for superfamily classification of amino acid sequences: Feature extraction, fuzzy clustering and prototype selection," *Fuzzy Sets Syst.*, vol. 152, pp. 5–16, 2005.

[87] U. Maulik and S. Bandyopadhyay, "Fuzzy partitioning using real coded variable length genetic algorithm for pixel cassification," *IEEE Trans. Geosci. Remote Sens.*, vol. 41, no. 5, pp. 1075–1081, May 2003.

[88] H. Iijima and Y. Naito, "Incremental prediction of the side-chain conformation of proteins by a genetic algorithm," in *Proc. IEEE Conf. Evolutionary Computation*, vol. 1, 1994, pp. 362–367.

[89] I. Ono, H. Fujiki, M. Ootsuka, N. Nakashima, N. Ono, and S. Tate, "Global optimization of protein 3-dimensional structures in NMR by a genetic algorithm," in *Proc. Congress Evolutionary Computation*, vol. 1, 2002, pp. 303–308.

[90] B. Contreras-Moreira, P. W. Fitzjohn, M. Offman, G. R. Smith, and P. A. Bates, "Novel use of a genetic algorithm for protein structure prediction: Searching template and sequence alignment space," *Proteins*, vol. 53, no. 6, pp. 424–429, 2003.

[91] P. Saxena, I. Whang, Y. Voziyanov, C. Harkey, P. Argos, M. Jayaram, and T. Dandekar, "Probing Flp: A new approach to analyze the structure of a DNA recognizing protein by combining the genetic algorithm, mutagenesis and non-canonical DNA target sites," *Biochem. Biophys. Acta.*, vol. 1340, no. 2, pp. 187–204, 1997.

[92] J. T. Pedersen and J. Moult, "Protein folding simulations with genetic algorithms and a detailed molecular description," *J. Mol. Biol.*, vol. 269, no. 2, pp. 240–259, 1997.

[93] M. Khimasia and P. Coveney, "Protein structure prediction as a hard optimization problem: The genetic algorithm approach," *Mol. Simul.*, vol. 19, pp. 205–226, 1997.

[94] R. Konig and T. Dandekar, "Improving genetic algorithms for protein folding simulations by systematic crossover," *BioSystems*, vol. 50, pp. 17–25, 1999.

[95] C. A. Del Carpio, "A parallel genetic algorithm for polypeptide three dimensional structure prediction: A transputer implementation," *J. Chem. Inf. Comput. Sci.*, vol. 36, no. 2, pp. 258–269, 1996.

[96] Rabow and H. A. Scheraga, "Improved genetic algorithm for the protein folding problem by use of a cartesian combination operator," *Protein Sci.*, vol. 5, pp. 1800–1815, 1996.

[97] J. R. Gunn, "Sampling protein conformations using segment libraries and a genetic algorithm," *J. Chemi. Phys.*, vol. 106, pp. 4270–4281, 1997.

[98] A. C. W. May and M. S. Johnson, "Improved genetic algorithm-based protein structure comparisons: Pairwise and multiple superpositions," *Protein Eng.*, vol. 8, pp. 873–882, 1995.

[99] M. J. Bayley, G. Jones, P. Willett, and M. P. Williamson, "Genfold: A genetic algorithm for folding protein structures using NMR restraints," *Protein Sci.*, vol. 7, no. 2, pp. 491–499, 1998.

[100] Z. Sun, X. Xia, Q. Guo, and D. Xu, "Protein structure prediction in a 210-type lattice model: Parameter optimization in the genetic algorithm using orthogonal array," *J. Protein Chem.*, vol. 18, no. 1, pp. 39–46, 1999.

[101] S. Schulze-Kremer, "Genetic algorithms and protein folding. Methods in molecular biology," *Protein Structure Prediction: Methods and Protocols*, vol. 143, pp. 175–222, 2000.

[102] A. M. Lesk, *Introduction to Bioinformatics*. London, U.K.: Oxford Univ. Press, 2002.

[103] Y. Xiao and D. Williams, "Genetic algorithms for docking of actinomycin D and deoxyguanosine molecules with comparison to the crystal structure of actinomycin ddeoxyguanosine complex," *J. Phys. Chem.*, vol. 98, pp. 7191–7200, 1994.

[104] D. R. Westhead, D. E. Clark, D. Frenkel, J. Li, C. W. Murray, B. Robson, and B. Waszkowycz, "PRO-LIGAND: An approach to de novo molecular design. 3. A genetic algorithm for structure refinement," *J. Comput.- Aided Mol. Design*, vol. 9, no. 2, pp. 139–148, 1995.

[105] G. M. Morris, D. S. Goodsell, R. S. Halliday, R. Huey, W. E. Hart, R. K. Belew, and A. J. Olsoni, "Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function," *J. Comput. Chem.*, vol. 19, no. 14, pp. 1639–1662, 1998.

[106] A. Bagchi, S. Bandyopadhyay, and U. Maulik, "Determination of molecular structure for drug design using variable string length genetic algorithm," in *Workshop on Soft Computing, High Performance Computing (HiPC) Workshops 2003: New Frontiersin High-Performance Computing,* Hyderabad, India, 2003, pp. 145–154.

[107] S. Bandyopadhyay, A. Bagchi, and U. Maulik, "Active site driven ligand design: An evolutionary approach," *J. Bioinf. Comput. Biol.*, vol. 3, no. 5, pp. 1053–1070, 2005.
[108] C. Chen, L. H. Wang, C. Kao, M. Ouhyoung, and W. Chen, "Molecular binding in structure-based drug design: A case study of the population-based annealing genetic algorithms," in *Proc. IEEE Int. Conf. Tools with Artificial Intelligence*, 1998, pp. 328–335.
[109] L. H. Wang, C. Kao, M. Ouh-Young, and W. Chen, "Molecular binding: A case study of the population-based annealing genetic algorithms," in *Proc. IEEE Int. Conf. Evolutionary Computation*, 1995, pp. 50–55.
[110] L. H. Wang, C. Kao, M. Ouh-Young, and W. C. Cheu, "Using an annealing genetic algorithm to solve global energy minimization problem in molecular binding," in *Proc. 6th Int. Conf. Tools with Artificial Intelligence*, 1994, pp. 404–410.
[111] C. D. Rosin, R. S. Halliday, W. E. Hart, and R. K. Belew, "A comparison of global and local search methods in drug docking," in *Proc. Int. Conf. Genetic Algorithms*, 1997, pp. 221–228.
[112] J. M. Yang and C. Y. Kao, "A family competition evolutionary algorithm for automated docking of flexible ligands to proteins," *IEEE Trans. Inf. Technol. Biomed.*, vol. 4, no. 3, pp. 225–237, Sep. 2000.
[113] C. M. Oshiro, I. D. Kuntz, and J. S. Dixon, "Flexible ligand docking using a genetic algorithm," *J. Comput.-Aided Mol. Design*, vol. 9, no. 2, pp. 113–130, 1995.
[114] D. E. Clark and D. R. Westhead, "Evolutionary algorithms in computer-aided molecular design," *J. Comput.-Aided Mol. Design*, vol. 10, no. 4, pp. 337–358, 1996.
[115] V. Venkatasubramanian, K. Chan, and J. Caruthers, "Computer aided molecular design using genetic algorithms," *Comput. Chem. Eng.*, vol. 18, no. 9, pp. 833–844, 1994.
[116] D. M. Deaven and K. O. Ho, "Molecular-geometry optimization with a genetic algorithm," *Phys. Rev. Lett.*, vol. 75, no. 2, pp. 288–291, 1995.
[117] G. Jones, P. Willett, and R. C. Glen, "Molecular recognition of receptor sites using a genetic algorithm with a description of desolvation," *J. Mol. Biol.*, vol. 245, pp. 43–53, 1995.
[118] G. Jones, P. Willett, R. C. Glen, A. R. Leach, and R. Taylor, "Further development of a genetic algorithm for ligand docking and its application to screening combinatorial libraries," *American Chemical Society Symposium Series*, vol. 719, Washington, DC: ACS, 1999, pp. 271–291.
[119] D. B. McGarrah and R. S. Judson, "Analysis of the genetic algorithm method of molecular conformation determination," *J. Comput. Chem.*, vol. 14, no. 11, pp. 1385–1395, 1993.
[120] T. Hou, J. Wang, L. Chen, and X. Xu, "Automated docking of peptides and proteins by using a genetic algorithm combined with a tabu search," *Protein Eng.*, vol. 12, pp. 639–647, 1999.
[121] P. Willet, "Genetic algorithms in molecular recognition and design," *Trends Biotechnol.*, vol. 13, no. 12, pp. 516–521, 1995.

Sankar K. Pal (M'81–SM'84–F'93) received the Ph.D. degree in radio physics and electronics from the University of Calcutta, Calcutta, India, in 1974, and the Ph.D. degree in electrical engineering along with DIC from Imperial College, University of London, London, U.K., in 1982.

He is the Director and Distinguished Scientist of the Indian Statistical Institute, Calcutta. He founded the Machine Intelligence Unit in 1993, and the Center for Soft Computing Research: A National Facility in 2004 at the Institute in Calcutta. He worked at the University of California, Berkeley and the University of Maryland, College Park, in 1986–1987; the NASA Johnson Space Center, Houston, TX, in 1990–1992 and 1994; and in the U.S. Naval Research Laboratory, Washington, DC, in 2004. He is a co-author of 13 books and about 300 research publications. Since 1997 he has been serving as a Distinguished Visitor of the IEEE Computer Society for the Asia-Pacific Region, and held several visiting positions in Hong Kong and Australian universities. He is an Associate Editor of *Pattern Recognition Letters, Neurocomputing, Applied Intelligence, Information Sciences, Fuzzy Sets and Systems, Fundamenta Informaticae and the International Journal of Computational Intelligence and Applications;*

Prof. Pal is a Fellow of the Third World Academy of Sciences, International Association for Pattern recognition, and all the four National Academies for Science/Engineering in India. He has received the 1990 S.S. Bhatnagar Prize (which is the most coveted award for a scientist in India), and many prestigious awards in India and abroad including the 1999 G. D. Birla Award, 1998 Om Bhasin Award, 1993 Jawaharlal Nehru Fellowship, 2000 Khwarizmi International Award from the Islamic Republic of Iran, 2000–2001 FICCI Award, 1993 Vikram Sarabhai Research Award, 1993 NASA Tech Brief Award, 1994 IEEE TRANSACTION NEURAL NETWORKS Outstanding Paper Award, 1995 NASA Patent Application Award, 1997 IETE-R. L. Wadhwa Gold Medal, and the 2001 INSA-S.H. Zaheer Medal. He is an Associate Editor of the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, and the IEEE TRANSACTIONS NEURAL NETWORKS He is a Member, Executive Advisory Editorial Board, IEEE TRANSACTIONS FUZZY SYSTEMS, *Int. Journal on Image and Graphics*, and *Int. Journal of Approximate Reasoning*; and a Guest Editor of *IEEE Computer.*

Sanghamitra Bandyopadhyay (SM'05) received the B.Sc. and B.Tech. degrees in physics and computer science in 1988 and 1992, respectively, the Masters degree in computer science from the Indian Institute of Technology (IIT), Kharagpur, in 1994, and the Ph.D. degree in computer science from the Indian Statistical Institute, Calcutta, in 1998.

Currently, she is an Associate Professor at the Indian Statistical Institute. She has worked for Los Alamos National Laboratory, in 1997, as a Graduate Research Assistant, in the University of New South Wales, in 1999, as a Post Doctoral Fellow, in the Department of Computer Science and Engineering, University of Texas at Arlington, in 2001 as Researcher, and in the Department of Computer Science and Engineering, University of Maryland, in 2004 as Visiting Research Faculty. Her research interests include evolutionary and soft computation, pattern recognition, data mining, bioinformatics, parallel and distributed systems and VLSI. She has published over 70 articles in international journals, conference, and workshop proceedings, edited books and journal special issues, and served on the committees of several conferences and workshops. She is on the editorial board of the *International Journal on Computational Intelligence*.

Dr. Bandyopadhyay is the first recipient of the Dr. Shanker Dayal Sharma Gold Medal and Institute Silver Medal for being the best all round post graduate performer in IIT, Kharagpur in 1994. She received the Indian National Science Academy and the Indian Science Congress Association Young Scientist Awards in 2000, as well as the Indian National Academy of Engineering Young Engineers' Award in 2002. She is serving as the Program Co-Chair of the 1st International Conference on Pattern Recognition and Machine Intelligence, 2005, to be held in Kolkata, India, and has served as the Tutorial Co-Chair, World Congress on Lateral Computing, 2004, held in Bangalore, India.

Shubhra Sankar Ray received the M.Sc. and M.Tech. degrees in electronic science and radiophysics and electronics from the University of Calcutta, Kolkata, India, in 2000 and 2002, respectively.

Since June 2003, he has been a Senior Research Fellow of the Council of Scientific and Industrial Research, New Delhi, India, working at the Machine Intelligence Unit, Indian Statistical Institute, Kolkata. His research interests include bioinformatics, evolutionary computation, neural networks, and data mining.

Bioinformatics in neurocomputing framework

S.S. Ray, S. Bandyopadhyay, P. Mitra and S.K. Pal

Abstract: Different bioinformatics tasks like gene sequence analysis, gene finding, protein structure prediction and analysis, gene expression with microarray analysis and gene regulatory network analysis are described along with some classical approaches. The relevance of intelligent systems and neural networks to these problems is mentioned. Different neural network based algorithms to address the aforesaid tasks are then presented. Finally some limitations of the current research activity are provided. An extensive bibliography is included.

1 Introduction

Over the past few decades, major advances in the field of molecular biology, coupled with advances in genomic technologies, have led to an explosive growth in the biological information generated by the scientific community. This deluge of genomic information has, in turn, led to an absolute requirement for computerised databases to store, organise and index the data, and for specialised tools to view and analyse the data.

Bioinformatics can be viewed as 'the use of computational methods to make biological discoveries' [1]. It is an interdisciplinary field involving biology, computer science, mathematics and statistics to analyse biological sequence data, genome content and arrangement, and to predict the function and structure of macromolecules. The ultimate goal of the field is to enable the discovery of new biological insights in addition to create a global perspective from which unifying principles in biology can be derived [2]. There are three important sub-disciplines within bioinformatics:

a) development of new algorithms and models to assess different relationships among the members of a large biological data set in a way that allows researchers to access existing information and to submit new information as they are produced;

b) analysis and interpretation of various types of data including nucleotide and amino acid sequences, protein domains, and protein structures; and

c) development and implementation of tools that enable efficient access and management of different types of information.

Artificial neural networks (ANN), a biologically inspired technology, are machinery for adaptation and curve fitting and are guided by the principles of biological neural networks. ANN have been studied for many years with the hope of achieving human like performance, particularly in the field of pattern recognition. They are efficient adaptive and robust classifiers, producing near optimal solutions and achieving high speed via massive parallelism. Therefore, the application of ANN for solving certain problems in bioinformatics, which need optimisation of computation requirements, and robust, fast and close approximate solutions, appears to be appropriate and natural. Moreover, the errors generated in experiments with bioinformatics data can be handled with the robust characteristics of ANN and minimised during the trainnig process. The problem of integrating ANN and bioinformatics constitutes a new research area.

This article provides a survey of the various neural network based techniques that have been developed over the past few years for different bioinformatics tasks.

2 Elements of bioinformatics

Deoxyribonucleic acid (DNA) and proteins are biological macromolecules built as long linear chains of chemical components. DNA strands consist of a large sequence of nucleotides, or bases. For example there are more than three billion bases in human DNA sequences. DNA plays a fundamental role in different biochemical processes of living organisms in two respects. First it contains the templates for the synthesis of proteins, which are essential molecules for any organism [3]. The second role in which DNA is essential to life is as a medium to transmit hereditary information (namely the building plans for proteins) from generation to generation.

The units of DNA are called nucleotides. One nucleotide consists of one nitrogen base, one sugar molecule (deoxyribose) and one phosphate. Four nitrogen bases are denoted by one of the letters A (adenine), C (cytosine), G (guanine) and T (thymine). A linear strand of DNA is paired to a complementary strand. The complementary property stems from the ability of the nucleotides to establish specific pairs (A–T and G–C). The pair of complementary strands then forms the double helix that was first suggested by Watson and Crick in 1953. Each strand therefore carries the entire information and the biochemical machinery guarantees that the information can be copied over and over again even when the 'original' molecule has long since vanished.

A gene is primarily made up of a sequence of triplets of the nucleotides (exons). Introns (non-coding sequence) may also be present within a gene. Not all portions of the DNA sequences are coding. A coding zone indicates that it is a

IEE Proceedings online no. 20045051

doi:10.1049/ip-cds:20045051

Paper first received 16th June 2004 and in final revised form 5th October 2004

The authors are with Machine Intelligence Unit, Indian Statistical Institute, Kolkata 700108, India

E-mail: shubhra_r@isical.ac.in

template for a protein. As an example, for the human genome only 3–5% of the portions are coding, i.e., they constitute the gene. There are sequences of nucleotides within the DNA that are spliced out progressively in the process of transcription and translation. In brief, the DNA consists of three types of non-coding sequences (shown schematically in Fig. 1).

1. Intergenic regions: regions between genes that are ignored during the process of transcription.
2. Intragenic regions (or introns): regions within the genes that are spliced out from the transcribed RNA to yield the building blocks of the genes, referred to as exons.
3. Pseudogenes: genes that are transcribed into the RNA and stay there, without being translated, owing to the action of a nucleotide sequence.

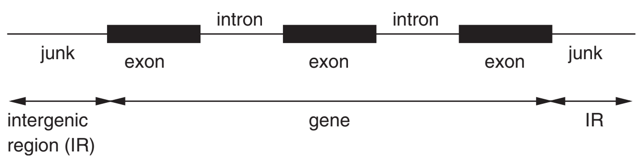


Fig. 1 *Various parts of DNA*

Proteins are made up of 20 different amino acids (or 'residues'), which are denoted by 20 different letters of the alphabet. Each of the 20 amino acids is coded by one or more triplets (or codons) of the nucleotides making up the DNA. Based on the genetic code the linear string of DNA is translated into a linear string of amino acids, i.e., a protein via mRNA [3].

3 Bioinformatics tasks

The different biological problems studied within the scope of bioinformatics can be broadly classified into two categories: genomics and proteomics which include genes, proteins, and amino acids. We describe below different tasks involved in their analysis along with their utilities.

3.1 Gene sequence analysis

The evolutionary basis of sequence alignment is based on the principles of similarity and homology [4]. Similarity is a quantitative measure of the fraction of two genes which are identical in terms of observable quantities. Homology is the conclusion drawn from data that two genes share a common evolutionary history; no metric is associated with this. The tasks of sequence analysis are as follows.

3.1.1 Sequence alignment: An alignment is a mutual arrangement of two or more sequences, that exhibits where the sequences are similar, and where they differ. An optimal alignment is one that exhibits the most correspondences and the least differences. It is the alignment with the highest score but may or may not be biologically meaningful. Basically there are two types of alignment methods, global alignment and local alignment. Global alignment [5] maximises the number of matches between the sequences along the entire length of the sequence. Local alignment [6] gives a highest scoring to local match between two sequences.

3.1.2 Pattern searching: This deals with searches for a nucleic pattern in a nucleic acid sequence, in a set of sequences or in a databank (e.g. INFO-BIOGEN) [7]. It is the potential for uncovering evolutionary relationships and patterns between different forms of life. With the aid of nucleotide and protein sequences, it should be possible to find the ancestral ties between different organisms. So far, experience indicates that closely related organisms have similar sequences and that more distantly related organisms have more dissimilar sequences. Proteins that show a significant sequence conservation indicating a clear evolutionary relationship are said to be from the same protein family. By studying protein folds (distinct protein building blocks) and families, scientists are able to reconstruct the evolutionary relationship between two species and to estimate the time of divergence between two organisms since they last shared a common ancestor.

3.1.3 Gene finding and promoter identification: In general a DNA strand consists of a large sequence of nucleotides, or bases. Owing to the huge size of the database, manual searching of genes, which code for proteins, is not practical. Therefore automatic identification of the genes from the large DNA sequences is an important problem in bioinformatics [8]. A cell mechanism recognises the beginning of a gene or gene cluster with the help of a promoter. The promoter is a region before each gene in the DNA that serves as an indication to the cellular mechanism that a gene is ahead. For example, the codon AUG (which codes for methionine) also signals the start of a gene. Recognition of regulatory sites in DNA fragments has become particularly popular because of the increasing number of completely sequenced genomes and mass application of DNA chips.

Promoters are key regulatory sequences that are necessary for the initiation of transcription. Experimental analysis has identified fewer than 10% of the potential promoter regions, assuming that there are at least 30,000 promoters in the human genome, one for each gene. On a genome-wide scale, pattern-based and genomic context-based computational approaches can suggest possible transcription factor-binding regions, but the rate of false-positive predictions is very high.

3.2 Protein analysis

Proteins are polypeptides, formed within cells as a linear chain of amino acids [9]. Within and outside of cells, proteins serve a myriad of functions, including structural roles (cytoskeleton), as catalysts (enzymes), transporters to ferry ions and molecules across membranes, and hormones to name just a few. There are twenty different amino acids that make up essentially all proteins on earth. Different tasks involved in protein analysis are as follows.

3.2.1 Multiple sequence alignment: Multiple amino acid sequence alignment techniques [1] are usually performed to fit one of the following scopes: (a) finding the consensus sequence of several aligned sequences; (b) helping in the prediction of the secondary and tertiary structures of new sequences; and (c) providing a preliminary step in molecular evolution analysis using phylogenetic methods for constructing phylogenetic trees.

In order to characterise protein families, one needs to identify shared regions of homology in a multiple sequence alignment; (this happens generally when a sequence search reveals homologies in several sequences). The clustering method can do alignments automatically but is subjected to some restrictions. Manual and eye validations are necessary in some difficult cases. The most practical and widely used method in multiple sequence alignment is the hierarchical extensions of pairwise alignment methods, where the principal is that multiple alignments are achieved by successive applications of pairwise methods.

3.2.2 Protein motif search:

A protein motif search [8] allows searching for a personal protein pattern in a sequence (personal sequence or an entry in a gene bank). Proteins are derived from a limited number of basic building blocks (domains). Evolution has shuffled these modules giving rise to a diverse repertoire of protein sequences, as a result proteins can share a global or local relationship. Protein motif search is a technique for searching sequence databases to uncover common domains/motifs of biological significance that categorise a protein into a family.

3.2.3 Structural genomics:

Structural genomics is the prediction of the 3-dimensional structure of a protein from the primary amino acid sequence [10]. This is one of the most challenging tasks in bioinformatics. The four levels of protein structure (Fig. 2) are

(a) primary structure: the sequence of amino acids that compose the protein,

(b) secondary structure: the spatial arrangement of the atoms constituting the main protein backbone, such as alpha helices and beta strands,

(c) tertiary structure: formed by packing secondary structural elements into one or several compact globular units called domains, and

(d) final protein may contain several polypeptide chains arranged in a quaternary structure.

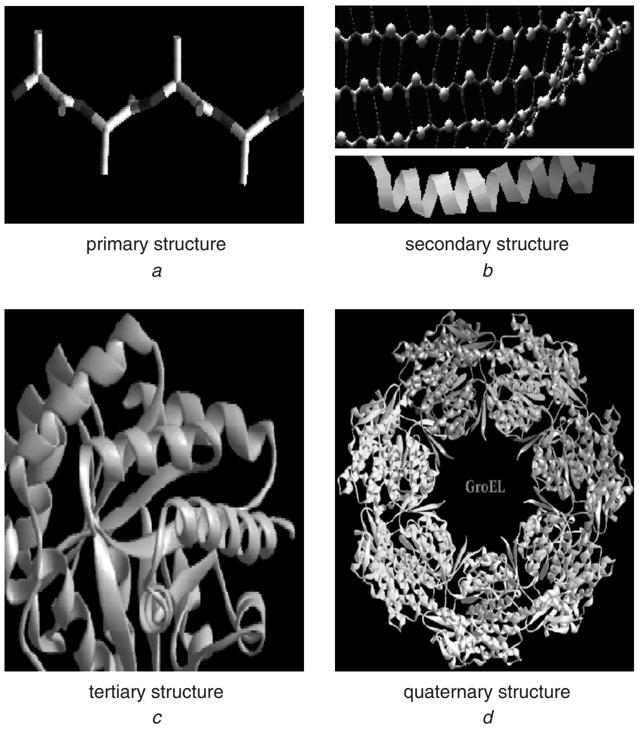


Fig. 2 *Different levels of protein structures*

Sequence similarity methods can predict the secondary and tertiary structures based on homology to known proteins. Secondary structure prediction can be made using Chou–Fasman [10], GOR, neural network, and nearest neighbour methods. Methods for tertiary structure prediction involve energy minimisation, molecular dynamics, and stochastic searches of conformational space.

3.3 Gene expression and microarrays

Gene expression is the process by which a gene's coded information is converted into the structures present and operating in the cell. Expressed genes include those that are transcribed into mRNA and then translated into protein and those that are transcribed into RNA but not translated into protein (e.g., transfer and ribosomal RNA). Not all genes are expressed and gene expression involves the study of the expression level of genes in the cells under different conditions. Conventional wisdom is that gene products that interact with each other are more likely to have similar expression profiles than if they do not [11].

Microarray technology [12] allows expression levels of thousands of genes to be measured at the same time. Comparison of gene expression between normal and diseased (e.g., cancerous) cells are also done by microarray. There are several names for this technology for example DNA microarrays, DNA arrays, DNA chips, gene chips. A microarray is typically a glass (or some other material) slide, on to which DNA molecules are attached at fixed locations (spots). There may be tens of thousands of spots on an array, each containing a huge number of identical DNA molecules (or fragments of identical molecules), of lengths from twenty to hundreds of nucleotides. For gene expression studies, each of these molecules ideally should identify one gene or one exon in the genome, however, in practice this is not always so simple and may not even be generally possible owing to families of similar genes in a genome. The spots are either printed on the microarrays by a robot, or synthesised by photolithography (similar to computer chip production) or by ink-jet printing.

Many unanswered, and important, questions could potentially be answered by correctly selecting, assembling, analysing, and interpreting microarray data. Clustering is commonly used in microarray experiments to identify groups of genes that share similar expressions. Genes that are similarly expressed are often co-regulated and involved in the same cellular processes. Therefore, clustering suggests functional relationships between groups of genes. It may also help in identifying promoter sequence elements that are shared among genes. In addition, clustering can be used to analyse the effects of specific changes in experimental conditions and may reveal the full cellular responses triggered by those conditions.

3.4 Gene regulatory network analysis

Another important and interesting question in biology is how gene expression is switched on and off, i.e., how genes are regulated [1]. Since almost all cells in a particular organism have an identical genome, differences in gene expression and not the genome content are responsible for cell differentiation (how different cell types develop from a fertilised egg) during the life of the organism.

Gene regulation in eukaryotes, is not well understood, but there is evidence that an important role is played by a type of proteins called transcription factors. The transcription factors can attach (bind) to specific parts of the DNA, called transcription factor binding sites (i.e., specific, relatively short combinations of A, T, C or G), which are located in so-called promoter regions. Specific promoters are associated with particular genes and are generally not too far from the respective genes, though some regulatory effects can be located as far as 30,000 bases away, which makes the definition of the promoter difficult.

Transcription factors control gene expression by binding the gene's promoter and either activating (switching on) the gene's transcription, or repressing it (switching it off). Transcription factors are gene products themselves, and therefore, in turn, can be controlled by other transcription factors. Transcription factors can control many genes, and some (probably most) genes are controlled by combinations of transcription factors. Feedback loops are possible. Therefore we can talk about gene regulation networks. The understanding, describing and modelling of such gene regulation networks is one of the most challenging problems in functional genomics. Microarrays and computational methods are playing a major role in attempts to reverse engineer gene networks from various observations. Note that in reality the gene regulation is likely to be a stochastic and not a deterministic process. Traditionally molecular biology has followed a so-called reductionist approach mostly concentrating on a study of a single or very few genes in any particular research project. With genomes being sequenced, this is now changing into a so-called systems approach.

4 Relevance of neural networks in bioinformatics

Artificial neural network (ANN) models try to emulate the biological neural network with electronic circuitry. Recently, ANNs have found widespread use for classification tasks and function approximation in many fields of medicinal chemistry and bioinformatics. For these kinds of data analysis mainly two types of networks are employed; the 'supervised' neural network (SNN) and the 'unsupervised' neural network (UNN). The main applications of SNNs (e.g. multilayer perceptrons (MLPs) are feedforward neural networks trained with the standard backpropagation algorithm) are function approximation, classification, pattern recognition and feature extraction, and prediction. Moreover, they are able to detect second and higher order correlations in patterns. This is specially important in biological systems, which frequently display nonlinear behaviour. These networks require a set of molecular compounds with known activities to model structure-activity relationships and are able to determine the relevant features in the data set, usually by means of training processes. This principle coined the term 'supervised' networks. Correspondingly, 'unsupervised' networks (e.g. Kohonen self-organising maps) can be applied to clustering and feature extraction tasks even without prior knowledge of molecular activities or properties. Unsupervised learning has the advantage that no previous knowledge about the system under study is required.

The main characteristics of ANNs are:

a) adaptability to new data/environment,

b) robustness/ruggedness to failure of components,

c) speed via massive parallelism, and

d) optimality w.r.t. error.

Let us now explain the functioning of an ANN in bioinformatics with an example of protein secondary structure prediction from a linear sequence of amino acids (Fig. 3).

Step 1: In the ANN usually a certain number of input 'nodes' are each connected to every node in a hidden layer.

Step 2: Every residue in a protein data bank (PDB) entry can be associated to one of the three secondary structures (helix, sheet or neither: coil). ANNs are designed with 21 input nodes (one for each residue including a null residue) and three output nodes coding for each of the three possible secondary structure assignments (helix, sheet and coil).

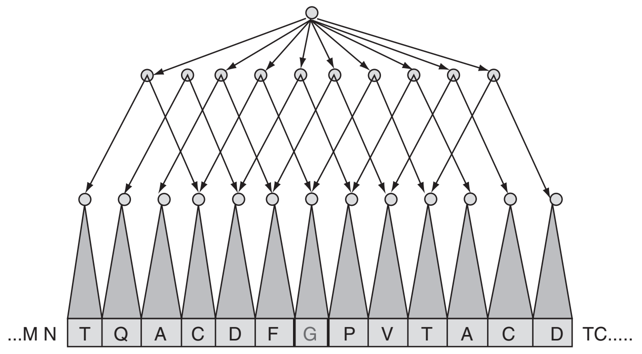


Fig. 3 *A linear chain of amino acids is applied as input to the ANN*

Step 3: Each node in the hidden layer is then connected to every node in the final output layer.

Step 4: The input and output nodes are restricted to binary values (1 or 0) when loading the data onto the network during training and the weights are then modified by the program itself during the training process.

Step 5: Helix can be coded as 0, 0, 1 on the three output nodes; sheet can be coded as 0, 1, 0 and coil as 1, 0, 0. A similar binary coding scheme can be used for the 20 input nodes for the 20 amino acids.

Step 6: To consider a moving window of n residues at a time, input layer should contain $20 \times n$ nodes plus one node at each position for a null residue.

Step 7: Each node will 'decide' to send a signal to the nodes it is connected to, based on evaluating its transfer function after all of its inputs and connection weights have been summed.

Step 8: Over 100 protein structures were used to train the network.

Step 9: Training proceeds by holding a particular data constant onto both the input and output nodes and iterating the network in a process that modifies the connection weights until the changes made to them approach zero.

Step 10: When such convergence is reached, the network is said to be trained and is ready to receive new (unknown) experimental data.

Step 11: Now the connection weights are not changed and the values of the hidden and output nodes are calculated in order to determine the structure of the input sequence of proteins.

Selection of unbiased and normalised training data, however, is probably just as important as the network architecture in the design of a successful NN.

5 Anns in bioinformatics

Let us now describe the different attempts made using ANNs in certain tasks of bioinformatics in the broad domains of sequence analysis, structure prediction, and gene analysis described in Section 3.

5.1 Sequence alignment

Given inputs extracted from an aligned column of DNA bases and the underlying Perkin Elmer Applied Biosystems (ABI) fluorescent traces, Allex *et al.* [13] trained a neural network to determine correctly the consensus base for the column. They compared five representations empirically; one uses only base calls and the others include trace information. The networks that incorporate trace information into their input representations attained the most accurate results for consensus sequence. Consensus accuracies ranging from 99.26% to 99.98% are acheived for coverages from two to six aligned sequences. In contrast, the network that only uses base calls in its input representation has over double that error rate.

In [14] a molecular alignment method with the Hopfield neural network (HNN) is discussed. Molecules are represented by four kinds of chemical properties (hydrophobic group, hydrogen-bonding acceptor, hydrogen-bonding donor, and hydrogen-bonding donor/acceptor), and then those properties between two molecules correspond to each other using HNN. The method is applied to three-dimensional quantitative structure-activity relationship (3D-QSAR) analysis and it reproduced successfully the real molecular alignments obtained from X-ray crystallography.

GenTHREADER is a neural network architecture that predicts similarity between gene sequences [15]. The effects of sequence alignment score and pairwise potential are the network outputs. GenTHREADER was used successfully for the structure prediction in two cases: case 1: ORF MG276 from *Mycoplasma genitalium* was predicted to share structure similarity with 1HGX; case 2: MG276 shares a low sequence similarity (10% sequence identity) with 1HGX.

A back-propagation neural network can grossly approximate the score function of the popular BLAST family of genomic sequence alignment and scoring tools. The resultant neural network may provide a processing speed advantage over the BLAST tool, but may suffer somewhat in comparison to the accuracy of BLAST. Further study is necessary to determine whether a neural network with additional hidden units or structural complexity could be used to more closely approximate BLAST. However, closer approximation may also limit the speed performance advantages enjoyed by the neural network approach.

Other related investigations in sequence analysis are available in [16, 17].

5.2 Gene finding and promoter identification

The application of artificial neural networks for discriminating the coding system of eukaryotic genes is investigated in [18]. Over 300 genes from eight eukaryotic organisms are chosen: human, mouse, rat, horse, ox, sheep, soybean and rabbit. From these genes different discrimination models are build which are relevant to genes promoter regions, poly(A) signals, splice site locations of introns and noose structures. The results showed that as long as the coding length is definite, the only correct coding region can be chosen from the large number of possible solutions discriminated by neural networks.

In [19] the quantitative similarity among tRNA gene sequences was acquired by analysis with an artificial neural network. The evolutionary relationship derived from ANN results was consistent with those from other methods. A new sequence was recognised to be a tRNA-like gene by a neural network on the analysis of similarity.

The work of Lukashin *et al.* [20] is one of the earlier investigations that discussed the problem of recognition of promoter sites in the DNA sequence in a neural network framework. The learning process involves a small (of the order of 10%) part of the total set of promoter sequences. During this procedure the neural network develops a system of distinctive features (key words) to be used as a reference in identifying promoters against the background of random sequences. The learning quality is then tested with the whole set. The efficiency of promoter recognition has been reported as 94 to 99% and the probability of an arbitrary sequence being identified as a promoter is 2 to 6%.

In [21] a multilayered feed-forward ANN architecture is trained for predicting whether a given nucleotide sequence is a mycobacterial promoter sequence. The ANN is used in conjunction with the caliper randomisation (CR) approach for determining the structurally/functionally important regions in the promoter sequences. This work shows that ANNs are efficient tools for predicting mycobacterial promoter sequences and determining structurally/functionally important sub-regions therein.

Other related investigations in promoter identification are available in [22, 23].

5.3 Protein analysis

The most successful techniques for prediction of the three-dimensional structure of protein rely on aligning the sequence of a protein of unknown structure to a homologue of known structure. Such methods fail if there is no homologue in the structural database, or if the technique for searching the structural database is unable to identify homologues that are present.

The work of Qian *et al.* [24] is one of the earlier investigations that discussed the protein structure prediction problem in a neural network framework. They used X-ray-derived crystal structures of globular proteins available at that time to train a NN to predict the secondary structure of non-homologous proteins. Over 100 protein structures were used to train this network. After training, when the NN was queried with new data, a prediction accuracy of 64% was obtained.

Rost *et al.* [25, 26] took advantage of the fact that a multiple sequence alignment contains more information about a protein than the primary sequence alone. Instead of using a single sequence as input into the network, they used a sequence profile that resulted from the multiple alignments. This resulted in a significant improvement in prediction accuracy to 71.4%. Recently, more radical changes to the design of NNs including bi-directional training and the use of the entire protein sequence as simultaneous input instead of a shifting window of fixed length has led to prediction accuracy above 71%.

The prediction of protein secondary structure using structured neural networks and multiple sequence alignments have been investigated by Riis and Krogh [27]. Separate networks are used for predicting the three secondary structures, ff-helix, fi-strand and coil. The networks are designed using *a priori* knowledge of amino acid properties with respect to the secondary structure and of the characteristic periodicity in ff-helices. This method gives an overall prediction accuracy of 66.3% when using seven-fold cross-validation on a database of 126 non-homologous globular proteins. Applying the method to multiple sequence alignments of homologous proteins increases the prediction accuracy significantly to 71.3% [27].

In [28] a method has been developed using ANNs for the prediction of beta-turn types I, II, IV and VIII. For each turn type, two consecutive feed-forward back-propagation networks with a single hidden layer have been used. The first sequence-to-structure network has been trained on single sequences in addition to on PSI-BLAST PSSM. The output from the first network along with PSIPRED [29] predicted secondary structure has been used as input for the second-level structure-to-structure network. The networks have been trained and tested on a non-homologous data set of 426 proteins chains by seven-fold cross-validation. The prediction performance for each turn type is improved by using multiple sequence alignment, second level structure-to-structure network and PSIPRED predicted secondary structure information.

The back-propagation neural network algorithm is a commonly used method for predicting the secondary structure of proteins. Wood *et al.* [30] compared the cascade-correlation ANN architecture [31] with the back-propagation ANN using a constructive algorithm and found that cascade-correlation achieves predictive accuracies comparable to those obtained by back-propagation, in shorter time. Ding *et al.* [32] used support vector machine (SVM) and the neural network (NN) learning methods as base classifiers for protein fold recognition, without relying on sequence similarity.

Other related investigations in protein structure prediction are available in [33–38].

5.4 Gene expression and microarray

Clustering is commonly used in microarray experiments to identify groups of genes that share similar expression. Genes that are similarly expressed are often co-regulated and involved in the same cellular processes. Therefore, clustering suggests functional relationships between groups of genes. It may also help in identifying promoter sequence elements that are shared among genes. In addition, clustering can be used to analyse the effects of specific changes in experimental conditions and may reveal the full cellular responses triggered by those conditions.

Most of the analysis of the enormous amount of information provided on microarray chips with regard to cancer patient prognosis has relied on clustering techniques and other standard statistical procedures. These methods are inadequate in providing the reduced gene subsets required for perfect classification. ANNs trained on microarray data from DLBCL lymphoma patients have, for the first time, been able to predict the long-term survival of individual patients with 100% accuracy [39]. Here it is shown that differentiating the trained network can narrow the gene profile to less than three dozen genes for each classification and artificial neural networks are superior tools for digesting microarray data.

Sawa *et al.* [40] described a neural network-based similarity index as a nonlinear similarity index and compared the results with other proximity measures for *Saccharomyces cerevisiae* gene expression data. Here it is shown that the clusters obtained using Euclidean distance, correlation coefficients, and mutual information were not significantly different. The clusters formed with the neural network-based index were more in agreement with those defined by functional categories and common regulatory motifs.

Diffuse large B-cell lymphoma (DLBCL) is the largest category of aggressive lymphomas. Less than 50% of patients can be cured by combination chemotherapy. Microarray technologies have recently shown that the response to chemotherapy reflects the molecular heterogeneity in DLBCL. On the basis of published microarray data, Ando *et al.* [41] described a fuzzy neural network (FNN) model to analyse gene expression profiling data for the precise and simple prediction of survival of DLBCL patients. From data on 5857 genes, this model identified four genes (CD10, AA807551, AA805611 and IRF-4) that could be used to predict prognosis with 93% accuracy. FNNs are powerful tools for extracting significant biological markers affecting prognosis, and are applicable to various kinds of expression profiling data for any malignancy.

Bicciato *et al.* [42] described a computational procedure for pattern identification, feature extraction, and classification of gene expression data through the analysis of an autoassociative neural network model. The identified patterns and features contain critical information about gene-phenotype relationships observed during changes in cell physiology. The methodology has been tested on two different microarray datasets, acute human leukemia and the human colon adenocarcinoma.

The Bayesian neural network is used with structural learning with forgetting for searching optimal network size and structure of microarray data in order to capture the structural information of gene expressions [43]. The process of Bayesian learning starts with a feed forward neural network (FFNN) and prior distribution for the network parameters. The prior distribution gives initial beliefs about the parameters before any data is observed. After new data are observed, the prior distribution is updated to the posterior distribution using Bayes rules. Multi-layer perceptron (MLP) is mainly considered as the network structure for Bayesian learning. Since the correlated data may include high levels of noise, efficient regularisation techniques are required to improve the generalisation performance. This involves network complexity adjustment and performance function modification. To do the latter, instead of the sum of squared error (SSE) on the training set, a cost function is automatically adjusted.

Vohradsky [44] used artificial neural networks as models of the dynamics of gene expression. The significance of the regulatory effect of one gene product on the expression of other genes of the system is defined by a weight matrix. The model considers multigenic regulation including positive and/or negative feedback. The process of gene expression is described by a single network and by two linked networks where transcription and translation are modelled independently. Each of these processes is described by different networks controlled by different weight matrices. Methods for computing the parameters of the model from experimental data are also shown.

Plausible neural network (PLANN) is another universal data analysis tool based upon artificial neural networks and is capable of plausible inference and incremental learning [45]. This tool has been applied to research data from molecular biological systems through the simultaneous analysis of gene expression data and other types of biological information.

Relevant investigations for gene expression and microarray are also available in [46].

5.5 Gene regulatory network

Adaptive double self-organising map (ADSOM) [47] provides a novel clustering technique for identifying gene regulatory networks. It has a flexible topology and it performs clustering and cluster visualisation simultaneously, thereby requiring no *a priori* knowledge about the number of clusters. ADSOM is developed based on a recently introduced technique known as double self-organising map (DSOM). DSOM combines features of the popular self-organising map (SOM) with two-dimensional position vectors, which serve as a visualisation tool to decide how many clusters are needed. Although DSOM addresses the problem of identifying unknown number of clusters, its free parameters are difficult to control to guarantee correct results and convergence. ADSOM updates its free parameters during training and it allows convergence of its position vectors to a fairly consistent number of clusters provided that its initial number of nodes is greater than the expected number of clusters. The number of clusters can be identified by visually counting the clusters formed by the position vectors after training. The reliance of ADSOM in identifying the number of clusters is proven by applying it to publicly available gene expression data from multiple biological systems such as yeast, human, and mouse. It may be noted that gene regulatory network analysis is a very recent research area, and neural network applications to it are scarce.

Appropriate definition of neural network architecture prior to data analysis is crucial for successful data mining. This can be challenging when the underlying model of the data is unknown. Using simulated data, Ritchie *et al.* [48] optimised back-propagation neural network architecture using genetic programming to improve the ability of neural networks to model, identify, characterise and detect nonlinear gene-gene interactions in studies of common human diseases. They showed that the genetic programming optimised neural network is superior to the traditional back-propagation neural network approach in terms of predictive ability and power to detect gene-gene interactions when non-functional polymorphisms are present.

6 Other bioinformatics tasks using ANNs

Dopazo *et al.* [49] described a new type of unsupervised growing self-organising neural network that expands itself following the taxonomic relationships existing among the sequences being classified. The binary tree topology of this neural network, opposite to other more classical neural network topologies, permits an efficient classification of sequences. The growing nature of this procedure allows to stop it at the desired taxonomic level without the necessity of waiting until a complete phylogenetic tree is produced. The time for convergence is approximately a linear function of the number of sequences. This neural network methodology is an excellent tool for the phylogenetic analysis of a large number of sequences.

Parbhane *et al.* [50] utilise an artificial neural network (ANN) for the prediction of DNA curvature in terms of retardation anomaly. The ANN captured the phase information and increased helix flexibility. Base pair effects in determining the extent of DNA curvature has been developed. The network predictions validate the known experimental results and also explain how the base pairs affect the curvature. The results suggest that ANN can be used as a model-free tool for studying DNA curvature.

Drug resistance is a very important factor influencing the failure of current HIV therapies. The ability to predict the drug resistance of HIV protease mutants may be useful in developing more effective and longer lasting treatment regimens. The HIV resistance is predicted to two current protease inhibitors, Indinavir and Saquinavir. This problem is handled in [51] from two perspectives. First, a predictor was constructed based on the structural features of the HIV protease-drug inhibitor complex. A particular structure was represented by its list of contacts between the inhibitor and the protease. Next, a classifier was constructed based on the sequence data of various drug resistant mutants. In both cases, SOMs were first used to extract the important features and cluster the patterns in an unsupervised manner. This was followed by subsequent labelling based on the known patterns in the training set. The classifier using the structure information is able to correctly recognise the previously unseen mutants with an accuracy of between 60 and 70%. The method is superior to a random classifier.

In [52] an ANN is trained to predict the sequence of the human TP53 tumor suppressor gene based on a p53 GeneChip. The trained neural network uses as input the fluorescence intensities of DNA hybridised to oligonucleotides on the surface of the chip. In this methodology errors are reported between zero and four in the predicted 1300 bp sequence when tested on wild-type TP53 sequence.

Neural network computations on DNA and RNA sequences are used in [53] to demonstrate that data compression is possible in these sequences. The result implies that a certain discrimination should be achievable between structured and random regions. The technique is illustrated by computing the compressibility of short RNA sequences such as tRNA.

A basic description of artificial neural networks and applications of neural nets to problems in human gene finding for three different types of data are discussed in [54].

7 Conclusion and scope of future research

Artificial neural networks (ANNs) are the first group of machine learning algorithms to be used on a biological pattern recognition problem. The rationale for applying computational approaches to facilitate the understanding of various biological processes are mainly:

- To provide a more global perspective in experimental design.
- To capitalise on the emerging technology of database-mining – the process by which testable hypotheses are generated regarding the function or structure of a gene or protein of interest by identifying similar sequences in better characterised organisms.

Neural networks appear to be a very powerful artificial intelligence (AI) paradigm to handle these issues [55]. The most important, and attractive, feature of ANNs is their capability of learning (generalising) from example (extracting knowledge from data). This feature makes the ANN an attractive choice for bioinformatics tasks. The combination of backpropagation learning algorithm and the feed-forward, layered networks have been applied to virtually all pattern recognition problems (like sequence analysis, protein analysis, gene finding) in bioinformatics. The reason for this is the simplicity of the algorithm, and the vast body of research that has studied these networks. Although these networks are theoretically capable of separating a problem space into appropriate classes irrespective of the complexity of the separation boundaries, one of the classical disadvantages of these networks is that a certain amount of a priori knowledge is required in order to build a useful network. A crucial factor in training a useful network is its size (number of layers, size of layers, and number of synaptic connections). In many cases, it takes a large number of simulations before a close-to-optimum size of the network is found. If the network is designed to be larger than this optimum size, it will memorise (also called over-fit) the data rather than generalising and extracting knowledge. If the network is chosen to be smaller than the optimum size, the network will never learn the entire task at hand. However, there have been several reports dealing with the determination of an appropriate size of a network for a particular task.

Let us consider self-organising map (SOM), as an example, which has been widely used in mining biological data. SOM has the distinct advantage that they allow a priori knowledge to be included in the clustering process and well suited for analysing patterns (e.g., microarray data). They are ideally suited to exploratory data analysis, allowing one to impose partial structure on the clusters (in contrast to the rigid structure of hierarchical clustering, the strong prior hypotheses used in Bayesian clustering, and the nonstructure of k-means clustering) facilitating easy visualisation and interpretation. SOMs have good computational properties and are easy to implement, reasonably fast, and are scalable to large data sets. The most prominent disadvantage of the SOM-based approach is that it is difficult to know when to stop the algorithm and it may get stuck to a local minima, so the map is allowed to grow indefinitely to a point where clearly different sets of patterns are identified.

Other soft computing tools, like fuzzy set theory and genetic algorithms, integrated with ANN [56] may also be used; based on the principles of case based reasoning [57]. Even though the current approaches in biocomputing are very helpful in identifying patterns and functions of proteins and genes, they are still far from being perfect. They are not only time-consuming, requiring Unix workstations to run on, but might also lead to false interpretations and assumptions due to necessary simplifications. It is therefore still mandatory to use biological reasoning and common sense in evaluating the results delivered by a biocomputing program. Also, for evaluation of the trustworthiness of the output of a program it is necessary to understand the mathematical/theoretical background of it to finally come up with a useful and senseful analysis.

8 Acknowledgment

This work is partly supported by grant 22(0346)/02/EMR-II of the Council of Scientific and Industrial Research (CSIR), New Delhi, under the project “Knowledge Based Connectionist Data Mining System: Design and Application”.

9 References

1 Baldi, P., and Brunak, S.: ‘Bioinformatics: the machine learning approach’ (MIT Press, Cambridge, MA, 1998)
2 Altman, R.B., Valencia, A., Miyano, S., and Ranganathan, S.: ‘Challenges for intelligent systems in biology’, *IEEE Intell. Syst.*, 2001, **16**, (6), pp. 14–20
3 Setubal, J., and Meidanis, J.: ‘Introduction to computational molecular biology’ (International Thomson Publishing, Boston, MA, 1999)
4 Nash, H., Blair, D., and Grefenstette, J.: ‘Comparing algorithms for large-scale sequence analysis’. Proc. 2nd IEEE Int. Symp. on Bioinformatics and Bioengineering (BIBE’01), 2001, pp. 89–96
5 Needleman, S.B., and Wunsch, C.D.: ‘A general method applicable to the search for similarities in the amino acid sequence of two proteins’, *J. Mol. Biol.*, 1970, **48**, pp. 443–453
6 Smith, T.F., and Waterman, M.S.: ‘Identification of common molecular sequences’, *J. Mol. Biol.*, 1981, **147**, pp. 195–197
7 Gautheret, D., Major, F., and Cedergren, R.: ‘Pattern searching/alignment with RNA primary and secondary structures: an effective descriptor for tRNA’, *Comp. Appl. Biosci.*, 1990, **6**, pp. 325–331
8 Fickett, J.W.: ‘Finding genes by computer: the state of the art’, *Trends Genet.*, 1996, **12**, (8), pp. 316–320
9 Salzberg, S.L., Searls, D.B., and Kasif, S.: ‘Computational methods in molecular biology’ (Elsevier Science, Amsterdam, 1998)
10 Chou, P., and Fasmann, G.: ‘Prediction of the secondary structure of proteins from their amino acid sequence’, *Adv. Enzym.*, 1978, **47**, pp. 145–148
11 Luscombe, N.M., Greenbaum, D., and Gerstein, M.: ‘What is bioinformatics? A proposed definition and overview of the field’, *Methods Informat. Med.*, 2001, **40**, (4), pp. 346–358
12 Quackenbush, J.: ‘Computational analysis of microarray data’, *Nat. Rev. Genetics*, 2001, **2**, pp. 418–427
13 Allex, C.F., Shavlik, J.W., and Blattner, F.R.: ‘Neural network input representations that produce accurate consensus sequences from DNA fragment assemblies’, *Bioinformatics*, 1999, **15**, (9), pp. 723–728
14 Arakawa, M., Hasegawa, K., and Funatsu, K.: ‘Application of the novel molecular alignment method using the Hopfield neural network to 3D-QSAR’, *J. Chem. Inf. Comput. Sci.*, 2003, **43**, (5), pp. 1396–1402
15 Jones, D.T.: ‘GenTHREADER: an efficient and reliable protein fold recognition method for genomic sequences’, *J. Mol. Biol.*, 1999, **287**, pp. 797–815
16 Hirst, J.D., and Sternberg, M.J.: ‘Prediction of structural and functional features of protein and nucleic acid sequences by artificial neural networks’, *Biochemistry*, 1992, **31**, (32), pp. 7211–7218
17 Petersen, S.B., Bohr, H., Bohr, J., Brunak, S., Cotterill, R.M., Fredholm, H., and Lautrup, B.: ‘Training neural networks to analyse biological sequences’, *Trends Biotechnol.*, 1990, **8**, (11), pp. 304–308
18 Cai, Y., and Chen, C.: ‘Artificial neural network method for discriminating coding regions of eukaryotic genes’, *Comput. Appl. Biosci.*, 1995, **11**, (5), pp. 497–501
19 Sun, J., Song, W.Y., Zhu, L.H., and Chen, R.S.: ‘Analysis of tRNA gene sequences by neural network’, *J. Comput. Biol.*, 1995, **2**, (3), pp. 409–416
20 Lukashin, A.V., Anshelevich, V.V., Amirikyan, B.R., Gragerov, A.I., and Frank-Kamenetskii, M.D.: ‘Neural network models for promoter recognition’, *J. Biomol. Struct. Dyn.*, 1989, **6**, (6), pp. 1123–1133
21 Kalate, R.N., Tambe, S.S., and Kulkarni, B.D.: ‘Artificial neural networks for prediction of mycobacterial promoter sequences’, *Comput. Biol. Chem.*, 2003, **27**, (6), pp. 555–564
22 Reese, M.G.: ‘Application of a time-delay neural network to promoter annotation in the *Drosophila melanogaster* genome’, *Comput. Chem.*, 2001, **26**, (1), pp. 51–56
23 Mahadevan, I., and Ghosh, I.: ‘Analysis of *E. coli* promoter structures using neural networks’, *Nucleic Acids Res.*, 1994, **22**, (11), pp. 2158–2165
24 Qian, N., and Sejnowski, T.J.: ‘Predicting the secondary structure of globular proteins using neural network models’, *J. Mol. Biol.*, 1988, **202**, (4), pp. 865–884
25 Rost, B., and Sander, C.: ‘Improved prediction of protein secondary structure by use of sequence profiles and neural networks’, *Proc. Nat. Acad. Sci.*, 1993, **90**, (16), pp. 7558–7562
26 Rost, B., and Sander, C.: ‘Prediction of protein secondary structure at better than 70% accuracy’, *J. Mol. Biol.*, 1993, **232**, pp. 584–599
27 Riis, S.K., and Krogh, A.: ‘Improving prediction of protein secondary structure using structured neural networks and multiple sequence alignments’, *J. Comput. Biol.*, 1996, **3**, pp. 163–183
28 Kaur, H., Raghava, G.P.: ‘A neural network method for prediction of beta-turn types in proteins using evolutionary information’, *Bioinformatics*, 2004, **20**, (16), pp. 2751–2758
29 McGuffin, L.J., Bryson, K., and Jones, D.T.: ‘The PSIPRED protein structure prediction server’, *Bioinformatics*, 2000, **16**, (4), pp. 404–405
30 Wood, M.J., and Hirst, J.D.: ‘Predicting protein secondary structure by cascade-correlation neural networks’, *Bioinformatics*, 2004, **20**, (3), pp. 419–420
31 Pasquier, C., Promponas, V.J., and Hamodrakas, S.J.: ‘PRED-CLASS: cascading neural networks for generalized protein classification and genome-wide applications’, *Proteins*, 2001, **44**, (3), pp. 361–369
32 Ding, C.H., and Dubchak, I.: ‘Multi-class protein fold recognition using support vector machines and neural networks’, *Bioinformatics*, 2001, **17**, (4), pp. 349–358
33 Berry, E.A., Dalby, A.R., and Yang, Z.R.: ‘Reduced bio basis function neural network for identification of protein phosphorylation sites: comparison with pattern recognition algorithms’, *Comput. Biol. Chem.*, 2004, **28**, (1), pp. 75–85
34 Shepherd, A.J., Gorse, D., and Thornton, J.M.: ‘A novel approach to the recognition of protein architecture from sequence using Fourier analysis and neural networks’, *Proteins*, 2003, **50**, (2), pp. 290–302
35 Pollastri, G., Baldi, P., Fariselli, P., and Casadio, R.: ‘Improved prediction of the number of residue contacts in proteins by recurrent neural networks’, *Bioinformatics*, 2001, **17**, (1), pp. 234–242
36 Lin, K., May, A.C., and Taylor, W.R.: ‘Threading using neural nEtwork (TUNE): the measure of protein sequence-structure compatibility’, *Bioinformatics*, 2002, **18**, (10), pp. 1350–1357
37 Cai, Y.D., Liu, X.J., and Chou, K.C.: ‘Prediction of protein secondary structure content by artificial neural network’, *J. Comput. Chem.*, 2003, **24**, (6), pp. 727–731
38 Dietmann, S., and Frommel, C.: ‘Prediction of 3D neighbours of molecular surface patches in proteins by artificial neural networks’, *Bioinformatics*, 2002, **18**, (1), pp. 167–174
39 O’Neill, M.C., and Song, L.: ‘Neural network analysis of lymphoma microarray data: prognosis and diagnosis near-perfect’, *BMC Bioinformatics*, 2003, **4**, (1), pp. 13–20
40 Sawa, T., and Ohno-Machado, L.: ‘A neural network-based similarity index for clustering DNA microarray data’, *Comput. Biol. Med.*, 2003, **33**, (1), pp. 1–15
41 Ando, T., Suguro, M., Hanai, T., Kobayashi, T., Honda, H., and Seto, M.: ‘Fuzzy neural network applied to gene expression profiling for predicting the prognosis of diffuse large B-cell lymphoma’, *Jpn. J. Cancer. Res.*, 2002, **93**, (11), pp. 1207–1212
42 Bicciato, S., Pandin, M., Didone, G., and Di Bello, C.: ‘Pattern identification and classification in gene expression data using an autoassociative neural network model’, *Biotechnol. Bioeng.*, 2003, **81**, (5), pp. 594–606
43 Liang, Y., Georgre, E.O., and Kelemen, A.: ‘Bayesian neural network for microarray data’. Technical Report, Department of Mathematical Sciences, University of Memphis, Memphis, TN 38152, USA
44 Vohradsky, J.: ‘Neural network model of gene expression’, *FASEB J.*, 2001, **15**, (3), pp. 846–854
45 PLANN Software, ‘PNN Technologies’, *Pasadena*, CA, USA
46 Herrero, J., Valencia, A., and Dopazo, J.: ‘A hierarchical unsupervised growing neural network for clustering gene expression patterns’, *Bioinformatics*, 2001, **17**, (2), pp. 126–136
47 Ressom, H., Wang, D., and Natarajan, P.: ‘Clustering gene expression data using adaptive double self-organizing map’, *Physiol. Genomics*, 2003, **14**, pp. 35–46
48 Ritchie, M.D., White, B.C., Parker, J.S., Hahn, L.W., and Moore, J.H.: ‘Optimization of neural network architecture using genetic programming improves detection and modeling of gene-gene interactions in studies of human diseases’, *BMC Bioinformatics*, 2003, **4**, (1), pp. 28–36

49 Dopazo, J., and Carazo, J.M.: ‘Phylogenetic reconstruction using an unsupervised growing neural network that adopts the topology of a phylogenetic tree’, *J. Mol. Evol.*, 1997, **44**, pp. 226–233
50 Parbhane, R.V., Tambe, S.S., and Kulkarni, B.D.: ‘Analysis of DNA curvature using artificial neural networks’, *Bioinformatics*, 1998, **14**, (2), pp. 131–138
51 Draghici, S., and Potter, R.B.: ‘Predicting HIV drug resistance with neural networks’, *Bioinformatics*, 2003, **19**, (1), pp. 98–107
52 Spicker, J.S., Wikman, F., Lu, M.L., Cordon-Cardo, C., Workman, C., ORntoft, T.F., Brunak, S., and Knudsen, S.: ‘Neural network predicts sequence of TP53 gene based on DNA chip’, *Bioinformatics*, 2002, **18**, (8), pp. 1133–1134
53 Alvager, T., Graham, G., Hutchison, D., and Westgard, J.: ‘Neural network method to analyse data compression in DNA and RNA sequences’, *J. Chem. Inf. Comput. Sci.*, 1997, **37**, (2), pp. 335–337
54 Sherriff, A., and Ott, J.: ‘Applications of neural networks for gene finding’, *Adv. Genet.*, 2001, **42**, pp. 287–297
55 Pal, S.K., Polkowski, L., and Skowron, A.: ‘Rough-neuro computing: A way of computing with words’ (Springer, Berlin, 2003)
56 Pal, S.K., and Mitra, S.: ‘Neuro-fuzzy pattern recognition: methods in soft computing paradigm’ (John Wiley, NY, 1999)
57 Pal, S.K., and Shiu, S.C.K.: ‘Foundations of soft case based reasoning’ (John Wiley, NY, 2004)

Appl Intell (2007) 26:183–195
DOI 10.1007/s10489-006-0018-y

Genetic operators for combinatorial optimization in TSP and microarray gene ordering

Shubhra Sankar Ray · Sanghamitra Bandyopadhyay · Sankar K. Pal

Published online: 9 November 2006

Abstract This paper deals with some new operators of genetic algorithms and[-27pc] demonstrates their effectiveness to the traveling salesman problem (TSP) and microarray gene ordering. The new operators developed are nearest fragment operator based on the concept of nearest neighbor heuristic, and a modified version of order crossover operator. While these result in faster convergence of Genetic Algorithm (GAs) in finding the optimal order of genes in microarray and cities in TSP, the nearest fragment operator can augment the search space quickly and thus obtain much better results compared to other heuristics. Appropriate number of fragments for the nearest fragment operator and appropriate substring length in terms of the number of cities/genes for the modified order crossover operator are determined systematically. Gene order provided by the proposed method is seen to be superior to other related methods based on GAs, neural networks and clustering in terms of biological scores computed using categorization of the genes.

Keywords Microarray · Gene analysis · Data mining · Biocomputing · Evolutionary algorithm · Soft computing

S. S. Ray (✉) · S. K. Pal
Center for Soft Computing Research: A National Facility, Indian Statistical Institute, Kolkata 700108, India
e-mail: shubhra_r@isical.ac.in

S. K. Pal
e-mail: sankar@isical.ac.in

S. Bandyopadhyay
Machine Intelligence Unit, Indian Statistical Institute, Kolkata 700108, India
e-mail: sanghami@isical.ac.in

1 Introduction

The Traveling Salesman Problem (TSP) is one of the top ten problems, which has been addressed extensively by mathematicians and computer scientists. It has been used as one of the most important test-beds for new combinatorial optimization methods [1]. Its importance stems from the fact there is a plethora of fields in which it finds applications e.g., shop floor control (scheduling), distribution of goods and services (vehicle routing), product design (VLSI layout), microarray gene ordering and DNA fragment assembly. Since the TSP has proved to belong to the class of NP-hard problems [2], heuristics and metaheuristics occupy an important place in the methods so far developed to provide practical solutions for large instances and any problem belonging to the NP-class can be formulated with TSP. The classical formulation is stated as: Given a finite set of cities and the cost of traveling from city I to city j, if a traveling salesman were to visit each city exactly once and then return to the home city, which tour would incur the minimum cost?

Over decades, researchers have suggested a multitude of heuristic algorithms, such as genetic algorithms (GAs) [3–6], tabu search [7, 8], neural networks [9, 10], and ant colonies [11] for solving TSP. Of particular interest are the GAs, due to the effectiveness achieved by this class of techniques in finding near optimal solutions in short computational time for large combinatorial optimization problems. The state-of-the-art techniques for solving TSP with GA incorporates various local search heuristics including modified versions of Lin-Kernighan (LK) heuristic [12–15]. It has been found that, hybridization of local search heuristics with GA for solving TSP leads to better performance, in general. Some important considerations in integrating GAs and Lin-Kernighan heuristic, selection of a proper representation strategy, creation of the initial population and designing of various genetic operators are discussed in [16]. A comprehensive discussion regarding different representation strategies for TSP is provided in [1].

For creating the initial population, random population based approach and nearest neighbor tour construction heuristic (NN) approach are commonly used. Regarding the random population based approach, consider the investigations in [5] and [6] as examples. A GA with immunity (IGA) is developed in [5]. It is based on the theory of immunity in biology, which mainly constructs an immune operator accomplished in two steps: (a) a vaccination and (b) an immune selection. Strategies and methods of selecting vaccines and constructing an immune operator are also mentioned in [5]. IGA can improve the searching ability and adaptability of TSP. Two operators of GA, namely, knowledge based multiple inversion (KBMI) and knowledge based neighborhood swapping (KBNS) are reported in [6]. KBMI helps in exploring the search space efficiently and prevents the GA from getting stuck in the local optima, whereas, KBNS, a deterministic operator, helps the stochastic environment of the working of the GA to derive an extra boost in the positive direction. The corresponding GA for solving TSP is referred to as SWAP_GATSP [6].

Nearest neighbor (NN) tour construction heuristic is a common choice to construct the initial population of chromosome for solving TSP with GAs. Investigations in this line include [4, 17–19]. In [4] a modified multiple-searching genetic algorithm (MMGA) is used with two kinds of chromosomes (namely, conservative and explorer). These two chromosomes operate under different crossover and mutation rates for tour improvement and to avoid the possibility of being trapped at local optima in TSP. Since the NN heuristic takes a locally greedy decision at each step, it is found that several cities that are neglected earlier, may need to be inserted at high costs in the end. This leads to severe mistakes in path construction.

Crossover operators of GAs are seen to rectify the mistakes in path construction by NN or any other approach. Various crossover operators such as order crossover [20], cycle crossover [21], partially matched crossover [3], edge-recombination crossover [22, 23], and matrix crossover [24] have been suggested for the TSP. Order crossover has been observed to be one of the best in terms of quality and speed, and yet is simple to implement for solving TSP using GA [1, 3, 6]. However, the random length of substring, chosen from the parent string for performing crossover may increase the computational time to some extent.

The TSP, with some minor modifications, can be used to model the microarray gene ordering (MGO) problem. In order to determine functional relationships between groups of genes that are often co-regulated and involved in the same cellular process, gene ordering is necessary. Gene ordering provides a sequence of genes such that those that are functionally related are closer to each other in the ordering [25]. This functional relationship among genes is determined by microarray gene expression levels. A microarray is typically a glass slide, onto which thousands of genes are attached at fixed locations (spots). By performing biological experiments gene expression levels are obtained from microarray [26]. A good solution of the gene ordering problem (i.e., finding optimal order of large DNA microarray gene expression data) has similar genes grouped together in clusters. Similarity between genes can be measured in terms of Euclidean distance, Pearson correlation, absolute correlation, Spearman rank correlation, etc. Investigations for clustering gene expression profiles include complete and average linkage (different versions of hierarchical clustering) [25, 27], self-organizing maps (SOM) [28] and Genetic Algorithms [29, 30].

Tsai et al. [29] formulated the MGO problem as TSP and applied family competition GA (FCGA) for solving it. They associated one imaginary city to each gene, and obtain the distance between any two cities (genes) from the matrix of inter gene distances. For microarray gene ordering it is necessary to minimize the distance between the genes that are in the neighborhood of each other, not the distant genes. However, Tsai et al. tried to minimize the distance between distant genes as well [29, 30]. This problem for TSP formulation in microarray gene ordering using GA is minimized in NNGA [30], where relatively long distances between genes are ignored for fitness evaluation. The present investigation has three parts. First, we define a new nearest fragment operator (NF) and a modified version of order crossover operator (viz., modified order crossover, MOC). The NF reduces the limitation of NN heuristic in path construction. This reduction is achieved by determining optimum number of fragments in terms of the number of cities and then greedily reconnecting them. The nearest fragment operator also takes care of the neighbor genes not the distant ones for MGO and provides good results without ignoring any long distances between genes for fitness evaluation. The modified version of order crossover operator (MOC) handles the indefinite computational time due to random length of substring and its random insertion in order crossover. This is done by systematically determining a appropriate substring length from the parent chromosome for performing crossover. While the position of the substring in the parent chromosome is chosen randomly, the length of the substring is predetermined in MOC. In the second part of the investigation, the effectiveness of the new operators for solving TSP is established. Finally, in the third part the microarray gene ordering problem is considered. Comparison of the proposed genetic operators is carried out with other techniques based on GAs, neural networks and clustering in terms of a biological score.

In Section 2 we provide, in brief, a formal definition of TSP and relevance of TSP in microarray gene ordering. The different components of GAs along with their implementation for solving TSP are discussed in Section 3. New operators such as NF and MOC, and the algorithm based on them for TSP and gene ordering are described in Section 4. Then we present in Section 5 the results obtained with our algorithm for different TSP instances. Section 6 concludes the investigation.

2 TSP definition and relevance in microarray gene ordering

Let $\{1, 2, \ldots, n\}$ be the labels of the n cities and $C = [c_{i,j}]$ be an $n \times n$ cost matrix where $c_{i,j}$ denotes the cost of traveling from city i to city j. The Traveling Salesman Problem (TSP) is the problem of finding the shortest closed route among n cities, having as input the complete distance matrix among all cities. The total cost A of a TSP tour is given by

$$A(n) = \sum_{i=1}^{n-1} C_{i,i+1} + C_{n,1} \tag{1}$$

The objective is to find a permutation of the n cities, which has minimum cost.

An optimal gene order, a minimum sum of distances between pairs of adjacent genes in a linear ordering $1, 2, \ldots, n$, can be formulated as [25]

$$F(n) = \sum_{i=1}^{n-1} C_{i,i+1}, \tag{2}$$

where n is the number of genes and $C_{i,i+1}$ is the distance between two genes i and $i+1$. In this study, the Euclidean distance is used to specify the distance $C_{i,i+1}$.

Let $X = x_1, x_2, \ldots, x_k$ and $Y = y_1, y_2, \ldots, y_k$ be the expression levels of the two genes in terms of log-transformed microarray gene expression data obtained over a series of k experiments. The Euclidean distance between X and Y is

$$C_{x,y} = \sqrt{\{x_1 - y_1\}^2 + \{x_2 - y_2\}^2 + \cdots + \{x_k - y_k\}^2}. \tag{3}$$

One can thus construct a matrix of inter-gene distances, which serves as a knowledge-base for mining gene order using GA. Using this matrix one can calculate the total distance between adjacent genes and find that permutation of genes for which the total distance is minimized. This is analogous to the traveling salesman problem. One can simply associate one imaginary city to each gene, and obtain the distance between any two cities (genes) from the matrix of inter gene distances. The formula (Eq. (2)) for optimal gene ordering is the same as used in TSP, except the distance from the last gene to first gene, which is omitted, as the tour is not a circular one.

```
begin GA
    Create initial population
    while generation_count < k do
        /* k = max. number of generations. */
        begin
            Selection and Elitism
            Produce children by crossover from
                        -selected parents
            Mutate the individuals
            Increment generation_count
        end
        Output the best individual found
end GA
```

Fig. 1 The Pseudo-code of Genetic Algorithm (GA)

3 Genetic algorithms for solving TSP and MGO

Genetic algorithms (GAs) [3] are randomized search and optimization techniques guided by the principles of evolution and natural genetics, and have a large amount of implicit parallelism. GAs perform multimodal search in complex landscapes and provide near optimal solutions for objective or fitness function of an optimization problem. In GAs, the parameters of the search space are encoded in the form of strings (chromosomes). A collection of such strings is called a population. Initially a random population is created, which represents different points in the search space. Based on the principle of survival of the fittest, a few among them are selected and each is assigned a number of copies that go into the mating pool. Biologically inspired operators like mating, crossover, and mutation are applied on these strings to yield a new generation of strings. The process of selection, crossover and mutation continues for a fixed number of generations or until a termination condition is satisfied. A general description of Genetic Algorithm is presented in this section for solving TSP using elitist model. Roughly, a genetic algorithm works as follows (see Fig. 1):

3.1 Chromosome representation and nearest-neighbor heuristic

Various representation methods are used to solve the TSP problem using GA. Some of these are binary representation, path representation, matrix representation, adjacency representation, ordinal representation [1]. In order to find the shortest tour for a given set of n cities using GAs, the path representation is more natural for TSP [1]. We have used this representation in our proposed GA. In path representation, the chromosome (or, string) corresponding to a TSP tour is an array of n integers which is a permutation of $(1, 2, \ldots, n)$, where an entry i in position j indicates that city i is visited in the jth time instant. The objective is to find a string with minimum cost.

For solving TSP, the nearest neighbor tour construction heuristic is a common choice to construct the initial population. The salesman starts at some random city and then visits the city nearest to the starting city. From there he visits the nearest city that was not visited so far, until all the cities are visited, and the salesman returns to the starting city. The NN tours have the advantage that they only contain a few severe mistakes, while there are long segments connecting nodes with short edges. Therefore such tours can serve as good starting tours for subsequent refinement using other sophisticated search methods. In NN the main disadvantage is that, several cities are not considered during the course of the algorithm and have to be inserted at high costs in the end. This leads to severe mistakes in path construction. To overcome the disadvantages of the NN heuristics, we propose a new heuristic operator, called the Nearest Fragment (NF) operator (discussed in Section 4). However, unlike NN heuristic that is used only for constructing the initial population, NF is used in every generation (iteration) of GA with a predefined probability for every chromosome in the population as a subsequent tour improvement method.

3.2 Selection and elitism

A number of different selection implementations have been proposed in the literature [3], such as roulette wheel selection, tournament selection, linear normalization selection. Here linear normalization selection, which has a high selection pressure [3], has been implemented. In linear normalization selection, an individual is ranked according to its fitness, and then it is allowed to generate a number of offspring proportional to its rank position. Using the rank position rather than the actual fitness values avoids problems that occur when fitness values are very close to each other (in which case no individual would be favored) or when an extremely fit individual is present in the population (in such a case it would generate most of the offspring in the next generation). This selection technique pushes the population toward the solution in a reasonably fast manner, avoiding the risk of a single individual dominating the population in the space of one or two generations.

A new population is created at each generation (iteration) and after selection procedure, chromosome with highest fitness (least cost) from the previous generation replaces randomly a chromosome from this new generation provided fitness of the fittest chromosome in the previous generation is higher than the best fitness in this current generation in the elitist model.

3.3 Crossover

As the TSP is a permutation problem, it is natural to encode a tour by enumerating the city indices in order. This approach has been dominant in GAs for solving the TSP. In such an encoding, the chromosomal location of a city is not fixed, and only the sequence is meaningful. Some representative crossovers performed on order-based encodings include cycle crossover [21], partially matched crossover [3] and order crossover [3, 20]. Order crossover has been found to be one of the best in terms of quality and speed [1], and yet is simple to implement. Below order crossover is described briefly.

Order crossover (OC). The order crossover operator [3, 20] selects at random a substring in one of the parent tours, and the order of the cities in the selected positions of this parent is imposed on the other parent to produce one child. The other child is generated in an analogous manner for the other parent. As an example consider two parents A and B, and a substring in A of length 3, selected randomly, as shown [3].

$A = 1\ 2\ 3\ |5\ 6\ 7|\ 4\ 8\ 9\ 0$

and

$B = 8\ 7\ 1\ |2\ 3\ 0|\ 9\ 5\ 4\ 6$

The cities in the selected substring in A (here, 5, 6, and 7) are first replaced by * in the receptor B.

$A = 1\ 2\ 3\ |5\ 6\ 7|\ 4\ 8\ 9\ 0$

and

$B = 8\ *\ 1\ |2\ 3\ 0|\ 9 * \ 4 *$

Now to preserve the relative order in the receiver, a sliding motion is made to leave the holes in the matching section marked in the receiver. The convention followed in [3] is to start this sliding motion in the second crossover site, so after the rearrangement we have

$A = 1\ 2\ 3\ |5\ 6\ 7|\ 4\ 8\ 9\ 0$

and

$B = 2\ 3\ 0\ |\ *\ *\ *\ |\ 9\ 4\ 8\ 1$

After that, the stars are replaced with the city names taken from the donor A resulting in the offspring B1

$B1 = 2\ 3\ 0\ |5\ 6\ 7|\ 9\ 4\ 8\ 1$

Similarly the complementary crossover from B to A yields

$A1 = 5\ 6\ 7\ |2\ 3\ 0|\ 4\ 8\ 9\ 1$

In order crossover (OC) the length of the substring for crossover (chosen from the parent string) is random and may often be significantly large; this can have an adverse impact on the computational time. This uncertainity is tackled with a small and predefined length of substring, obtained after extensive empirical studies, for crossover (discussed in Section 3).

3.4 Mutation

For TSP, the simple inversion mutation (SIM) is one of the leading performers [1]. Here simple inversion mutation (SIM) is performed on each string probabilistically as follows: Select randomly two cut points in the string, and reverse the substring between these two cut points. For example consider the tour

(1 2 |3 4 5| 6 7 8)

and suppose that the first cut point is chosen randomly between 2nd city and 3rd city, and the second cut point between the 5th city and the 6th city as shown. Then the resulting string will be

$C =$ (1 2 |5 4 3| 6 7 8)

4 New operators of GA

In this section, some new operators of GAs for solving TSP and microarray gene ordering are described. These are nearest fragment (NF) and modified order crossover (MOC). The genetic algorithm designed using these operators is referred to as FRAG_GA. The structure of the proposed FRAG_GA is provided in Fig. 2.

```
begin FRAG_GA
    Create initial population with Nearest-Neighbor Heuristic
    while generation_count < $k$ do
        /* $k$ = max. number of generations. */
        begin
            Apply NF heuristic or (NF and LK) heuristic
            Elitism
            Linear Normalized Selection
            MOC
            Mutation
            Increment generation_count
        end
        Output the best individual found
end FRAG_GA
```

Fig. 2 The Pseudo-code for FRAG_GA

The basic steps of the FRAG_GA are as follows:

Step 1. Create the string representation (chromosome of GA) for a TSP tour (an array of n integers), which is a permutation of 1, 2, ..., n with Nearest-Neighbor heuristic. Repeat this step to form the population of GA.
Step 2. The NF heuristic is applied on each chromosome probabilistically.
Step 3. Each chromosome is upgraded to local optimal solution using chained LK heuristic probabilistically. (If Step 3. is used in the GA we denote it as FRAG_GALK and otherwise as FRAG_GA.).
Step 4. Fitness of the entire population is evaluated and elitism is used, so that the fittest string among the child population and the parent population is passed into the child population.
Step 5. Using the evaluated fitness of entire population, linear normalized selection procedure is used.
Step 6. Chromosomes are now distributed randomly. Modified Order Crossover operator is applied between two consecutive chromosomes probabilistically.
Step 7. Simple inversion mutation (SIM) is performed on each string probabilistically.
Step 8. Generation count of GA is incremented and if it is less than the maximum number of generations (predefined) then from Step 2 to Step 6 are repeated.

Local search heuristics, such as 2-swap, 2-opt [19], 3-opt [19], and Lin-Kernighan (LK) heuristic [12–14, 16], have been extensively applied in GAs for solving TSPs. These techniques exchange some edges of parents to generate new children. Usually, stronger local search methods correspond to better performing GAs. The mechanisms by which these methods add and preserve edges vary. 2-swap arbitrarily changes two cities at a time, removing four edges at random and adding four edges at random. 2-opt, 3-opt and LK exchange edges if the generated solution is better than the original one. In each iteration, 2-opt and 3-opt exchange two and three edges respectively, while, LK exchanges a variable number of edges. In the present investigation Concorde version of chained-LK [31] is used for fair comparison with [16]. In the following sections, the new operators NF and MOC are described in details.

4.1 Nearest fragment heuristic (NF)

In this process, each string (chromosome in GA) is randomly sliced in *frag* fragments. The value of *frag* is determined by FRAG_GA in terms of the total no. of cities/genes (n) for a particular TSP instance (or microarray data). The systematic process of determining *frag* is described later in this section. As an example, let us consider a string P that is sliced into three random fragments (1–8), (9–14) and (15–20) for a

20-city problem.

$P = 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ |9\ 10\ 11\ 12\ 13\ 14|\ 15\ 16\ 17\ 18\ 19\ 20$

For tour construction the first fragment (9–14) is chosen randomly. From the last city of that fragment (14) the nearest city that is either a start or an end point of a not yet visited tour fragment is determined from the cost matrix. In this example, let the nearest city (among 1, 8, 15 and 20) be 20. The fragment containing the nearest city is connected to the selected fragment, with or without inversion depending on whether the nearest city is the last city of a fragment or not respectively. In this example , the fragment 15–20 is inverted and connected to fragment 9–14, resulting in the following partial tour $P1$.

$P1 = 9\ 10\ 11\ 12\ 13\ 14\ |20\ 19\ 18\ 17\ 16\ 15$

The process is repeated until all fragments have been reconnected. From the last city (15) of $P1$ the nearest city from unvisited fragment (1–8) is say 1. From this result the final string $P2$, shown below, is formed.

$P2 = 9\ 10\ 11\ 12\ 13\ 14\ 20\ 19\ 18\ 17\ 16\ 15\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8$

The basic steps of choosing frag value systematically for a TSP instance with n cities are:

Step (1) Set *frag* value to $frag_{min}$.
Step (2) Run FRAG_GA with the selected *frag* value for x generations and store the number of times the best tour cost is decreased from one generation to the next for that *frag* value. Denote the stored values by $Decr_{cost}$.
Step (3) Increase *frag* by amount $\Delta frag$.
Step (3) Repeat Step 2 to 3 until $frag <= frag_{max}$
Step (4) Find θ consecutive *frag* values for which the summation of corresponding $Decr_{cost}$ values is maximum.
Step (5) The best *frag* value is set to the average of the selected five consecutive *frag* values.
Step (6) Repeat Step 1 to 5 ten times and the average of the best *frag* values is fixed as the final *frag* value for NF for a particular TSP instance.

In this study we have used $frag_{min} = \frac{n}{50}$, $frag_{max} = \frac{n}{2}$, $x = \frac{n}{10}$, $\Delta frag = \frac{n}{50}$, and $\theta = 5$, though experiments were conducted for a few other values as well with similar results. The value of $frag_{max}$ is not set to n as this will lead to NN heuristic. Also, the crossover operator was disabled. The motivation for setting the initial frag value to a low one (and consequently fragment lengths are larger) and then increasing it is that, first exploring the distant neighbors reduces the chances of locking at a local optimal tour for the GA. The probabilistic use of NF also helps to come out from local optimal solution by leaving some chromosomes for mutation and crossover operators to explore.

As an example, consider a 100 city problem with $frag = \frac{n}{16}$, and consequently the fragment length is 16 on an average. As the initial population of the FRAG_GA is formed with NN heuristic there is a likelihood that a city at one end of a fragment is close to the 16th neighbor of the similar end of the next/previous fragment and consequently, they may be connected by the NF heuristic. In the later generations of the GA, using $\frac{n}{16}$ fragments in NN heuristic, explores on an average, from any city to the 16th city in the chromosome rather than the 16th neighbor. Due to random slicing of the chromosome, some fragment lengths will be obviously greater than 16 and some less than 16, and consequently different types of neighbors will be considered. The lowest frag value is set to $\frac{n}{16}$ from the studies in [14], where it is mentioned that good/optimal results are obtained for most of the TSP instances in the TSP library [32] with a search space near about 16 neighbors. The more distant neighbors are mostly explored with mutation and crossover operators.

4.2 Modified order crossover (MOC)

As already mentioned, in order crossover the length of a substring is chosen randomly and can lead to an increase in the computational time, this uncertainty can be minimized if the length of the substring for performing crossover can be fixed to a small value. However, no study has been reported in the literature for determining an appropriate value of the length of a substring for performing order crossover. Such an attempt is made in this article for finding a small substring length for MOC that provides good results for TSP/microarray data with the lowest computational cost.

Unlike order crossover, where the substring length is randomly chosen, in MOC substring length is determined automatically by the FRAG_GA in a similar way of choosing frag value in Section 4.1. In the process of choosing appropriate substring length, NF heuristic is also present in FRAG_GA with its final frag value. As final frag value for NF is determined without any crossover operator, it is preferable to start the process of choosing substring length for MOC initially with a very small value like $\frac{n}{32}$ (very close to no MOC) and then increasing it. For example, for a 10 city problem let the systematically chosen substring length by FRAG_GA is 2. Now for the parents A and B the chromosomes may be as follows

$A = 0\ 9\ 8\ 4\ |5\ 6|\ 7\ 3\ 2\ 1$

and

$B = 9\ 5\ 4\ 1\ |2\ 3|\ 0\ 6\ 8\ 7$

The cities in the selected substring in A (here, 5 and 6) are first replaced by $*$ in the receptor B.

$B = 9 * 4\ 1\ |2\ 3|\ 0 * 8\ 7$

Now to preserve the relative order in the receiver, the convention followed in [3] is to gather the holes in the second crossover site and insert the substring there. But this convention leads to loss of information and increases randomness in the receiver because, after insertion of substring 56 in B neither 5 is nearer to 3, nor 6 is nearer to 0. To reduce this randomness, in MOC the holes are gathered in the position of the last deleted city (here city 6) of the receiver B.

$B = 9\ 4\ 1\ 2\ 3\ 0\ |\ * *|\ 8\ 7$

So after substring insertion, B is as follows:

$B = 9\ 4\ 1\ 2\ 3\ 0\ 5\ 6\ 8\ 7$

Now, at least one edge of the substring is nearer to the next city (city 6 is nearer to 8 according to chromosome B, and this information is preserved). Same convention is followed for inserting substring in chromosome A.

5 Experimental results

FRAG_GA is implemented in C on Pentium-4 (1.2 GHz) and the results are compared with those obtained using SWAP_GATSP [6], MMGA [4], IGA [5], OX_SIM, (standard GA with order crossover and simple inversion mutation) [1] MOC_SIM (Modified order crossover and SIM), and self organizing map (SOM) [10] for solving TSP. For fair comparison with the above mentioned methods Lin-Kernighan (LK) heuristic is not used with FRAG_GA, whereas, for comparison with HeSEA [16] and other LK based methods each chromosome in FRAG_GALK is updated probabilistically with 20 runs of consecutive chained LK and mutation (as recommended in [16]). Several benchmark TSP instances, for which the comparative study with various recently developed pure genetic algorithms (without LK), and SOM are available in the literature, are taken from the TSPLIB [32] without any bias on data sets. These include Grtschels24.tsp, bayg29.tsp, Grtschels48.tsp, eil51.tsp, St70.tsp, eil76.tsp, kroA100.tsp, d198.tsp, ts225.tsp, pcb442.tsp and rat783.tsp. For comparative study between HeSEA, FRAG_GALK, and other LK based methods the available TSP instances are lin318, rat783, pr1002, vm1084, pcb1173, u1432, u2152, pr2392, pcb3038, fnl4461, and usa13509. For biological microarray gene ordering, Cell Cycle cdc15, Cell Cycle and Yeast Complexes datasets are chosen [33]. The three data sets consists of about 782, 803 and 979 genes respectively, which are cell cycle regulated in Saccharomyces cerevisiae, with different number of experiments (24, 59 and 79 respectively) [26]. Each dataset is classified into five groups termed G1, S, S/G2, G2/M, and M/G1 by Spellman et al. [26]. Results are compared with those obtained using GAs [29, 30], different versions of hierarchical clustering [25, 27] and self-organizing map (SOM) [28] for solving microarray gene ordering. Throughout the experiments for FRAG_GA, SWAP_GATSP, OX_SIM, and MOC_SIM the population size is set equal to 100. Crossover probability is fixed at 0.6 and mutation probability is fixed at 0.02 across the generations. For FRAG_GA and FRAG_GALK the probability of applying NF heuristic is fixed at 0.3. Using these parameters FRAG_GA first systematically determines and stores the appropriate *frag* value for NF heuristic and substring length for MOC for each problem instance in a way mentioned in Sections 4.1 and 4.2, and then with these values, tour cost is optimized. Table 1 shows the various parameters of different genetic algorithms used in this current investigation.

Table 1 Different parameters of FRAG_GA, SWAP_GATSP, OX_SIM, and MOC_SIM

Population size	NF Probability for FRAG_GA	Crossover probability	Mutation probability
100	0.3	0.6	0.02

First, we provide results comparing our method (FRAG_GA) with other methods that do not use LK heuristics and then comparisons of results are provided with our method incorporating LK heuristic (FRAG_GALK) with other LK based methods .

5.1 Comparison with other GA approaches for TSP

Table 2 summarizes the results obtained over 30 runs by running the FRAG_GA, SWAP_GATSP [6], OX_SIM and MOC_SIM [1] on the aforesaid eleven different TSP instances. For SWAP_GATSP, OX_SIM and MOC_SIM, the overlapping parameters (Table 1) are taken from FRAG_GA. For each problem the total number of cities and the optimal tour cost are mentioned below the problem name in the first column. The total number of generations in which the best result and the average result are obtained is mentioned in columns 3–6 within parentheses. The error percentages are shown in third row for each problem, where the error percentage is defined as

$$E = \frac{average - optimal}{optimal} \times 100. \quad (4)$$

Experimental results (Tables 2 and 3) using FRAG_GA are found to be superior in terms of quality of solution (best result, average result and error percentage) with less number

Table 2 Comparison of the results over 30 runs obtained using FRAG_GA, SWAP_GATSP, OX_SIM, and MOC_SIM for different TSP instances

Problem		FRAG_GA	SWAP_GATSP	OX_SIM	MOC_SIM
Grtschels24	best	1272 (13)	1272 (50)	1272 (800)	1272 (600)
24	average	1272 (100)	1272 (200)	1322 (1500)	1272 (1500)
1272	error (%)	0.0000	0.0000	3.9308	0.0000
Bayg29	best	1610 (30)	1610 (60)	1620 (1000)	1610 (700)
29	average	1610 (100)	1615 (200)	1690 (1500)	1622 (1500)
1610	error (%)	0.0000	0.3106	4.9689	0.7453
Grtschels48	best	5046 (40)	5046 (200)	5097 (2500)	5057 (1700)
48	average	5054 (150)	5110 (700)	5410 (3000)	5184 (3000)
5046	error (%)	0.1585	1.2683	7.2136	2.7348
eil51	best	426 (45)	439 (220)	493 (2500)	444 (1600)
51	average	432 (150)	442 (700)	540 (3000)	453 (3000)
426	error (%)	1.4085	3.7559	26.7606	6.3380
St70	best	675 (40)	685 (600)	823 (4500)	698 (4500)
70	average	679 (150)	701 (1000)	920 (7500)	748 (7500)
675	error (%)	0.5926	3.8519	36.2963	10.8148
eil76	best	538 (75)	548 (700)	597 (5000)	562 (3800)
76	average	544 (150)	555 (1000)	620 (7500)	580 (7500)
538	error (%)	1.1152	3.1599	15.2416	7.8067
KroA100	best	21282 (80)	21397 (2000)	21746 (10000)	21514 (8200)
100	average	21303 (500)	21740 (3000)	22120 (12000)	21825 (12000)
21282	error (%)	0.0987	2.1521	3.9376	2.5515
d198	best	15780 (850)	15980 (4000)	16542 (10000)	16122 (9000)
198	average	15865 (2000)	16106 (4000)	17987 (16000)	16348 (16000)
15780	error (%)	0.5387	2.0659	13.9861	3.5995
ts225	best	126643 (1000)	127012 (4000)	135265 (10000)	128994 (10000)
225	average	126778 (2000)	128467 (4000)	138192 (16000)	130994 (16000)
126643	error (%)	0.1066	1.4403	9.1193	3.4356
pcb442	best	50778 (1900)	52160 (8000)	53320 (16000)	52852 (13000)
442	average	50950 (4000)	53800 (8000)	56330 (26000)	54173 (26000)
50778	error (%)	0.3387	5.9514	10.9339	6.6860
rat783	best	8850 (7500)	9732 (12000)	10810 (28000)	10155 (20000)
783	average	9030 (16000)	10087 (16000)	11136 (40000)	10528 (40000)
8806	error (%)	2.5437	14.5469	26.4592	19.5548

of generations when compared with those of other existing GAs [1, 4–6]. It is evident from the table that for different TSP instances the error percentages are lowest for FRAG_GA and the error percentages for MOC_SIM is much less than OX_SIM . The average of error percentages over all the TSP instances for MOC_SIM is 5.8425, which is also less than 14.4407 of OX_SIM. The error averages clearly indicates that the modification of order crossover improves its performance significantly over the existing order crossover which uses random substring length and its random insertion. The average of error percentages over all the TSP instances for FRAG_GA and SWAP_GATSP are 0.6274 and 3.5003 respectively.

Figure 3 shows a comparison of FRAG_GA, SWAP_GATSP and OX_SIM when the fitness value of the fittest string is plotted with iteration. The three programs were run for 12000 iterations for kroa100.tsp with population 100. At any iteration, the FRAG_GA has the lowest tour cost. It took 15.36 seconds, 19.94 seconds and 15.14 seconds by FRAG_GA, SWAP_GATSP and OX_SIM respectively for executing 12000 iterations. Moreover, only FRAG_GA is seen to converge at around 250 iterations at the optimal cost value of 21282 km. On the other hand, the cost is 21397 km for SWAP_GATSP after 3100 iterations and 21990 km for OX_SIM even after 12000 iterations.

Table 3 Average results for various GAs

Problem	Optimal	IGA	MMGA	SOM	FRAG_GA
eil51	426	499	446	432	432
st70	675	–	–	683	679
eil76	538	611	568	556	544
Kroa100	21282	24921	22154	–	21303
d198	15780	17925	16360	–	15865
ts225	126643	135467	129453	–	126778
pcb442	50778	59380	55660	55133	50950

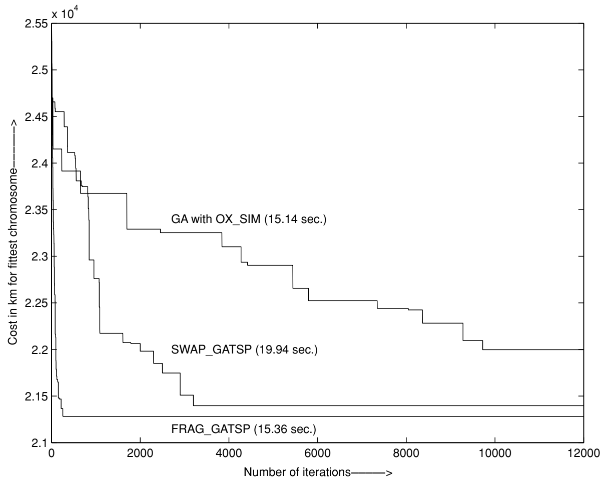


Fig. 3 Cost of fittest string Vs. Iteration for kroa100.tsp

Note that FRAG_GA takes almost the same time as OX_SIM using one more operator (NF), but provides better result in less number of paths. It is further to be pointed out that the NF operator creates an overhead, leading to an increase in the computation time for FRAG_GA, as compared to OX_SIM. However, this is compensated by the gain obtained in using the proposed MOC operator. As a consequence, the time required to execute one iteration, on an average, becomes almost equal for both FRAG_GA and OX_SIM. Similar observations are also made when the proposed method is compared with other GAs [1, 6] and other methods like Self Organizing Map (SOM) [10].

In Table 3 average results of FRAG_GA are compared to other GA based approaches viz., IGA and MMGA (whose results are taken from [4]) and Self Organizing Map (SOM) [10]. As can be seen from the table, the proposed approach is again found to consistently outperform IGA, MMGA, and SOM.

5.2 Comparison with other LK based approaches for TSP

Table 4 summarizes the results obtained over 20 runs by running the FRAG_GALK on different TSP instances mentioned in first column. 20 runs of LK [31] and mutation are applied on randomly chosen 50 chromosomes (among those who are not operated with NF heusristic) in each generation of FRAG_GALK. For HeSEA (with LK) [16], LKH (Multi-trial LK) [13], iterated LK (ILK) [19], and tabu search with LK [8] the results are taken from [16]. While FRAG_GALK, HeSEA, LKH, and concorde chained LK (concorde) [31] are executed on Pentium-4 (1.2 GHz) personal computer, ILK and tabu search with LK are executed on Silicon Graphics 196 MHz MIPS R1000 and Pentium III 800 MHz respectively in [16]. For fair comparison Concorde chained LK is executed separately for same time as FRAG_GALK, but on average concorde converged to the mentioned solutions (in terms of error) before the allocated time. The total number of cities and the optimal tour cost are mentioned below the problem name in the first column. The error percentages (Equation 4) are shown in first row for each problem. The average number of generations over 20 runs for which the error percentages are obtained is mentioned in second row for each TSP instance. The third row for each TSP instance shows the average time in seconds taken by each method. From the table it is clear that FRAG_GALK produces comparable results with HeSEA with same version of LK in less computational time, whereas the quality of solution of FRAG_GALK is better than other algorithms with comparable computational time. So FRAG_GALK seems to be a better TSP solver among the existing ones. The time gain obtained by FRAG_GALK over HeSEA is due to probabilistic single run of computationally effective NF heuristic and MOC over each chromosome in FRAG_GALK, whereas, HeSEA uses 20 runs of edge-assembly crossover between the selected chromosomes and for all possible combinations of chromosomes with probability 1. Generations of LKH, ILK, and tabu with LK are not available.

5.3 Results for microarray gene ordering

FRAG_GA is applied for ordering the genes based on their expression levels obtained from microarray datasets. Performance of FRAG_GA for gene ordering is compared with other methods based on GAs, clustering and neural networks. GA based investigations include NNGA [30] and FCGA [29] (discussed in Section 1). Clustering methods can be broadly divided into hierarchical and nonhierarchical clustering approaches. Hierarchical clustering approaches [25, 27] group gene expressions into trees of clusters. They start with singleton sets and keep on merging the closest sets until all the genes form a single cluster. Complete-linkage and average-linkage belong to this category of clustering technique, differing only in the way the distance between clusters is defined. Nonhierarchical clustering approaches separate genes into groups according to the degree of similarity (as quantified by Euclidian distances, Pearson correlation) among genes. The relationships among the genes in a particular cluster generated by nonhierarchical clustering methods are lost. Self-organizing map (SOM) [28], a particular class of neural network, performs nonhierarchical clustering.

Table 5 summarizes the results in terms of the sum of gene expression distances (Eq. (2)), by executing the FRAG_GA, complete linkage, average linkage and SOM on the three different microarray datasets (results in terms of sum of gene expression distance and code for NNGA and FCGA are not available in the literature [29, 30]). For FRAG_GA and SOM, best and average results obtained over 30 runs are provided,

Table 4 Average results for various LK based algorithms

Problem		FRAG_GALK	HeSEA+LK	LKH	Concorde	ILK	Tabu+LK
lin318	error (%)	0.0000	0.0000	0.1085	0.0000	–	–
318	generation	2.8	3.2	–	–	–	–
42029	time (sec.)	1.4	2.3	1.4	1.4	–	–
rat783	error (%)	0.0000	0.0000	0.0000	0.1761	–	–
783	generation	8.0	8.4	–	–	–	–
8806	time (sec.)	5.9	39.1	2.2	5.9	–	–
pr1002	error (%)	0.0000	0.0000	0.0000	0.0215	0.1482	0.8794
1002	generation	22.2	12.0	–	–	–	–
259045	time (sec.)	34.6	91.0	7.5	34.6	298.0	1211.4
vm1084	error (%)	0.0000	0.0000	0.0068	0.0172	0.0217	0.3932
1084	generation	23.6	10.2	–	–	–	–
239297	time (sec.)	34.2	80.6	12.6	34.2	377.0	597.0
pcb1173	error (%)	0.0000	0.0000	0.0009	0.0070	0.0088	0.6996
1173	generation	22.5	11.5	–	–	–	–
56892	time (sec.)	38.7	84.5	11.8	39.0	159.0	840.0
u1432	error (%)	0.0000	0.0000	0.0000	0.0153	0.0994	0.4949
1432	generation	22.0	11.0	–	–	–	–
152970	time (sec.)	37.7	107.0	6.9	38.0	224.0	775.0
u2152	error (%)	0.0000	0.0000	0.0495	0.0242	0.1743	0.7517
2152	generation	31.5	17.5	–	–	–	–
64253	time (sec.)	48.3	211.0	135.0	49.0	563.0	1624.0
pr2392	error (%)	0.0000	0.0000	0.0000	0.0294	0.1495	0.6492
2392	generation	25.0	14.5	–	–	–	–
378032	time (sec.)	46.6	208.0	26.2	47.0	452.0	1373.0
pcb3038	error (%)	0.0000	0.0000	0.0068	0.1123	0.1213	0.8708
3038	generation	120.6	29.7	–	–	–	–
137694	time (sec.)	245.0	612.0	226.0	219.0	572.0	1149.0
fnl4461	error (%)	0.0014	0.0005	0.0027	0.0734	0.1358	0.9898
4461	generation	265.0	67.8	–	–	–	–
182566	time (sec.)	519.0	2349.0	528.0	519.0	889.0	1018.0
usa13509	error (%)	0.0061	0.0074	0.0065	0.1201	0.1638	0.8897
13509	generation	1102.5	223.0	–	–	–	–
19982859	time (sec.)	19203.0	34984.0	19573.0	19203.0	10694.0	5852.0

Table 5 Comparison of the results over 30 runs in terms of sum of gene expression distances for microarray data using various algorithms

	Cell cycle cdc15		Cell cycle		Yeast complexes	
Algorithms	Best	Average	Best	Average	Best	Average
FRAG_GA	1272 (1690)	1278 (4000)	2349 (2320)	2362 (4000)	3382 (3890)	3396 (6000)
Complete-linkage	1419	1419	2534	2534	3634	3634
Average-linkage	1433	1433	2559	2559	3681	3681
SOM	1874 (100000)	1905 (100000)	3018 (100000)	3094 (100000)	4376 (200000)	4449 (200000)

whereas, for complete and average linkage results remain same for all runs. The genetic parameters for FRAG_GA are the same as used before (see Table 1). For FRAG_GA and SOM the total number of generations/iterations, for which the best and average results are obtained are mentioned in columns 2–7 within parentheses. From the table it is clear that FRAG_GA produces superior gene ordering than related methods in terms of sum of the gene expression distances.

A biological score, that is different from the fitness function, is used to evaluate the final gene ordering. The biological

Table 6 Comparison of the best results over 30 runs in terms of $S(N)$ values for microarray data

Algorithms	Cell cycle cdc15	Cell cycle	Yeast complexes
FRAG_GA	540	635	384
NNGA	539	634	384
FCGA	521	627	–
Complete-linkage	498	598	340
Average-linkage	500	581	331
SOM	461	578	306

score is defined as [29]

$$S(n) = \sum_{i=1}^{n-1} s_{i,,i+1}$$

where

$$s_{i,,i+1} = 1, \quad \text{if gene } i \text{ and } i+1 \text{are in the same group}$$
$$= 0, \quad \text{if gene } i \text{ and } i+1 \text{ are not in the same group}$$

Using this, a gene ordering would have a higher score when more genes within the same group are aligned next to each other. So higher values of $S(n)$ indicate better gene ordering. For example consider the genes YML120C, YJR048W, YMR002W and YDR432W belonging to groups G2/M, S/G2, S/G2 and G2/M respectively. In the above-mentioned ordering they will return a biological score of $0 + 1 + 0 = 1$, whereas if they are ordered like YJR048W, YMR002W, YDR432W and YML120C then the score will be $1 + 0 + 1 = 2$. The scoring function is therefore seen to reflect well the order of genes in biological sense. Note that, although $S(n)$ provides a good quantitative index for gene ordering, using it as the fitness function in GA based ordering is not practical, since the information about gene categories is unknown for most of the genes in the real world .

Table 6 shows the best results over 30 runs of the above methods in terms of $S(n)$ value, where larger values are better ($S(n)$ values for NNGA are FCGA are taken from [30]). It is clear that FRAG_GA and NNGA [30] are comparable and they both dominate others. Note that FRAG_GA is a conventional GA, while NNGA (hybrid GA) is a one using LK heuristic [12]. The main reason for the good results obtained by FRAG_GA is that, biological solutions of microarray gene ordering lie in more than one sub optimal point (in terms of gene expression distance) rather than one optimal point and there exists different gene orders with same biological score.

6 Discussion and conclusions

A new "nearest fragment operator" (NF) and a modified version of order crossover operators (MOC) of GAs are described along with demonstrating their suitability for solving both TSP and microarray gene ordering (MGO) problem. A systematic method for determining appropriate number of fragments in NF and appropriate substring length in terms of the number of cities/genes in MOC are also provided. These newly designed genetic operators showed superior performance on both TSP and gene ordering problem. The said operators are capable of aligning more genes with the same group next to each other compared to other algorithms, thereby producing better gene ordering. Infact, FRAG_GA produces comparable and sometimes even superior results than NNGA, a GA which implements Lin-Kernighan local search, for solving MGO problem in terms of biological score.

The representation used in the present investigation is a direct one (integer i = city/gene i) and also used in all other state-of-the-art TSP solvers using genetic algorithm and LK heuristic based approaches. An indirect representation, like offset-based representation, in general takes more computational time in representation, whereas, there is no chance for improving the solution quality over optimal results for most of the TSP instances.

An advantage of FRAG_GALK is that the quality of the solution seems to be more stable than that obtained by LKH and concorde chained LK, when used to solve the benchmark TSP problems. An evolutionary algorithm for solving combinatorial optimization problems should comprise mechanisms for preserving good edges and inserting new edges into offspring, as well as mechanisms for maintaining the population diversity. In the proposed approach, nearest fragment heuristic, modified order crossover, and LinKernighan local search preserve good edges and add new edges. The proposed method can seamlessly integrate NF, MOC, and LK to improve the overall search.

The present investigation indicates that incorporation of the new operators in FRAG_GA and LK in FRAG_GALK yield better results as compared to other pure GAs, Self Organizing Map, and related LK based TSP solvers. With its superior results in reasonable computation time FRAG_GALK can be considered as one of the state-of-the-art TSP solver.

Acknowledgment This work is partially supported by the grant no. 22(0346)/02/EMR-II of the Council of Scientific and Industrial Research (CSIR), New Delhi.

References

1. Larranaga P, Kuijpers C, Murga R, Inza I, Dizdarevic S (1999) Genetic algorithms for the traveling salesman problem: a review of representations and operators. Artificial Intell Rev 13:129–170
2. Garey MR, Johnson, DS (1979) Computers and intractability: a guide to the theory of NP-completeness. W. H. Freeman and Co., San Francisco

3. Goldberg DE (1989) Genetic algorithm in search, optimization and machine learning, Machine Learning, Addison-Wesley, New York
4. Tsai CF, Tsai CW, Yang T (2002) A modified multiple-searching method to genetic algorithms for solving traveling salesman problem. In: IEEE int conf systems, Man and cybernetics, vol. 3, pp 6–9
5. Jiao L, Wang L (2000) A novel genetic algorithm based on immunity. IEEE Transactions on Systems, Man and Cybernetics, Part A 30(5):552–561
6. Ray SS, Bandyopadhyay S, Pal SK (2004) New operators of genetic algorithms for traveling salesman problem. Cambridge, UK, ICPR-04 2:497–500
7. Fiechter CN (1994) A parallel tabu search algorithm for large traveling salesman problems. Discrete Appl Math Combin Oper Res Comput Sci 51:243–267
8. Zachariasen M, Dam M (1995) Tabu search on the geometric traveling salesman problem. In: Proc. of int conf on metaheuristics, pp 571–587
9. Potvin JY (1993) The traveling salesman problem: a neural network perspective. ORSA J Comput 5:328–348
10. Bai Y, Zhang W, Jin Z (2006) An new self-organizing maps strategy for solving the traveling salesman problem. Chaos, Solitons & Fractals 28(4):1082–1089
11. Stutzle T, Dorigo M (1999) ACO algorithms for the traveling salesman problem, evolutionary algorithms in engineering and computer science. John Wiley and Sons
12. Lin S, Kernighan BW (1973) An effective heuristic for the traveling salesman problem. Oper Res 21(2):498–516
13. Helsgaun K (2000) An effective implementation of the Lin-Kernighan traveling salesman heuristic. Eur J Oper Res 1:106–130
14. Applegate D, Cook W, Rohe A (2000) Chained Lin-Kernighan for large traveling salesman problems. Tech Rep Dept Comput Appl Math Rice Univ
15. Gamboa D, Rego C, Glover F (2006) Implementation analysis of efficient heuristic algorithms for the traveling salesman problem. Computers & Operations Res 33(4):1154–1172
16. Tsai HK, Yang JM, Tsai YF, Kao CY (2004) An evolutionary algorithm for large traveling salesman problems. IEEE Transactions on Systems, Man and Cybernetics, Part B: Cyebernetics 34(4):1718–1729
17. Reinelt G (1994) The traveling salesman: computational solutions for TSP applications. Lecture notes in computer science, Springer-Verlag 840
18. Bentley JL (1992) Fast algorithms for geometric traveling salesman problems. ORSA J Computing 4(4):387–411
19. Johnson DS, McGeoch LA (1996) The traveling salesman problem: a case study in local optimization. Local search in combinatorial optimization. Wiley and Sons, New York
20. Davis L (1985) Applying adapting algorithms to epistatic domains. In: Proc. int. joint conf. artificial intelligence, Quebec, canada
21. Oliver I, Smith D, Holland J (1987) A study of permutation crossover operators on the traveling salesman problem. Second int. conf. genetic algorithms, pp 224–230
22. Starkweather T, McDaniel S, Mathias K, Whitley D, Whitley C (1991) A comparison of genetic sequencing operators. 4th Int. conf. genetic algorithms, pp 69–76
23. Whitley D, Starkweather T, Fuquay D (1989) Scheduling problems and traveling salesman: the genetic edge recombination operator. 3rd Int. conf. genetic algorithms, pp. 133–140
24. Homaifar A, Guan S, Liepins G (1993) A new approach on the traveling salesman problem by genetic algorithms. 5th Int conf genetic algorithms, pp 460–466
25. Biedl T, Brejov B, Demaine ED, Hamel AM, Vinar T (2001) Optimal arrangement of leaves in the tree representing hierarchical clustering of gene expression data. Tech Rep 2001–14, Dept Computer Sci., Univ. Waterloo
26. Spellman PT, Sherlock G, Zhang MQ, Iyer VR, Anders K, Eisen MB, Brown PO, Botstein D, Futcher B (1998) Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisia by microarray hybridization. Molecular Biology Cell 9:3273–3297
27. Eisen MB, Spellman PT, Brown PO, Botstein D (1998) Cluster analysis and display of genome-wide expression patterns. In: Proc. national academy of sciences, vol. 95, pp 14863–14867
28. Tamayo P, Slonim D, Mesirov J, Zhu Q, Kitareewan S, Dmitrovsky E, Lander ES, Golub TR (1999) Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation. In: Proc. national academy of sciences, pp 2907–2912
29. Tsai HK, Yang JM, Kao CY (2002) Applying genetic algorithms to finding the optimal gene order in displaying the microarray data. GECCO, pp. 610–617
30. Lee SK, Kim YH, Moon BR, (2003) Finding the Optimal Gene Order in Displaying Microarray Data. GECCO, pp. 2215–2226
31. Applegate D, Bixby R, Chvtal V, Cook W (2003) Concorde package. [online]. www.tsp.gatech.edu/concorde/downloads/codes/src/co031219.tgz.
32. TSPLIB, http://www.iwr.uniheidelberg.de/groups/comopt/software/TSPLIB95/.
33. Website, http://www.psrg.lcs.mit.edu/clustering/ismb01/optimal.html.

Shubhra Sankar Ray is a Visiting Research Fellow at the Center for Soft Computing Research: A National Facility, Indian Statistical Institute, Kolkata, India. He received the M.Sc. in Electronic Science and M.Tech in Radiophysics & Electronics from University of Calcutta, Kolkata, India, in 2000 and 2002, respectively. Till March 2006, he had been a Senior Research Fellow of the Council of Scientific and Industrial Research (CSIR), New Delhi, India, working at Machine Intelligence Unit, Indian Statistical Institute, India. His research interests include bioinformatics, evolutionary computation, neural networks, and data mining.

Sanghamitra Bandyopadhyay is an Associate Professor at Indian Statistical Institute, Calcutta, India. She did her Bachelors in Physics and Computer Science in 1988 and 1992 respectively. Subsequently, she did her Masters in Computer Science from Indian Institute of Technology (IIT), Kharagpur in 1994 and Ph.D in Computer Science from Indian Statistical Institute, Calcutta in 1998.

She has worked in Los Alamos National Laboratory, Los Alamos, USA, in 1997, as a graduate research assistant, in the University of New South Wales, Sydney, Australia, in 1999, as a post doctoral fellow, in the Department of Computer Science and Engineering, University of Texas at Arlington, USA, in 2001 as a faculty and researcher, and in the Department of Computer Science and Engineering, University of Maryland Baltimore County, USA, in 2004 as a visiting research faculty.

Dr. Bandyopadhyay is the first recipient of Dr. Shanker Dayal Sharma Gold Medal and Institute Silver Medal for being adjudged the best all round post graduate performer in IIT, Kharagpur in 1994. She has received the Indian National Science Academy (INSA) and the Indian Science Congress Association (ISCA) Young Scientist Awards in 2000, as well as the Indian National Academy of Engineering (INAE) Young Engineers' Award in 2002. She has published over ninety articles in international journals, conference and workshop proceedings, edited books and journal special issues and served as the Program Co-Chair of the 1st International Conference on Pattern Recognition and Machine Intelligence, 2005, Kolkata, India, and as the Tutorial Co-Chair, World Congress on Lateral Computing, 2004, Bangalore, India. She is on the editorial board of the International Journal on Computational Intelligence. Her research interests include Evolutionary and Soft Computation, Pattern Recognition, Data Mining, Bioinformatics, Parallel & Distributed Systems and VLSI.

Sankar K. Pal (www.isical.ac.in/∼sankar) is the Director and Distinguished Scientist of the Indian Statistical Institute. He has founded the Machine Intelligence Unit, and the Center for Soft Computing Research: A National Facility in the Institute in Calcutta. He received a Ph.D. in Radio Physics and Electronics from the University of Calcutta in 1979, and another Ph.D. in Electrical Engineering along with DIC from Imperial College, University of London in 1982.

He worked at the University of California, Berkeley and the University of Maryland, College Park in 1986-87; the NASA Johnson Space Center, Houston, Texas in 1990-92 & 1994; and in US Naval Research Laboratory, Washington DC in 2004. Since 1997 he has been serving as a Distinguished Visitor of IEEE Computer Society (USA) for the Asia-Pacific Region, and held several visiting positions in Hong Kong and Australian universities. Prof. Pal is a Fellow of the IEEE, USA, Third World Academy of Sciences, Italy, International Association for Pattern recognition, USA, and all the four National Academies for Science/Engineering in India. He is a co-author of thirteen books and about three hundred research publications in the areas of Pattern Recognition and Machine Learning, Image Processing, Data Mining and Web Intelligence, Soft Computing, Neural Nets, Genetic Algorithms, Fuzzy Sets, Rough Sets, and Bioinformatics.

He has received the 1990 S.S. Bhatnagar Prize (which is the most coveted award for a scientist in India), and many prestigious awards in India and abroad including the 1999 G.D. Birla Award, 1998 Om Bhasin Award, 1993 Jawaharlal Nehru Fellowship, 2000 Khwarizmi International Award from the Islamic Republic of Iran, 2000–2001 FICCI Award, 1993 Vikram Sarabhai Research Award, 1993 NASA Tech Brief Award (USA), 1994 IEEE Trans. Neural Networks Outstanding Paper Award (USA), 1995 NASA Patent Application Award (USA), 1997 IETE-R.L. Wadhwa Gold Medal, the 2001 INSA-S.H. Zaheer Medal, and 2005-06 P.C. Mahalanobis Birth Centenary Award (Gold Medal) for Lifetime Achievement.

Prof. Pal is an Associate Editor of IEEE Trans. Pattern Analysis and Machine Intelligence, IEEE Trans. Neural Networks [1994–98, 2003–06], Pattern Recognition Letters, Neurocomputing (1995–2005), Applied Intelligence, Information Sciences, Fuzzy Sets and Systems, Fundamenta Informaticae, Int. J. Computational Intelligence and Applications, and Proc. INSA-A; a Member, Executive Advisory Editorial Board, IEEE Trans. Fuzzy Systems, Int. Journal on Image and Graphics, and Int. Journal of Approximate Reasoning; and a Guest Editor of IEEE Computer.

Gene ordering in partitive clustering using microarray expressions

Shubhra Sankar Ray[1,]*, Sanghamitra Bandyopadhyay[2] and Sankar K Pal[1]
[1]*Center for Soft Computing Research: A National Facility,* [2]*Machine Intelligence Unit, Indian Statistical Institute, Kolkata 700 108, India*

**Corresponding author (Fax, 91-33-2578 8699; Email, shubhra_r@isical.ac.in)*

A central step in the analysis of gene expression data is the identification of groups of genes that exhibit similar expression patterns. Clustering and ordering the genes using gene expression data into homogeneous groups was shown to be useful in functional annotation, tissue classification, regulatory motif identification, and other applications. Although there is a rich literature on gene ordering in hierarchical clustering framework for gene expression analysis, there is no work addressing and evaluating the importance of gene ordering in partitive clustering framework, to the best knowledge of the authors. Outside the framework of hierarchical clustering, different gene ordering algorithms are applied on the whole data set, and the domain of partitive clustering is still unexplored with gene ordering approaches. A new hybrid method is proposed for ordering genes in each of the clusters obtained from partitive clustering solution, using microarray gene expressions. Two existing algorithms for optimally ordering cities in travelling salesman problem (TSP), namely, FRAG_GALK and Concorde, are hybridized individually with self organizing MAP to show the importance of gene ordering in partitive clustering framework. We validated our hybrid approach using yeast and fibroblast data and showed that our approach improves the result quality of partitive clustering solution, by identifying subclusters within big clusters, grouping functionally correlated genes within clusters, minimization of summation of gene expression distances, and the maximization of biological gene ordering using MIPS categorization. Moreover, the new hybrid approach, finds comparable or sometimes superior biological gene order in less computation time than those obtained by optimal leaf ordering in hierarchical clustering solution.

Ray S S, Bandyopadhyay S and Pal S K 2007 Gene ordering in partitive clustering using microarray expressions; *J. Biosci.* **32 1**019–1025]

1. Introduction

The recent advances in DNA array technologies have resulted in a significant increase in the amount of genomic data. The most powerful and commonly used technique is that involving microarray, which has enabled the monitoring of the expression levels of more than thousands of genes simultaneously. A key step in the analysis of gene expression data is the identification of groups/clusters of genes that manifest similar expression patterns. This translates to the algorithmic problem of clustering and ordering of gene expression data.

The present article deals with the tasks of ordering genes within clusters obtained from self-organizing map (SOM) (Tamayo *et al* 1999). Although there is a rich literature on gene ordering in hierarchical clustering framework (Eisen *et al* 1998; Biedl *et al* 2001; Bar-Joseph *et al* 2001), there is no work addressing and evaluating the importance of gene ordering for gene expression analysis in partitive clustering framework, to the best knowledge of the author. Partitive clustering methods determine unique clusters but do not order genes within cluster and the relationships among the genes in a particular cluster are generally lost. To obtain this relationship among genes in clusters, we propose a

Keywords. Computational biology; evolutionary algorithms; genomics; linear programming; proteomics; soft computing

Abbreviations used: GA, genetic algorithm; MIPS, Munich Information for Protein Sequences; NF, nearest-neighbor; SOM, self-organizing map; TSP, traveling salesman problem novel hybrid method where, an existing pure gene ordering algorithm called "FRAG_GALK" (Ray *et al* 2007), is used to order genes in each clustering solution of SOM (Tamayo *et al* 1999). For the purpose of comparison, instead of FRAG_GALK, an existing traveling salesman problem (TSP) solver Concorde (Applegate *et al* 2003) using linear programming, and optimal leaf ordering in hierarchical clustering solution (applied over the whole data set not partitive clustering solution) by Bar-Joseph *et al* (2001), are also used. Utility of the new hybrid algorithm is shown in improving the quality of the clusters provided by any partitive clustering algorithm by,

- identification of subclusters within big clusters,
- grouping functionally correlated genes within clusters,
- the maximization of biological gene ordering using MIPS categorization, and
- using less computation time than those obtained by optimal leaf ordering in hierarchical clustering solution.

2. Existing approaches

2.1 *Distance measure*

The most popular and probably most simple measures for finding global similarity between genes are the Pearson correlation, a statistical measure of linear dependence between random variables.

Let $X = x_1, x_2, \dots, x_k$ and $Y = y_1, y_2, \dots, y_k$ be the expression vectors of the two genes in terms of log-transformed microarray gene expression data obtained over a series of k experiments. Using Pearson correlation the distance between gene X and Y can be formulated as

$$C_{x,y} = 1 - P_{x,y}, \tag{1}$$

where $P_{x,y}$ represents the centered Pearson correlation and is defined as

$$P_{x,y} = \frac{1}{k}\sum_{i=1}^{k}\left(\frac{x_i - \bar{X}}{\sigma_x}\right)\left(\frac{y_i - \bar{Y}}{\sigma_y}\right), \tag{2}$$

where $\bar{X}$ and σ_x are the mean and standard deviation of the gene X, respectively.

2.2 *Gene ordering methods*

Hierarchical clustering does not determine unique clusters. Thus the user has to determine which of the subtrees are clusters and which subtrees are only a part of a bigger cluster. So in the framework of hierarchical clustering a gene ordering algorithm helps the user to identify clusters by means of visual display and interpret the data (Bar-Joseph *et al* 2001), whereas, in partitive clustering clusters are identified by the algorithm automatically and the solutions are robust and not sensible to noise (Tamayo *et al* 1999) like hierarchical clustering. For partitive clustering based approaches as well as for hierarchical clustering approaches microarray gene ordering (MGO) within clusters using gene expression information is necessary for the following reasons:

(i) Gene ordering helps to identify subclusters in big clusters by means of visual inspection of the ordered gene expression data (Bar-Joseph *et al* 2001).
(ii) Genes that are adjacent in a linear ordering are often functionally co-regulated and involved in the same cellular process (Bar-Joseph *et al* 2001). Biological analysis is often done in the context of this linear ordering.
(iii) The relationships among the genes in a particular cluster generated by partitive clustering algorithms are generally lost. This relationship (closer or distant) among genes within clusters can be obtained using gene ordering approaches.
(iv) It provides smooth display of clustered genes, where the functionally related genes are nearer in the ordering.

Ideally, one would like to obtain a linear order of all genes that puts similar genes close to each other; such that for any two consecutive genes the distance between them is small. So, gene ordering problem is similar to TSP (Pal *et al* 2006) where, cities are ordered instead of genes (Biedl *et al* 2001; Ray *et al* 2007; Tsai *et al* 2004). Let $\{1, 2, \dots, n\}$ be the labels of the n cities and $C = [c_{i,j}]$ be an $n \times n$ distance matrix where $c_{i,j}$ denotes the distance of traveling from city i to city j. The TSP is the problem of finding the shortest closed route among n cities, having as input the complete distance matrix among all cities. The total cost A of a TSP tour is given by

$$A(n) = \sum_{i=1}^{n-1} C_{i,i+1} + C_{n,1}. \tag{3}$$

The objective is to find a permutation of the n cities, which has minimum distance. Similarly, an optimal gene order can be obtained by minimizing the summation of gene expression distances (or maximizing summation of gene expression similarities) between pairs of adjacent genes in a linear ordering 1,2,..., n. This can be formulated as (Biedl *et al* 2001)

$$F(n) = \sum_{i=1}^{n-1} C_{i,i+1}, \tag{4}$$

where n is the number of genes and $c_{i,j+1}$ is the distance/similarity between two genes i and $i + 1$ obtained from distance/similarity matrix. The formula (eq. 4) for optimal gene ordering is the same as used in TSP, except the distance from the last gene to first gene, which is omitted, as the tour is not a circular one. In the related investigations, FRAG_GALK (Ray *et al* 2007) and HeSGA (heterogeneous selection genetic algorithm (Tsai *et al* 2004), was applied to order genes of the whole dataset, and consequently clustering information was missing from the ordering solution.

A method of ordering genes for a partitive clustering solution is currently missing. Here, we define the summation of gene expression distances for a partitive clustering solution as

$$F_1(n) = \sum_{j=1}^{k} \sum_{i=1}^{n_j - 1} C^{j}_{i,i+1}, \tag{5}$$

where k is the total number of clusters, n_j is the number of genes in cluster j, and $C^{j}_{i,j+1}$ is the distance/similarity between two genes i and $i + 1$ in cluster j.

In this investigation we have used two different gene ordering algorithms, FRAG_GALK (Ray *et al* 2007) and Concorde's TSP solver (Applegate *et al* 2003), to order genes of individual clusters found by SOM, as they can obtain the optimal order of cities to many of the TSPLIB instances; the largest having 13,509 and 15,112 cities, respectively. While FRAG_GALK is a genetic algorithm (GA) (Pal *et al* 2006) based TSP solver, Concorde is a linear programming based TSP solver and much slower than FRAG_GALK. Here we briefly discuss the various steps used in FRAG_GALK, which are also available in Ray *et al* (2007). The steps are:

Step 1: Create the string representation (chromosome of GA) for a gene order (an array of n integers), which is a permutation of 1, 2, ······ , n with nearest-neighbor (NF) heuristic. Repeat this step to form the initial population of GA.
Step 2: The NF heuristic is applied on each chromosome probabilistically.
Step 3: Each chromosome is upgraded to local optimal solution using chained LK heuristic (Applegate *et al* 2003) probabilistically.
Step 4: Fitness of the entire population is evaluated and elitism is used, so that the fittest string among the child population and the parent population is passed into the child population.
Step 5: Using the evaluated fitness of entire population, linear normalized selection procedure is used.
Step 6: Chromosomes are now distributed randomly and modified order crossover operator is applied between two consecutive chromosomes probabilistically.
Step 7: Simple inversion mutation (SIM) is performed on each string probabilistically.
Step 8: Generation count of GA is incremented and if it is less than the maximum number of generations (predefined) then from step 2 to step 6 are repeated.

3. Materials and methods

3.1 *Description of data sets*

In the present investigation, data sets like cell cycle (Sherlock *et al* 2001), yeast complex (Eisen *et al* 1998; Bar-Joseph *et al* 2001), all yeast (Eisen *et al* 1998; Website: Eisenlab: *http://rana.lbl.gov./EisenData.htm*) and fibroblast (Iyer *et al* 1999) are chosen. Table 1 shows the name of the data sets, number of genes in each dataset, number of biological gene categories, name of experiment types and number of time points under each type, and finally the total number of time points for a particular dataset. The first three data sets of Saccharomyces cerevisiae consist of 652, 979 and 6221 genes, and 184, 79 and 80 time points, respectively. The genes in the three data sets are classified according to the top level classification (hierarchical structure) of Munich Information for Protein Sequences (MIPS) (*http://www.mips.com*) into 16, 16, and 18 categories, respectively. For the cell cycle data, first we have downloaded 652 cell cycle regulated gene names from the MIPS website. These gene names are then uploaded in Stanford Microarray Database (Sherlock *et al* 2001) and corresponding gene expression values are downloaded with default parameters by selecting all the cell cycle, sporulation, heat shock and diauxic shift experiments. The fibroblast dataset consists of 517 genes and 18 time points related to the response of human fibroblasts to serum. According to gene omnibus (GO) annotation, 517 fibroblast genes are distributed in 1347 categories. After downloading, the order of genes (along with their expression vectors) is randomized in each dataset to remove initial gene order bias.

Table 1. Summary for different microarray data sets

Dataset	No. of genes	Category	Experiments performed				Total
Cell cycle	652	MIPS 16	Cell cycle 93	Sporulation 9	Shock 56	Diauxic shift 26	184
Yeast complex	979	MIPS 16	Cell cycle 18+14+15	Sporulation 7+4	Shock 6+4+4	Diauxic shift 7	79
All yeast	6221	MIPS 18	Cell cycle 60	Sporulation 13	Diauxic shift 7		80
Fibroblast	517	GO 1347	Serum response 12	Cycloheximide 6			18

3.2 *New hybrid algorithm for ordering genes in partitive clustering*

It is mentioned in § 2.2 that, FRAG_GALK is applied separately on each of the gene clusters found by SOM to identify subclusters within large clusters, and to group the functionally correlated genes within clusters. The number of nodes/clusters of SOM are chosen according to the number of MIPS categories (top level of hierarchical tree) for yeast data, and available information in Sharan *et al* (2003) for fibroblast data.

4. Biological interpretation

In case of cell cycle, yeast complex, and all yeast data the MIPS functional categorization is available for most of the genes. The categorization is hierarchical in nature and allows a gene to belong to more than one category. A biological score, that is different from the similarity/distance measures, is used to evaluate the final gene ordering. Each gene that has undergone MIPS categorization can belong to one or more category, while there are many unclassified genes also (no category). A vector $V(\mathrm{g}) = (v_1, v_2, \dots, v_j)$ is used to represent the category status of each gene g, where j is the number of categories. The value of v_j is 1 if gene g is in the jth category; otherwise is zero. Based on the information about categorization, the score of a gene order for multiple class genes is defined as (Tsai *et al* 2004)

$$S(n) = \sum_{i=1}^{N-1} G(g_i, g_{i+1}), \tag{6}$$

where N is the number of genes, g_i and g_{i+1} are the adjacent genes and $\mathrm{G}(g_i, g_{i+1})$ is defined as

$$G(g_i, g_{i+1}) = \sum_{k=1}^{j} V(g_i)_k V(g_{i+1})_k, \tag{7}$$

where $V(g_i)_k$ represents the kth entry of vector $V(g_i)$. For example consider the genes $g_1, g_2, \dots, g_5$, which are classified into 15 categories and represented by the following vectors:

$V(g_1) = (1,0,1,1,0,0,0,0,0,0,0,0,0,0,0)$

$V(g_2) = (1,1,1,1,0,0,0,0,0,0,0,0,0,0,0)$

$V(g_3) = (0,0,1,0,0,0,0,0,1,0,0,0,0,0,0)$

$V(g_4) = (0,0,0,1,1,0,0,0,0,1,0,0,0,0,0)$ and

$V(g_5) = (0,0,0,0,1,0,0,0,0,1,0,0,0,0,0)$.

Considering the gene order g_1,g_2,g_3,g_4,g_5,

$G(g,g_2) = 3$, $G(g_2,g_5) = 1$, $G(g_3,g_4) = 0$, $G(g_4,g_5) = 2$, and

$S(n) = G(g_1,g_2) + G(g_2,g_3) + G(g_3,g_4) + G(g_4,g_5)$
$= 3 + 1 + 0 + 2 = 6$

Using scoring function $S(n)$, a gene ordering would have a higher score when more genes within the same group are aligned next to each other. So higher values of $S(n)$ are better and can be used to evaluate the goodness of a particular gene order.

5. Experimental results

Experiments of gene ordering are conducted in Matlab 7 on Sun Fire V 890 (1.2 GHz and 8 GB RAM). The codes for Bar-Joseph et al's (2001) leaf ordering in hierarchical clustering solution are downloaded from (Venet 2003). Performance of the proposed FRAG_GALK for gene ordering is compared mainly with Concorde's linear programming algorithm and Bar-Joseph et al's method. SOM is available in Expander (Sharan *et al* 2003) and used with 16, 16, and 18 clusters for clustering cell cycle, yeast complex, and all yeast data sets, respectively, as genes in these datasets are classified according to MIPS into 16, 16, and 18 functional categories. For fibroblast data SOM is used with 6 clusters as 6 gene clusters are identified in Sharan *et al* (2003). Finally FRAG_ GALK and Concorde are applied separately on the gene clusters obtained by SOM, and Bar-Joseph et al's method is applied on the average linkage based hierarchical clustering solution for each dataset.

5.1 *Relevance of gene ordering in partitive clustering*

To show the utility of the hybrid method in identifying different subclusters within big clusters and grouping the functionally correlated genes within clusters, here for illustration, the visual displays are presented for fibroblast (Figure 1a, b) and yeast complex (Figure 1c, d) data. Using SOM fibroblast genes are first clustered in 6 clusters (stated previously). Visual display of these 6 clusters is shown in figure 1a. Observing this visual pattern no subcluster can be identified in each cluster. After applying FRAG_GALK on each cluster, closely related genes with similar expressions are aligned next to each other as shown in Figure 1b. Gene ordering here suggests that 2 or more subclusters exists at least in clusters 1, 4 and 6, and it will be useful to increase the number of nodes of SOM to at least 9 for fibroblast data. Note that, Iyer *et al* (1999) identified 10 clusters of genes for this data using average linkage clustering.

Yeast Complex data is first clustered in 16 groups using SOM. Visual display of first 6 clusters/groups is shown in figure 1c. When the genes are ordered in each cluster with FRAG_GALK, 4, 4, 5, and 2 distinct subclusters are identified using visual display in clusters 2, 3, 4, and 5 respectively. Genes names along with their functional categories (indexes) for each subcluster within cluster 4 are shown in table 2 for the purpose of illustration. Names of

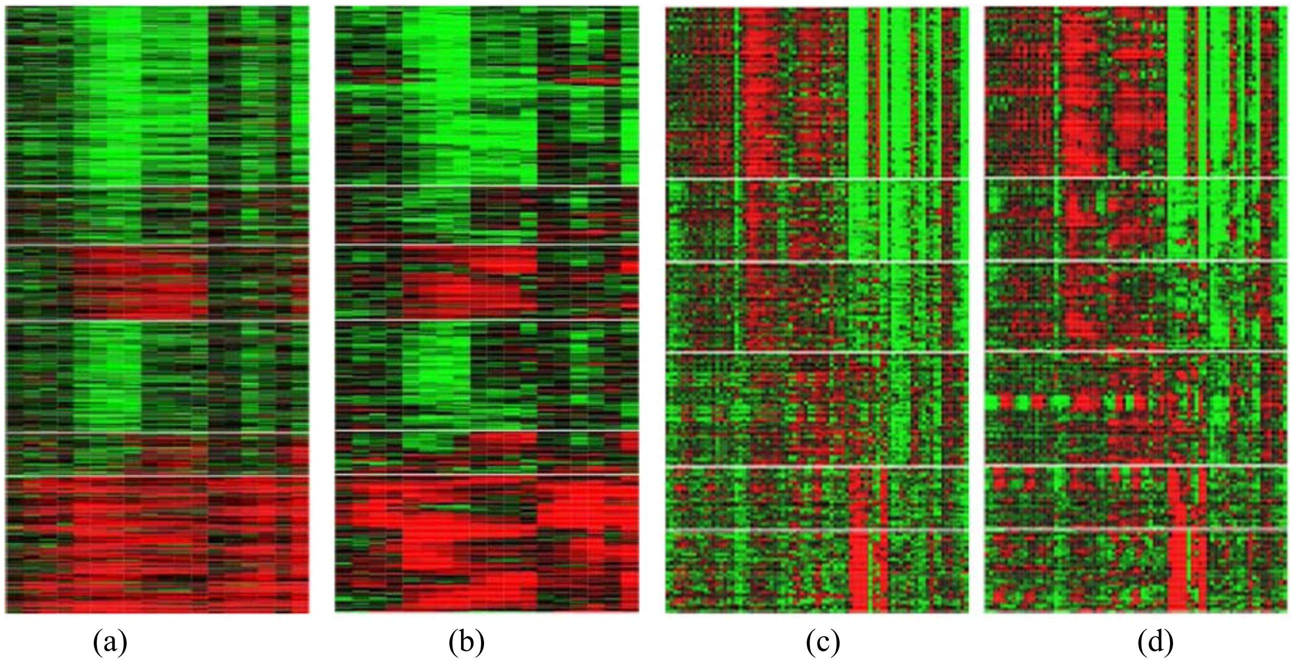


Figure 1. Comparing SOM with 'SOM+FRAG_GALK' for Fibroblast data (**a** and **b** respectively) and Yeast Complex data (**c** and **d** respectively). The expression profiles are represented as lines of coloured boxes using Expander (Sharan *et al* 2003), each of which corresponds to a single experiment.

Table 2. Gene subclusters found by SOM+FRAG_GALK and their functional category indexes in cluster 4 for yeast complex data

Cluster	Subcluster	Genes	Functional index
4	1	YLR093C, YNL121C, YLR170C, YML112W, YBR160W, YBR171W, YLR378C, YML019W, YPL234C, YOR039W	6
	2	YKR068C, YLL050C, YGL200C, YML012W, YPL218W, YKL080W, YDR086C, YNL153C, YKL122C, YLR292C, YGL112C, YLR268W YLR447C	6 and 9
	3	YBR010W, YNL031C, YBL003C, YDR225W, YDR224C, YNL030W, YBR009C, YBL002W, YPL256C	3, 4, and 7
	4	YJL025W, YPR101W, YMR061W, YGR195W, YOR244W, YLR105C, YDL043C, YPR056W, YPR057W	4
	5	YGL100W, YNL261W, YKL144C, YNL151C, YJL008C, YER148W	7

the functional categories corresponding to their indexes are shown in table 3. These subclusters of highly coregulated genes cannot be identified if SOM is used alone. For example, all the 9 genes in the 3rd subcluster of cluster 4 (YBR010W, YNL031C, YBL003C, YDR225W, YDR224C, YNL030W, YBR009C, YBL002W and YPL256C) are involved in cell cycle and DNA processing, transcription, and protein with binding function or cofactor requirement. While using SOM these 9 genes are distributed in the cluster 4, after ordering genes in cluster 4 of SOM with FRAG_GALK, they (the 9 genes) are tightly grouped and identified easily using visual display. With all these ordered and clustered genes one can easily zoom in a useful small subset of genes in a cluster which cannot be done alone with partitive clustering methods. In a similar way, subclusters within big clusters are identified by Concorde for all the data sets.

5.2 *Comparative Performance of Algorithms*

The ultimate goal of an ordering algorithm is to order the genes in a way that is biologically meaningful. In this regard, table 4 compares the performance of our proposed two hybrid approaches using FRAG_GALK and Concorde with Bar-Joseph's (Bar-Joseph *et al* 2001) leaf ordering in hierarchical clustering solution in terms of the F_1 value

Table 3. Indexes and corresponding functional category

Functional index	Corresponding functional category
1	Metabolism
2	Energy
3	Cell cycle and DNA processing
4	Transcription
5	Protein synthesis
6	Protein fate (folding, modification, destination)
7	Protein with binding function or cofactor requirement
8	Protein activity regulation
9	Cellular transport, transport facilitation and transport routes

Table 4. Summation of gene expression distances (F_1), biological score (S), and computation time of ordering in seconds (within parenthesis) for different algorithms

	Data sets		
Algorithm	Cell cycle	Yeast complex	All yeast
SOM	442.94 354	547.16 792	3446.60 1730
SOM +FRAG_ GALK	301.72 386 (0.7)	330.54 1011 (1.13)	1919.15 2356 (125)
SOM +concorde	301.72 386 (3.41)	330.54 1011 (15.26)	1919.15 2356 (2272)
Bar-Joseph	300.51 381 (1.8)	330.17 1024 (3.34)	1920.82 2350 (1989)

(eq. 5), S value (eq. 6), and computation time. The performance of an algorithm is better if F_1 value is smaller and S value is larger. For Fibroblast data, no biological score is provided as genes in the same biological group for this data are rare. From the biological scores (table 4), it is evident that FRAG_GALK provides biologically comparable gene order with respect to Concorde and sometimes superior gene order than 'leaf ordering in hierarchical clustering solution' by Bar-Joseph *et al* (2001), for all datasets in least computational time. For example, FRAG_GALK took 125 seconds to order all yeast data (6221 genes) as compared to Concorde and Bar-Joseph *et al*'s method which took 2272 and 1989 seconds respectively.

6. Conclusion

A hybrid method of gene ordering in partitive clustering and its utility in finding useful subgroups of genes within cluster, grouping functionally correlated genes within clusters, maximization of biological gene ordering using MIPS categorization, and minimization of computation time, are demonstrated. The hybrid approach not only determines unique clusters, but also preserves the biologically meaningful relationships among the genes within clusters. Moreover, the hybrid method using SOM with FRAG_ GALK not only requires less computation time (125 s for 18 clusters of all yeast data) but also less amount of RAM (0.1 GB RAM for clusters with 1000 genes) than original Bar-Joseph's method (1989 s and 2 GB RAM for all yeast data). With the hybrid approaches one can easily zoom in a useful small subset of genes in a cluster, which cannot be done alone with partitive clustering methods.

In FRAG_GALK, parallel searching (with large population in genetic algorithm) for optimal gene order in gene clusters (closely related genes) is performed. While this results in reduced searching time for FRAG_GALK than Concorde and Bar-Joseph's method, in terms of biological score FRAG_GALK is comparable with Concorde and sometimes superior to Bar-Joseph's method. It is evident from the experimental results that, the combination of partitive clustering and FRAG_GALK is a promising tool for microarray gene expression analysis.

References

Applegate D, Bixby R, Chvtal V and Cook W 2003 Concorde Package. [Online], *www.tsp.gatech.edu/concorde/downloads/ codes/src/co031219.tgz*

Bar-Joseph Z, Gifford D K and Jaakkola T S 2001 Fast optimal leaf ordering for hierarchical clustering; *Bioinformatics* **17** 2229

Biedl T, Brejov B, Demaine E D, Hamel A M and Vinar T 2001 *Optimal arrangement of leaves in the tree representing hierarchical clustering of gene expression data* (Technical report, Department of Computer Sciemce, University of Waterloo)

Eisen M B, Spellman P T, Brown P O and Botstein D 1998 Cluster analysis and display of genome-wide expression patterns; *Proc. Natl Acad. Sci., USA* **95** 14863–14868

Iyer V R, Eisen M B, Ross D T, Schuler G, Moore T, Lee J C F, Trent J M, Staudt L M *et al* 1999 The transcriptional program in the response of human fibroblasts to serum; *Science* **283** 83–87

Pal S K, Bandyopadhyay S and Ray S S 2006 Evolutionary computation in bioinformatics: A review; *IEEE Trans. Systems Man Cybernetics Part C* **36** 601–615

Ray S S, Bandyopadhyay S and Pal S K 2007 Genetic operators for combinatorial optimization in TSP and microarray gene ordering; *Appl. Intelligence* **26** 183–195

Sharan R, Maron-Katz A and Shamir R 2003 CLICK and EXPANDER: a system for clustering and visualizing gene expression data; *Bioinformatics* **19** 1787–1799

Sherlock G, Hernandez-Boussard T, Kasarskis A, Binkley G, Matese J C, Dwight S S, Kaloper M, Weng S *et al* 2001 The Stanford microarray database; *Nucleic Acids Res.* **29** 152–155

Tamayo P, Slonim D, Mesirov J, Zhu Q, Kitareewan S, Dmitrovsky E, Lander E S and Golub T R 1999 Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation; *Proc. Natl. Acad. Sci. USA* **96** 2907–2912

Tsai H K, Yang J M, Tsai Y F and Kao C Y 2004 An evolutionary approach for gene expression patterns; *IEEE Trans. Info. Tech. Biomed.* **8** 69–78

Venet D 2003 MatArray: a Matlab toolbox for microarray data; *Bioinformatics* **19** 659–660

*e*Publication: 28 June 2007

Dynamic Range-Based Distance Measure for Microarray Expressions and a Fast Gene-Ordering Algorithm

Shubhra Sankar Ray,
Sanghamitra Bandyopadhyay, *Senior Member, IEEE*, and
Sankar K. Pal, *Fellow, IEEE*

***Abstract*—This investigation deals with a new distance measure for genes using their microarray expressions and a new algorithm for fast gene ordering without clustering. This distance measure is called "*Maxrange* distance," where the distance between two genes corresponding to a particular type of experiment is computed using a normalization factor, which is dependent on the dynamic range of the gene expression values of that experiment. The new gene-ordering method called "Minimal Neighbor" is based on the concept of nearest neighbor heuristic involving $O(n^2)$ time complexity. The superiority of this distance measure and the comparability of the ordering algorithm have been extensively established on widely studied microarray data sets by performing statistical tests. An interesting application of this ordering algorithm is also demonstrated for finding useful groups of genes within clusters obtained from a nonhierarchical clustering method like the self-organizing map.**

***Index Terms*—Bioinformatics, clustering, combinatorial optimization, data mining, dynamic range, evolutionary algorithm, gene expression, ordering, self-organizing map (SOM), soft computing.**

I. Introduction

The recent advances in DNA array technologies have resulted in a significant increase in the amount of genomic data [1], [2]. The most powerful and commonly used technique is that involving microarray, which has enabled the monitoring of the expression levels of more than thousands of genes simultaneously. Due to the large quantity of information available from microarray, it is necessary to find an appropriate distance measure for genes and to employ a process of classification of the data in order to obtain initial conclusions about the genes.

This investigation deals with the tasks of measuring the distance between genes, their unidirectional ordering without clustering, and ordering within clusters. The widely used measures for finding the similarity between genes are the Pearson correlation and the Euclidean distance. In computing the similarity, all the aforementioned measures do not assign appropriate weights to gene expressions obtained from different types of experiments, where the expressions differ by orders of magnitude from one type to another. Consequently, gene expression values in the lower dynamic range do get dominated by those with higher dynamic range. A new similarity measure between genes called "*Maxrange* distance" is defined in this correspondence, where local (for a particular type of experiment) similarities between two genes are first normalized with a factor dependent on the dynamic range of gene expression values of that experiment (type) and then summed to find a global distance.

Gene ordering [3] is primarily necessary for identifying groups of highly coregulated genes (discussed in detail in Section II-B). Existing methods using evolutionary algorithms [4], [5], local search [4], [5], and Concorde's linear programming [6] for finding the optimal gene order spend most of the time in repetitive searching for the lowest value of the sum of global similarities within gene groups of the same biological category and result in the same biological score for all possible permutations of genes within the same group. To avoid this situation, a fast gene-ordering algorithm called "Minimal Neighbor" (MN), using nearest neighbor (NN) tour construction heuristic and involving $O(n^2)$ time complexity, is described.

The superiority of the proposed *Maxrange* distance measure over related measures is established by using them on three different ordering algorithms and one hybrid algorithm. Similarly, the comparability of the MN algorithm as compared to two existing algorithms is demonstrated for three different distance measures. An interesting application of the MN for ordering genes in the clusters found by the self-organizing map (SOM) is also demonstrated.

II. Existing Approaches

A. Gene Clustering Methods

Clustering methods can be broadly divided into hierarchical and nonhierarchical clustering approaches. Hierarchical clustering approaches (single, complete, and average linkage) [1]–[3] group gene expressions into trees of clusters. They start with singleton sets and merge all genes until all nodes belong to only one set. Nonhierarchical clustering approaches, such as k means [7], SOM [8], and CLICK [9], separate genes into groups according to the degree of distance among genes. The relationships among the genes in a particular cluster generated by nonhierarchical clustering methods are lost.

B. Gene-Ordering Methods

Hierarchical clustering does not determine unique clusters. So, in the framework of hierarchical clustering, a gene-ordering algorithm helps the user to identify subtrees that are clusters by means of visual display and interpret the data [3]. For nonhierarchical clustering-based approaches as well as for hierarchical clustering approaches, microarray gene ordering within clusters using gene expression information is necessary for the following reasons:

1) Gene ordering helps to identify subclusters in big clusters by means of visual inspection of the clustered gene expression data [3].
2) Genes that are adjacent in linear ordering are often functionally coregulated and involved in the same cellular process [1], [2]. Biological analysis is often done in the context of this linear ordering [3].
3) It provides smooth display of clustered genes, where the functionally related genes are nearer in the ordering [2].
4) The relationships among the genes in a particular cluster generated by nonhierarchical clustering algorithms are lost. This relationship (closer or distant) among genes within clusters can be obtained using gene-ordering approaches.

An optimal gene order can be obtained by minimizing the summation of gene expression distances (or maximizing summation of gene expression similarities) between pairs of adjacent genes in a linear ordering $1, 2, \ldots, n$. This can be formulated as [2]

$$F(n) = \sum_{i=1}^{n-1} C_{i,i+1} \tag{1}$$

Manuscript received April 12, 2006; revised August 29, 2006. This paper was recommended by Associate Editor W. Pedrycz.

S. S. Ray and S. K. Pal are with the Center for Soft Computing Research, Indian Statistical Institute, Calcutta 700108, India (e-mail: shubhra_r@isical.ac.in; sankar@isical.ac.in).

S. Bandyopadhyay is with the Machine Intelligence Unit, Indian Statistical Institute, Calcutta 700108, India (e-mail: sanghami@isical.ac.in).

Digital Object Identifier 10.1109/TSMCB.2006.889812

TABLE I
SUMMARY FOR DIFFERENT MICROARRAY DATA SETS

Dataset	No. of genes	Category	Experiments performed				Total
Cell Cycle	652	MIPS 16	Cell Cycle (-1.2 to 1.2) 93	sporulation (-3.0 to 3.0) 9	shock (-1.5 to 1.5) 56	diauxic shift (-2.0 to 2.0) 26	184
Yeast Complex	979	MIPS 16	Cell Cycle (-1.2 to 1.2) 18+14+15	sporulation (-3.0 to 3.0) 7+4	shock (-1.5 to 1.5) 6+4+4	diauxic shift (-2.0 to 2.0) 7	79
All Yeast	6221	MIPS 18	Cell Cycle (-1.2 to 1.2) 60	sporulation (-3.0 to 3.0) 13	diauxic shift (-2.0 to 2.0) 7		80
Fibroblast	517	GO 1347	Serum response (-3.0 to 3.0) 12	cycloheximide (-3.0 to 3.0) 6			18
Herpes	106	GeneBank 5	No KSHV (-13.0 to 13.0) 1	-TPA (-13.0 to 13.0) 7	TPA (-13.0 to 13.0) 13		21

where n is the number of genes, and $C_{i,i+1}$ is the distance/similarity between two genes i and $i+1$ obtained from the distance/similarity matrix.

A hybrid method (first clustering then ordering) for ordering genes for a hierarchical clustering solution is proposed in [3]. A method for ordering genes for a nonhierarchical clustering solution is currently missing. Although gene-ordering methods exist (described in the next paragraph), the utility and application of these methods to individual clusters of nonhierarchical solution are not reported. In the current investigation, the summation of gene expression distances for a nonhierarchical solution is defined as

$$F_1(n) = \sum_{j=1}^{k} \sum_{i=1}^{n_j - 1} C_{i,i+1}^{j} \tag{2}$$

where k is the total number of clusters, n_j is the number of genes in cluster j, and $C_{i,i+1}^{j}$ is the distance/similarity between two genes i and $i+1$ in cluster j obtained from the distance/similarity matrix.

Tsai *et al.* [4] formulated the gene-ordering problem as a travelling salesman problem (TSP). Concorde's TSP solver [6] can obtain the optimal solutions to 107 of the 110 TSPLIB [10] instances; the largest having 15 112 cities. Thus, Concorde appears to be the best TSP solver currently available, and in Section V, comparisons of results for gene ordering are shown with Concorde. Related works on gene ordering are also available in [5] and [11].

III. MATERIALS AND METHODS

A. Preliminary Concepts of Microarray Technology

Fluorescence is currently the predominant method for microarray signal detection [12]. A critical component of a fluorescence scanner is the photomultiplier tube (PMT), in which fluorescent photons produce electrons that are amplified by the PMT gain. For many microarray scanners, the calibration curve (i.e., the curve showing the relationship between dye concentration and fluorescence intensity) depends on the PMT gain setting [12]. This PMT gain is also varied for different types of experiments of different biological origin. DNA microarray measurements normally assume a linear relationship between the detected fluorescent signal and the concentration of the fluorescent dye that is incorporated into the clone DNA or RNA molecules synthesized from the test sample. Each PMT has its own linear dynamic range within which signal intensity increases linearly with the increase of fluorescent dye concentration [12]. This linear dynamic range also fixes the dynamic range of the recorded microarray data (log ratio values) [12] within which the data values are most reliable and used as the normalization factor in the proposed distance measure to remove variations of biological origin. For example, in Cell-Cycle-related experiments, for dye Cy5, the PMT gain at 960 V fixes the intensity range from x1 to x2, and for dye Cy3, the PMT gain at 760 V fixes the intensity range from y1 to y2. So the linear dynamic range of PMT fixes the linear dynamic range of the data from $\log_2 x1/y1$ to $\log_2 x2/y2$. Note that this dynamic range is available either from the supplementary information (website) of the article/data (Yeast data) or upon request to the authors (Herpes data) and not from the data sets, and hence is not sensitive to outliers. However, due to the wide concentration range for genes expressed in a biological sample, the detected fluorescence intensity does not necessarily remain in the linear range for all genes tiled on a microarray. The proposed dynamic range-based normalization (described in Section III-C) belongs to the category of between-slide or multiple-slide normalization [13]. The two other normalization factors in this category, which aim to allow experiment-to-experiment comparisons when different types of experiment have substantially different spreads in log ratios, are median absolute deviation (MAD) and variance regularization. The two normalization methods, viz., MAD and variance regularization, are dynamic range estimators (not the real one) and implemented for the purpose of comparison. However, the results obtained were not very encouraging.

B. Description of Data Sets

For gene ordering, data sets like Cell Cycle [14], Yeast Complex [1], [3], All Yeast [1], [15], Fibroblast [16], and Herpes [17] are chosen. Table I shows the name of the data sets, number of genes in each data set, number of gene categories, name of experiment types and number of experiments performed under each type, and total number of experiments performed for a particular data set. The dynamic range of expression values of each experiment type is shown within parentheses. The dynamic range of available data represents log ratios of -1.2 to 1.2 for the cell-cycle experiments, -3.0 to 3.0 for sporulation, -1.5 to 1.5 for the shock experiments, -2.0 to 2.0 for the diauxic shift, -3.0 to 3.0 for Fibroblast data, and -13.0 to 13.0 for Herpes data. Herpes data are generated using radioactive probes instead of fluorescent probes, and hence, a higher linear dynamic range is observed compared to other data sets. The first three data sets of

Saccharomyces cerevisiae are classified into 16, 16, and 18 groups, respectively, according to the Munich Information for Protein Sequences (MIPS) [18] categorization. The genes in Fibroblast data are classified into 1347 categories according to the Gene Omnibus annotation. In Herpes data, the genes are broadly assigned to five functional groups and available in [17]. For the Cell-Cycle data, first, we downloaded 652 Cell-Cycle-regulated gene names from the MIPS website. These gene names were then uploaded in the Stanford Microarray Database [14], and corresponding gene expression values are downloaded with default parameters by selecting all the cell cycle, sporulation, heat shock, and diauxic shift experiments. Microarray experiments often produce multiple missing expression values, normally due to various experimental problems. In this correspondence, all the genes with more than 50% missing gene expression values are first eliminated from the data set. Thereafter, for the remaining genes, missing gene expression values are estimated using LSimpute [19] software, a statistical java-based package to estimate missing values.

C. New Distance Measure

A number of measures of distance in studying the behavior of two genes can be used, such as Manhattan [20], Euclidean [20], and Pearson correlation distance [2]. Pearson correlation is oversensitive to large threefold changes (peaks) in gene expression profiles due to multiplication of expression vectors in dot product style and therefore leads to false interpretation of distance between genes in certain cases. Moreover, it is observed that often microarray data consist of different sets of expression values corresponding to different experiment types. Existing distance measures usually take the same normalization factor (like standard deviation for Pearson correlation) for a gene. This normalization factor is independent of the type of experiment, varies from gene to gene, and performs global normalization to all the expression values for a particular gene, thus loosing useful local information. But a closer look at the gene expression data reveals that the dynamic range of expression values differs with the type of experiment and remains the same for all the genes in the data set. So, using the same normalization factor is undesirable for all types of experiments, where expression values differ by orders of magnitude from one kind of experiment to another. Consequently, it may be appropriate and better if normalization is performed

- separately for the different types of experiment with different normalizing factors; thereby preserving the local information;
- keeping the same set of normalization factors for all the genes in the data set.

Such an attempt is made in this correspondence, where two new distance measures are developed using Manhattan distance and Euclidean distance, respectively (to avoid oversensitivity to threefold changes), in which normalization is dependent on the type of experiment. This, in turn, results in equal weighting of distance values for different experiment types. The normalization factor is chosen as the linear dynamic range of data values obtained from PMT for a particular type of experiment.

Let

$$X = x_1^{e_1}, \ldots, x_{i_1}^{e_1}, x_1^{e_2}, \ldots, x_{i_2}^{e_2}, \ldots, x_1^{e_m}, \ldots, x_{i_m}^{e_m}$$
$$Y = y_1^{e_1}, \ldots, y_{i_1}^{e_1}, y_1^{e_2}, \ldots, y_{i_2}^{e_2}, \ldots, y_1^{e_m}, \ldots, y_{i_m}^{e_m}$$

be the expression vectors (levels) of the two genes in terms of log-transformed microarray gene expression data obtained over a series of m different types of experiment $(e_1, e_2, \ldots, e_m)$ consisting of

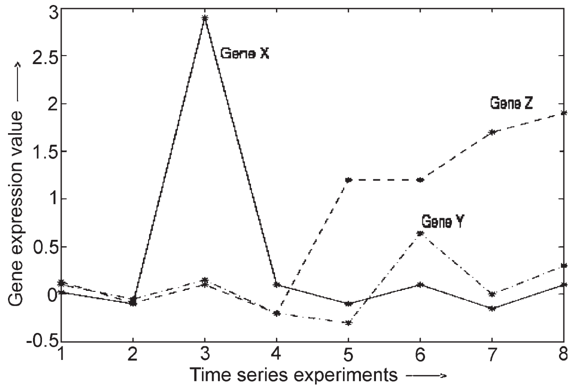


Fig. 1. Expression profile for three genes. According to *Maxrange-M*, the distance between genes X and Y is smaller than Z and Y, which is in opposition with Pearson correlation and Euclidean distance.

$i_1 + i_2 + \cdots + i_m$ experiments in total. Using Manhattan distance, the *Maxrange* distance between X and Y is defined as

$$Maxrange - M_{X,Y} = \frac{1}{m} \sum_{r=1}^{m} \frac{1}{i_r} \times \frac{\sum_{j=1}^{i_r} \left| x_j^{e_r} - y_j^{e_r} \right|}{\text{Max}_{e_r} - \text{Min}_{e_r}} \tag{3}$$

where Max_{e_r} and Min_{e_r} are the maximum and minimum $\log_2(R/G)$ values obtained from the linear dynamic range of the PMT (or radioactive probe) for an experiment of type e_r.

The following can be stated about the measure:

1) $0 \leq Maxrange - M_{X,Y} \leq 1$;
2) $Maxrange - M_{X,Y} = 0$ if and only if $X = Y$;
3) $Maxrange - M_{X,Y} = Maxrange - M_{Y,X}$ (symmetric).

Using the Euclidean distance, the *Maxrange* distance between X and Y is defined as

$$Maxrange - E_{X,Y} = \frac{1}{m} \sum_{r=1}^{m} \frac{1}{i_r} \times \frac{\sqrt{\sum_{j=1}^{i_r} \left(x_j^{e_r} - y_j^{e_r} \right)^2}}{\text{Max}_{e_r} - \text{Min}_{e_r}}. \tag{4}$$

Throughout the literature, we have used *Maxrange-M* and *Maxrange-E* for representing *Maxrange* distance measure using Manhattan and Euclidean distance, respectively.

Let three genes X, Y, and Z with four different types of experiments have the gene expression values $X = 0.02, -0.1, 2.9, 0.1, -0.1, 0.1, -0.15, 0.1$, $Y = 0.1, -0.05, 0.15, -0.2, -0.3, 0.64, 0.0, 0.3$, and $Z = 0.13, -0.09, 0.1, -0.2, 1.2, 1.2, 1.7, 1.9$.

Assume that the first two expression values for all the genes correspond to cell-cycle experiments with dynamic range between 1.2 and -1.2, the third and fourth values correspond to sporulation experiments with dynamic range between 3.0 and -3.0, the fifth and sixth values correspond to shock experiments with dynamic range between 1.5 and -1.5, and the seventh and eighth values correspond to diauxic shift experiments with dynamic range between 2.0 and -2.0. So, the *Maxrange-M* distance and the Pearson correlation distance between genes X and Y are 0.11208 and 0.85202, respectively.

To illustrate the difference between *Maxrange-M* and Pearson correlation, consider Gene X and Gene Y in Fig. 1, which shows two profiles (of length 8), which are highly similar according to the *Maxrange-M* but almost dissimilar (uncorrelated) according to Pearson correlation. This is mainly due to the comparatively large value of the threefold change in Gene X. As opposed to this, in *Maxrange-M*, sensitivity to threefold change is avoided using Manhattan distance, and normalization with a dynamic range of experiments correctly reflects the fact that both profiles have similar expressions for three types of experiments, namely cell cycle, shock, and diauxic shift, and differs in only one expression (among two expressions) for sporulation experiments. *Maxrange-E* distance also shows similar performance as *Maxrange-M*. The Euclidean distance between X and Y is 2.8382, and between Y and Z, it is 2.8317. But X differs with Y in only one expression value of high-range experiment type ($\text{Max}_{e_r} - \text{Min}_{e_r} = 6$), whereas Z differs with Y in three expression values of relatively small-range experiment type. So in the case of Euclidean distances, experiment types with high range dominate experiment types with small-range ones. As opposed to these, the *Maxrange-M* distance between X and Y is 0.11208, which is less than the distance between Y and Z (0.19365). The *Maxrange-E* distance between X and Y is also less than the distance between Y and Z.

D. New Ordering Algorithm

Existing methods, using evolutionary algorithms [4], [5] for finding the optimal gene order, spend most of the time in repetitive searching for the lower value of the sum of gene expression distances in gene groups (genes belonging to same category) and result in the same biological score for all possible permutations of genes within the same group. Under this situation, to avoid repetitive searching, the NN tour construction heuristic can be used to find a near-optimal gene order in terms of gene expression distance. The NN tour has the advantage that it commits only a few severe mistakes in tour construction, while there are long segments connecting nodes with short edges. It has a disadvantage that several genes that are not considered during the course of the algorithm are inserted at high costs in the end. To overcome this to some extent, we propose a new heuristic-based MN algorithm.

Let $1, 2, \ldots, i, \ldots, n$ represent the indices of n genes in the microarray data set, and let the distance between gene i and $i+1$ be denoted as $C_{i,i+1}$. Given this microarray data set of n genes to be ordered and pairwise distance/similarity (of each gene with all other genes) kept in an $n \times n$ matrix (after calculating), the different steps of applying MN are explained below.

Step 1) Find the closest (most similar) pair of genes and merge them into a single array (string) so that there remains $n-2$ genes.

Step 2) Consider only the two end genes of the new array and find the two closest genes for each of them from the remaining genes. Out of these two selected genes, find the one closer to one of the end genes of the array and then place it next to that. The other selected gene is not connected and kept with the remaining genes. The index of this gene is stored for use in the next step. (Note that if both the selected genes are the same in this step, then no gene index can be stored and in the next step we have to compute twice for the selection of two genes, else, only one closest gene is needed to be computed.)

Step 3) Repeat Step 2) until all genes are aligned into a single array of size n.

The computational complexity of Step 1) is $O((n/2)^2)$ as the distance matrix is a symmetric one. This step can also be performed during the calculation of $n \times n$ distance matrix. For Steps 2)–3), the worst case complexity is $O(2*(n-2)*n)$. So the total complexity of the algorithm is $O(n^2)$.

E. New Hybrid Algorithm for Ordering Genes in Nonhierarchical Clustering

It is mentioned in Section II-B that a method for ordering genes for a nonhierarchical clustering solution is currently missing, and that the utility and application of existing gene-ordering methods to individual clusters of nonhierarchical solution are not reported in literature. Here, we propose a simple hybrid algorithm where MN is applied separately on each of the gene clusters found by SOM to identify subclusters within large clusters and to group functionally correlated genes within clusters. This algorithm is referred to as "SOM + MN." The number of nodes/clusters of SOM is chosen according to MIPS categories for Yeast data and available information in relevant literature for Fibroblast and Herpes data. This hybrid method is proposed to show the efficiency of MN in improving the solution quality of a nonhierarchical solution in a computationally effective way.

IV. Biological Interpretation

A biological score, which is different from the similarity/distance measures, is used to evaluate the final gene ordering. Each gene that has undergone MIPS categorization can belong to one or more categories, while there also are many unclassified genes (no category). A vector $V(g) = (v_1, v_2, \ldots, v_j)$ is used to represent the category status of each gene g, where j is the number of categories. The value of v_j is 1 if gene g is in the jth category and 0 otherwise. Based on information about categorization, the score of a gene order for multiple-class genes is defined as [4]

$$S(n) = \sum_{i=1}^{N-1} G(g_i, g_{i+1}) \tag{5}$$

where N is the number of genes, g_i and g_{i+1} are the adjacent genes, and $G(g_i, g_{i+1})$ is defined as

$$G(g_i, g_{i+1}) = \sum_{k=1}^{j} V(g_i)_k V(g_{i+1})_k \tag{6}$$

where $V(g_i)_k$ represents the kth entry of vector $V(g_i)$. Note that $S(n)$ can also be used as the scoring function for single-class genes. Using scoring function $S(n)$, a gene ordering would have a higher score when more genes within the same group are aligned next to each other.

V. Experimental Results

The algorithms of gene ordering and clustering are implemented using mex files in Matlab 7 on Sun Fire V 890 (1.2 GHz and 8 GB RAM). The codes for single, average, and complete linkage and the method of Bar-Joseph *et al.* [3] are downloaded from [21]. The performances of the proposed *Maxrange-M* and *Maxrange-E* distance are compared with Pearson correlation, Euclidean distance, and Manhattan distance, whereas the MN algorithm for gene ordering is compared mainly with Concorde's linear programming [6] algorithm. SOM is used with 16, 16, 18, 6, and 5 nodes (clusters) for clustering Cell Cycle, Yeast Complex, All Yeast, Fibroblast, and Herpes data, respectively. Finally, MN is applied separately on the gene clusters obtained by SOM in the new hybrid algorithm (SOM + MN).

A. Comparative Performance of Algorithms and Distance Measures

Table II shows the summation of gene expression distances in terms of $F(n)$ (computed using (1) for Concorde, MN, and Bar-Joseph)

TABLE II
SUMMATION OF GENE EXPRESSION DISTANCES COMPUTED IN TERMS OF $F(n)$ [(1) FOR CONCORDE, MN, AND BAR-JOSEPH] AND $F_1(n)$ [(2) FOR SOM + MN] VALUE FOR DIFFERENT ORDERING ALGORITHMS (ALGO.) AND DISTANCE MEASURES (DIST.)

		Data Sets				
Dist.	Algo.	Cell cycle	Yeast comp.	All Yeast	Fibro-blast	Herpes
1	Concorde	69.00	80.87	463.85	26.21	2.73
	MN	71.50	84.31	481.69	27.87	2.89
	B-joseph	71.67	84.75	493.51	27.87	2.80
	SOM+MN	73.91	88.27	505.12	29.27	3.12
2	Concorde	286.51	306.14	1773.15	71.82	11.69
	MN	298.25	327.92	1874.23	81.53	12.37
	B-joseph	300.51	330.17	1920.82	81.71	12.12
	SOM+MN	323.67	361.17	1970.16	99.96	14.34
3	Concorde	3913.9	3244.7	20302.2	851.9	419.5
	MN	4039.4	3386.1	21101.7	902.2	431.7
	B-joseph	4051.4	3388.9	21530.4	897.3	431.4
	SOM+MN	4197.9	3518.3	21525.8	943.3	475.2

and $F_1(n)$ value (computed using (2) for SOM + MN) with (1) *Maxrange-M*, (2) Pearson correlation, and (3) Euclidean distance for all the data sets and four ordering algorithms. Hereafter, the serial numbers of these distances are used to denote them in the tables. In this comparative study among ordering algorithms, Concorde provides the lowest sum of gene expression distances in terms of $F(n)$ (1) value for all the distance measures and data sets, although it has the highest computational complexity ($O(2^n)$). MN and Bar-Joseph's algorithm provide comparable results in terms of $F(n)$ value.

The ultimate goal of an ordering algorithm is to order the genes in a way that is biologically meaningful. In this regard, Table III compares the performance of our proposed approach with those of the other ordering methods in terms of the S value (5). Three distance measures are considered, namely: 1) *Maxrange-M*; 2) Pearson; and 3) Euclidean. The biological scores corresponding to Manhattan distance are found to be comparable to those for Pearson correlation distance and hence omitted here. The percentages of improvement over the lowest biological score (in terms of S value) in a particular data set are shown within parentheses and defined as

$$PI_{i,j} = \frac{d_{i,j} - \min_i(d_{i,j})}{\min_i(d_{i,j})} \times 100 \tag{7}$$

where $d_{i,j}$ indicates the biological score (S value) in the ith row and jth column of the result matrix in the concerned tables (Tables III and IV), and $\min_i(d_{i,j})$ indicates the minimum biological score in column j for all i.

Table IV shows the performance of our proposed approach "SOM + MN" with respect to SOM alone for the same set of parameters. These PI values in Tables III and IV are used in the next section for conducting t-tests.

For Fibroblast data, no biological score can be provided as genes in the same biological group for these data are rare. For each of the distance measure and any algorithm, the biological scores (in terms of S value) obtained using MAD (or variance regularization factor) normalization are found to be inferior to the biological scores with *Maxrange* normalization and hence are not provided here. Although in most cases, *Maxrange-E* distance is found to be superior to Euclidean distance and inferior to *Maxrange-M*; for All Yeast data, it performs better ($S(n) = 2431$) than *Maxrange-M* ($S(n) = 2388$) for the MN algorithm. However, the superiority of *Maxrange-M* is evident when

TABLE III
BIOLOGICAL SCORE AND PERCENTAGE OF IMPROVEMENT (PI) VALUE (WITHIN PARENTHESES) FOR DIFFERENT GENE-ORDERING ALGORITHMS (ALGO.) AND DISTANCE (DIST.) MEASURES

		Data Sets			
Algo. & complexity	Dist.	Cell cycle	Yeast comp.	All Yeast	Herpes
Concorde	1	420 (10.24)	1089 (11.69)	2383 (6.05)	44 (29.41)
	2	400 (4.99)	1039 (6.56)	2350 (4.58)	34 (0.00)
$O(2^n)$	3	400 (4.99)	1051 (7.79)	2415 (7.48)	40 (17.65)
MN	1	425 (11.55)	1075 (10.26)	2388 (6.28)	43 (26.47)
	2	403 (5.77)	1031 (5.74)	2349 (4.54)	37 (8.82)
$O(n^2)$	3	406 (6.56)	1010 (3.59)	2382 (6.01)	40 (17.65)
B-Joseph	1	423 (11.02)	1074 (10.15)	2371 (5.52)	43 (26.47)
et.al.	2	381 (0.00)	1024 (5.03)	2350 (4.58)	38 (11.76)
$O(n^4)$	3	421 (10.50)	1013 (3.90)	2346 (4.41)	40 (17.65)
SOM+MN	1	409 (7.35)	1039 (6.56)	2335 (3.92)	42 (23.53)
	2	386 (1.31)	1002 (2.77)	2302 (2.45)	38 (11.76)
$O(n^2)$	3	381 (0.00)	975 (0.00)	2247 (0.00)	37 (8.82)

TABLE IV
BIOLOGICAL SCORE AND PERCENTAGE OF IMPROVEMENT (PI) VALUE (WITHIN PARENTHESES) FOR "SOM + MN" AND SOM

		Data Sets			
Algo. & complexity	Dist.	Cell cycle	Yeast complexes	All Yeast	Herpes
SOM+MN	1	409 (13.30)	1039 (15.19)	2335 (14.40)	42 (20.00)
$O(n^2)$	2	386 (6.93)	1002 (11.09)	2302 (12.79)	38 (8.57)
	3	381 (5.54)	975 (8.09)	2247 (10.09)	37 (5.71)
SOM	1	389 (7.76)	973 (7.87)	2100 (2.89)	40 (14.29)
	2	369 (2.22)	944 (4.66)	2073 (1.57)	36 (2.86)
$O(n^2)$	3	361 (0.00)	902 (0.00)	2041 (0.00)	35 (0.00)

different types of experiments are present in a particular microarray data. For example, superior results are obtained with *Maxrange-M* for most of the available algorithms for the Cell Cycle, Yeast Complex, and All Yeast data sets (shown in first row for each algorithm in Table III). The available measures for gene distance, like Manhattan distance, Euclidean distance, and Pearson correlations, are suitable for

TABLE V
RESULTS OF t-TEST FOR DIFFERENT PAIRS OF DISTANCE MEASURES

	Pairs of distance measure	
	Maxrange-M & Pearson	*Maxrange-M* & Euclidean
t	3.4247	2.1563
p	$0.001 > p$	$0.02 > p$

TABLE VI
RESULTS OF t-TEST FOR DIFFERENT PAIRS OF ALGORITHMS

	Algorithm pairs		
	MN & Concorde	MN & B-Joseph	SOM+MN & SOM
t	0.051	0.067	4.103
p	$p > 0.5$	$p > 0.5$	$0.0001 > p$

the same type of experiments in microarray data, but they are unable to assign different weights of distance for different types of experiments. In contrast, the *Maxrange-M* and *Maxrange-E* distance provides this flexibility, and hence, better results are obtained for multiple types of experiments.

B. Statistical Analysis of Maxrange-M Distance Measure and MN Ordering Algorithm

To statistically compare the performance of *Maxrange-M* distance with Pearson correlation in the case of ordering algorithms, t-tests are performed with the PI (7) values shown within parentheses in Table III using

$$t = \frac{\overline{PI_1} - \overline{PI_2}}{\sqrt{\frac{VariancePI_1}{n_1} + \frac{VariancePI_2}{n_2}}} \quad (8)$$

where $\overline{PI_1}$ and $VariancePI_1$ are the mean and the variance of all the available PI values for *Maxrange-M* distance in Table III. PI_2 is used for Pearson correlation and $n_1 = n_2 = 16$, as there are 16 PI values available in total from Table III for each of the distance measures with four data sets and four algorithms. So, the degrees of freedom for t-test are $16 \times 2 - 2 = 30$. Similarly, t-test is also performed for *Maxrange-M* distance and Euclidean distance. The two t values and related p values are shown in Table V. The alternative hypothesis (H_1) that the average of “percentages of improvement over the lowest biological score” for the *Maxrange-M* distance is better than the related one (Pearson or Euclidean) is used in the calculation of t-statistics. After finding the p values (from t-table) for corresponding t values, we reject the null hypothesis for both cases with significance level of 0.001 and 0.02, respectively, which suggests that there is strong evidence against the null hypothesis in favor of the alternative.

Similar types of t-tests for the MN and related algorithm (Concorde or Bar-Joseph) are also performed with the percentages of improvement shown in Table III. The results are shown in Table VI. For each algorithm, there are 12 PI values (for four data sets and three distance measures), and hence, $12 \times 2 - 2 = 22$ degrees of freedom are available for each t-test. From the results of t-test and p values, the null hypothesis that “there is no difference between the averages of “percentages of improvement over the lowest biological score” for the two algorithms” is accepted for the pairs MN–Concorde and MN–Bar-Joseph. The alternative hypothesis that the average of “percentages of improvement over the lowest biological score” for “SOM + MN” is better than SOM is favored in t-test with the PI values shown in Table IV.

From the biological scores (Table III) and t-test results (Table VI), it is evident that MN provides biologically comparable gene order with respect to Concorde for all data sets and distance measure. Note that the time complexity of MN is $O(n^2)$, whereas the time complexity of Concorde is $O(2^n)$, where n is the number of genes. Therefore, it is preferable to use the MN algorithm since it has the minimum complexity. For example, MN took 0.008 s to order Yeast Complex data (979 genes) as compared to Concorde and Bar-Joseph's method that took 272 and 3.328 s, respectively.

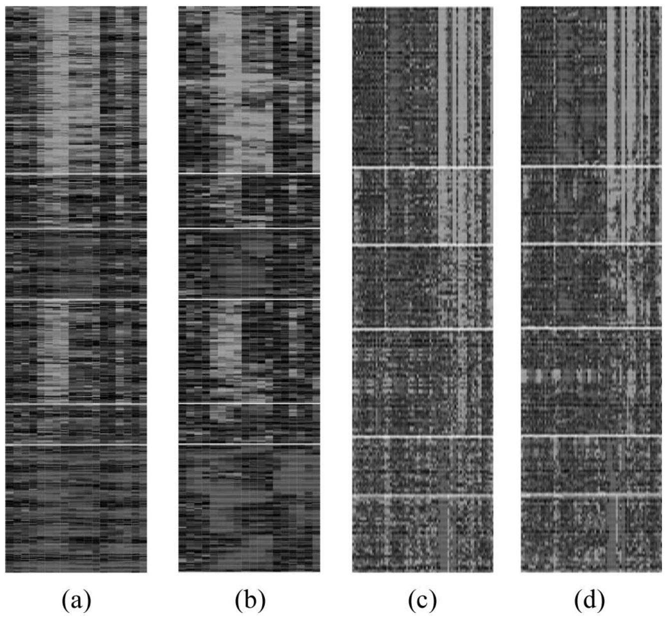


Fig. 2. Comparing SOM with “SOM + MN” for (a) and (b) Fibroblast data and (c) and (d) Yeast Complex data using *Maxrange-M* distance. The expression profiles are represented as lines of colored boxes using treeview software [1]. Some grouped genes obtained by MN [(b) and (d)] have similar expression patterns.

C. Subcluster Identification and Grouping of Correlated Genes by MN with SOM

To show how MN helps to identify subclusters within large clusters and groups functionally correlated genes within clusters to improve the solution quality of a nonhierarchical solution, MN is applied separately on the gene clusters found by SOM. The results/improvements found by combining these two algorithms are shown in Tables II, III, and VI. Here, the visual displays are presented for Fibroblast [Fig. 2(a) and (b)] and Yeast Complex [Fig. 2(c) and (d)] data. Fibroblast genes are first clustered using SOM with six nodes. A visual display of these six clusters is shown in Fig. 2(a). Observing this visual pattern, no subcluster can be identified in each cluster. After applying MN on each cluster, closely related genes with similar expressions are aligned next to each other, as shown in Fig. 2(b). Gene ordering here suggests that two or more subclusters exist at least in Clusters 1, 4, and 6, and it will be useful to increase the number of nodes of SOM to at least nine for Fibroblast data. Note that Iyer *et al.* [16] identified ten clusters of genes for these data.

The Yeast Complex data set is first clustered in 16 groups using SOM with 16 nodes. A visual display of the first six clusters/groups is shown in Fig. 2(c). When the genes are ordered in each cluster with

TABLE VII
GENE SUBCLUSTERS IN THIRD AND FOURTH CLUSTER AND THEIR FUNCTIONAL CATEGORY INDEXES FOR YEAST COMPLEX DATA. THESE SUBCLUSTERS ARE IDENTIFIED USING SOM + MN

Clu-ster	Sub-cluster	Genes	Functional index
3	1	YMR260C, YDR429C, YPL237W, YLR406C, YJR007W, YER025W, YPR041W, YDR172W, YDR211W	5
	2	YDR212W, YIL142W, YPL210C, YKL057C, YPL243W	6 and 7
	3	YLR060W, YOR260W, YDL040C, YKR026C, YLR291C, YBR142W, YBL087C, YHL001W, YDR450W, YHL033C, YBR191W, YBR189W, YBR048W, YBR118W	5
	4	YBR142W, YHR062C, YHR065C, YNR003C, YMR043W, YIL021W, YOR210W, YDR194C, YHR069C	4
4	1	YLR093C, YNL121C, YLR170C, YML112W, YBR160W, YBR171W, YLR378C, YML019W, YPL234C, YOR039W	6
	2	YKR068C, YLL050C, YGL200C, YML012W, YPL218W, YKL080W, YDR086C, YNL153C, YKL122C, YLR292C, YGL112C, YLR268W YLR447C	6 and 9
	3	YBR010W, YNL031C, YBL003C, YDR225W, YDR224C, YNL030W, YBR009C, YBL002W, YPL256C,	3, 4, and 7
	4	YJL025W, YPR101W, YMR061W, YGR195W, YOR244W, YLR105C, YDL043C, YPR056W, YPR057W	4

MN, four, four, five, and two distinct subclusters are identified using visual display in clusters 2, 3, 4, and 5, respectively. Gene names along with their functional category (indexes) for each subcluster within the third and fourth cluster are shown in Table VII. The name of the functional categories corresponding to their index is shown in Table VIII. For example, all the nine genes in the third subcluster of cluster 4 (YBR010W, YNL031C, YBL003C, YDR225W, YDR224C, YNL030W, YBR009C, YBL002W, and YPL256C) are involved in Cell Cycle and DNA processing, Transcription, and Protein with Binding Function or Cofactor Requirement. While using SOM, these genes are distributed in the cluster 4 and no subcluster can be identified. After ordering with MN, they are tightly grouped and identified easily using visual display.

VI. CONCLUSION

A new measure called *Maxrange*, for evaluating the distance between genes, and a new MN gene-ordering algorithm are described in this correspondence. These are used for efficiently ordering the genes in terms of their expression values for complete microarray data sets as well as in individual clusters found by SOM for those data sets. In *Maxrange-M* and *Maxrange-E* distance, normalization is performed separately with different normalizing factors for different types of experiment. This makes it suitable for both single type and multiple types of experiments. As a basic distance measure, Manhattan/Euclidean

TABLE VIII
FUNCTIONAL INDEXES AND CORRESPONDING FUNCTIONAL CATEGORIES

Functional index	Functional Category
1	Metabolism
2	Energy
3	Cell Cycle and DNA Processing
4	Transcription
5	Protein Synthesis
6	Protein Fate (folding, modification, destination)
7	Protein with Binding Function or Cofactor Requirement
8	Protein Activity Regulation
9	Cellular Transport, Transport Facilitation and Transport Routes

distance is used in *Maxrange* for their insensitiveness to large threefold changes in the gene expression profiles.

In MN, the repetitive searching for optimal gene order in gene groups (closely related genes) is avoided. While this results in reduced time complexity ($O(n^2)$) for MN, in terms of biological score, it is comparable with Concorde ($O(2^n)$), the best TSP solver currently available. Also, it will be computationally expensive to apply Concorde or similar local search-based evolutionary algorithms to order genes in individual clusters of a nonhierarchical clustering solution. A novel hybrid method of gene ordering in SOM and its utility in finding useful subgroups of genes within clusters is also demonstrated. Experiments for each data set are also conducted with $\sqrt{n}$ nodes for SOM. In all these cases, the cluster number increased marginally, many nodes are found with no genes associated with them, and some clusters are found where genes belong to different biological categories and cannot be identified without gene ordering.

A huge number of different types of experiment by different research groups all over the world are conducted over genes to find the functional correlation between them. In the future, more experiments are likely to be appended in the same existing microarray. This demands a distance measure like *Maxrange-M*, and a growing number of genes for the same microarray data sets require fast ordering algorithm like MN. It is evident from the experimental results that *Maxrange-M* with MN performs the best in such situations. As such, this combination seems to be a promising tool for microarray- and gene-expression-related experiments.

ACKNOWLEDGMENT

The authors would like to thank B. L. Narayan for his help in statistical analysis, anonymous reviewers for their suggestions in improving the quality of research, and Dr. P. Kellam for providing the linear dynamic range of Herpes data.

REFERENCES

[1] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein, "Cluster analysis and display of genome-wide expression patterns," in *Proc. Nat. Acad. Sci.*, Dec. 1998, vol. 95, no. 25, pp. 14 863–14 868.
[2] T. Biedl, B. Brejová, E. D. Demaine, A. M. Hamel, and T. Vinar, "Optimal arrangement of leaves in the tree representing hierarchical clustering of gene expression data," Dept. Comput. Sci., Univ. Waterloo, Waterloo, ON, Canada, Tech. Rep. 2001-2014, 2001.
[3] Z. Bar-Joseph, D. K. Gifford, and T. S. Jaakkola, "Fast optimal leaf ordering for hierarchical clustering," *Bioinformatics*, vol. 17, no. 90 001, pp. 22–29, 2001.

[4] H. K. Tsai, J. M. Yang, Y. F. Tsai, and C. Y. Kao, "An evolutionary approach for gene expression patterns," *IEEE Trans. Inf. Technol. Biomed.*, vol. 8, no. 2, pp. 69–78, Jun. 2004.

[5] C. Cotta, A. Mendes, V. Garcia, P. Franca, and P. Moscato, "Applying memetic algorithms to the analysis of microarray data," in *Proc. Evo Workshops*, 2003, pp. 22–32.

[6] D. Applegate, R. Bixby, V. Chvátal, and W. Cook, (2003), *Concorde Package*. [Online]. Available: www.tsp.gatech.edu/concorde/downloads/codes/src/co031219.tgz

[7] R. Herwig, A. J. Poustka, C. Muller, C. Bull, H. Lehrach, and J. O'Brien, "Large-scale clustering of cDNA-fingerprinting data," *Genome Res.*, vol. 9, no. 11, pp. 1093–1105, Nov. 1999.

[8] P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E. S. Lander, and T. R. Golub, "Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation," *Proc. Nat. Acad. Sci.*, vol. 96, no. 6, pp. 2907–2912, Mar. 1999.

[9] R. Sharan and R. Shamir, "CLICK: A clustering algorithm with applications to gene expression analysis," in *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, 2000, pp. 307–316.

[10] TSPLIB. [Online]. Available: http://www.iwr.uniheidelberg.de/groups/comopt/software/TSPLIB95/

[11] J. S. de Sousa, L. de C. T. Gomes, G. B. Bezerra, L. N. de Castro, and F. J. V. Zuben, "An immune-evolutionary algorithm for multiple rearrangements of gene expression data," *Genet. Program. Evol. Mach.*, vol. 5, no. 2, pp. 157–179, 2004.

[12] L. Shi *et al.*, "Microarray scanner calibration curves: Characteristics and implications," *BMC Bioinformatics*, vol. 6, no. (Suppl2):S11, pp. 1–14, 2005.

[13] Y. H. Yang, S. Dudoit, P. Luu, and T. P. Speed, "Normalization of cdna microarray data," in *Proc. SPIE—Microarrays: Optical Technologies and Informatics,* M. L. Bittner, Y. Chen, A. N. Dorsel, E. R. Dougherty, Eds., 2001, vol. 4266, pp. 141–152.

[14] G. Sherlock *et al.*, "The Stanford microarray database," *Nucleic Acids Res.*, vol. 29, no. 1, pp. 152–155, 2001.

[15] Website. [Online]. Available: http://rana.lbl.gov/EisenData.htm

[16] V. R. Iyer *et al.*, "The transcriptional program in the response of human fibroblasts to serum," *Science*, vol. 283, no. 5398, pp. 83–87, 1999.

[17] R. G. Jenner, M. M. Albà, C. Boshoff, and P. Kellam, "Kaposi's sarcoma-associated herpesvirus latent and lytic gene expression as revealed by dna arrays," *J. Virol.*, vol. 75, no. 2, pp. 891–902, 2001.

[18] *Munich Information for Protein Sequences*. [Online]. Available: http://www.mips.com

[19] T. H. Bo, B. Dysvik, and I. Jonassen, "Lsimpute: Accurate estimation of missing values in microarray data with least squares methods," *Nucleic Acids Res.*, vol. 32, no. 3, 2004. e34, pp. online.

[20] E. F. Krause, *Taxicab Geometry: An Adventure in Non-Euclidean Geometry*. New York: Dover, 1986.

[21] D. Venet, "MatArray: A Matlab toolbox for microarray data," *Bioinformatics*, vol. 19, no. 5, pp. 659–660, 2003.

Combining Multi-Source Information through Functional Annotation based Weighting: Gene Function Prediction in Yeast

Shubhra Sankar Ray[1], Sanghamitra Bandyopadhyay[2], *Senior Member, IEEE,* and Sankar K. Pal[1], *Fellow, IEEE,*
[1]Center for Soft Computing Research: A National Facility, [2]Machine Intelligence Unit,
Indian Statistical Institute,
Kolkata 700108
Email: {shubhra_r, sanghami, sankar}@isical.ac.in

Abstract

Motivation: One of the important goals of biological investigation is to predict the function of unclassified gene. An approach in this direction involves identifying the group of its closest classified genes and assigning the common biological function of the group to the unclassified gene, using different sources of information, such as microarray gene expressions, protein sequences, and phenotypic profiles. Even in a model organism like Yeast, there are more than 1000 genes with unknown biological function defined in Munich Information for Protein Sequences (MIPS) and Saccharomyces Genome Database (SGD). Although there is a rich literature on Bayesian networks and their importance in multi data source integration and gene function prediction, there is hardly any similar work in a functional annotation based weighting framework. In this investigation, we propose a new scoring framework using functional annotations, called *Biological Score* (BS), for predicting the function of some of the unclassified Yeast genes.

Methods: The *Biological Score* is computed by first evaluating the similarities between genes, arising from different data sources, in a common framework, and then integrating them in a linear combination style. We use phenotypic profiles, cDNA microarray expression, Kyoto Encyclopedia of Genes and Genomes (KEGG) pathway information, protein similarity through transitive homologues, and protein-protein interaction information as data sources. The relative weight of each data source, in the score, is determined adaptively by utilizing the information on functional annotations (Yeast GO-Slim: Process) of classified genes, available from SGD. Genes are clustered by a method called *K-BS*, where, for each gene, a cluster comprising that gene and its K nearest neighbors is computed using the proposed score (BS) and evaluated with MIPS annotation.

Results: We predict the functional categories of 417 classified genes from 417 clusters with 98.20 positive predictive value (PPV), using *K-BS* and a *P*-value cut-off 1×10^{-13}. This cut-off is then used in the method to predict the functional categories of 12 unclassified Yeast genes.

Conclusions: Our experimental results indicate that considering multiple data sources and estimating their weights with Yeast GO-Slim process annotation in the score can considerably enhance its PPV over individual data source. It has been found that even a small proportion of annotated genes can provide improvements in true positive gene pairs.

Index Terms

gene expression, protein sequence, transitive homology, phenotypic profile, combinatorial optimization, bioinformatics.

I. Background

Increasing quantities of high-throughput biological data have become available in recent years. Many of these, such as phenotypic profiles [1], gene expression microarrays [2], protein sequences [3], KEGG pathway [4], protein-protein interaction data [5], [6], protein phylogenetic profiles [7] and Rosetta Stone sequence [8] assess functional relationships between genes on a large scale. These high-throughput data can be the key to assign accurate functional annotation to a significant number of unclassified genes [9]. Microarray analysis can provide gene function prediction by analyzing coexpression relationships in a high-throughput fashion. While gene expressions and phenotypic profiles are excellent tools for hypothesis generation, they alone often lack the degree of specificity needed for accurate gene function prediction. This improvement in specificity can be achieved through the incorporation of heterogeneous functional data in an integrated analysis [9].

The value of combining informations, obtained from different methods, for gene function predictions, has been illustrated by several studies [3], [9]. Marcotte et al. [3] predicted many potential protein functions for Saccharomyces cerevisiae based on a heuristic combination of different types of data, where confidence levels for protein-protein links are defined subjectively on a case-by-case basis. Von Mering et al. [10] first developed quantitative methods to measure functional relationship among genes from three different sources of information (including gene fusion, chromosomal proximity and phylogenetic profiles) and predicted functional modules by using a clustering algorithm. In [9], heterogeneous data sources are integrated in Bayesian network approach and functional modules are predicted by using a clustering algorithm based on the principle of *KNN* algorithm. The network is constructed on some independence assumptions about different data sources and uses conditional probability tables based on information elicited from yeast experts. Lee et al. [11] compared different classes of data (including functional links extracted through literature search) and integrated them by using Bayesian Score. In this approach, all available log likelihood scores, derived from the various data sets and lines of evidences, are added to find a combined similarity. Spirin and Mirny [12] developed algorithms to analyze the structural properties of a predicted interaction network to identify the subsets of genes that are densely connected among themselves, but sparsely connected with others. Yanai and Delisi [13] predicted gene links by combining three different types of links through the union operation. The gene modules predicted by all these studies have shown some level of consistency with the well-established biological concepts as described in MIPS [14], KEGG [4], and other public data-bases.

In spite of the remarkable power and potential to address inferential processes, there are some inherent limitations in Bayesian networks. The problem centers around the quality and extent of the prior beliefs used in Bayesian inference processing. A Bayesian network is only as useful as the reliability of this prior knowledge. An excessively optimistic or pessimistic expectation of the quality of these prior beliefs will distort the entire network and invalidate the result.

While most of the works regarding data source integration are based on Bayesian network, the approach of integrating information from data sources in a linear combination style through functional annotation based weights, is still unexplored. Moreover, all the pre-mentioned works do not incorporate transitive nature of protein homology and KEGG pathway similarity extraction excluding Yeast genes. In this investigation, we present a new computational framework, using functional annotation based weighting, for the prediction of gene function in yeast through phenotypic profile similarity, gene expression similarity, protein similarity by transitive homology, Kyoto Encyclopedia of Genes and Genomes (KEGG) pathway similarity, interacting protein information and evaluation of these information by MIPS [14] gene annotation. The novelty of our method lies in the way of estimating the weights in a linear combination style, using gene annotations in Eq. 7 (described in Section II-C). In a related work, Lee et al. [11] used a single free parameter for estimating weights where they pointed out that a heuristic modification to the strict Bayesian approach performs better for integrating the diverse functional linkage data sets by incorporating the relative weighting of the data (see supplementary material of [11]). In this approach, all available log likelihood scores derived from the various data sets are added with a rank-order dependent weighting scheme. The resulting weighted sum (WS), scoring the functional linkage between a pair of genes, is calculated as:

$$WS = \sum_{i=1}^{n} \frac{L_i}{D^{i-1}}, \tag{1}$$

where L represents the log likelihood score for the gene linkage from a single data set, D is a free parameter (weight) roughly representing the relative degree of dependence between the various data sets, and i is the rank index in order of descending magnitude of the n log likelihood scores for the given gene pair. The free parameter D ranges from 1 to α, and is chosen to optimize overall performance (positive predictive value (PPV) and coverage) on the functional benchmark. In this method it is not possible to have equal weights for any two data sources for $D > 1$. Moreover, here the weights follow a strict geometric series, which in most cases will not reflect the relative importance of the data sources. All these limitations are not present in our proposed scheme.

II. Methods

We mainly focus on integrating phenotypic profiles, microarray gene expression, KEGG pathway related protein database in Protein Information Resource (PIR) [15], protein sequence similarity by transitive homology, and protein-protein interaction information as data sources. The main steps of our methodology for predicting gene functions can be summarized as:

i) extract pairwise similarity of genes, obtained from different data sources (see Section II-A);

ii) separately re-score the similarities in a common framework of Yeast GO-Slim: Process annotations (see Section II-B);

iii) integrate the re-scored similarities from different data sources through the proposed scoring framework (see Section II-C) and calculate the combined score;

iv) for each gene g, form a cluster comprising that gene and its K nearest neighbors using the proposed score and predict the function of g by noting the functional enrichment of the cluster using MIPS annotation (see Section II-D).

Each of the above steps are discussed in detail in the following subsections. In brief, in the proposed scoring framework, the weights of the re-scored similarities from different data sources are determined by adaptively maximizing the PPV of the score, using Yeast GO-Slim process annotations [16] of known genes. The weighting scheme enables all possible weighting (including equal and zero weighting) of data sources by first assigning each data source a weight that varies from 0 to α and then optimizing the weights using an objective function, involving the PPV between gene pairs using Yeast GO-Slim process annotation. The aim is to predict gene function from clustering solutions, rather than obtaining detailed interaction relationships among the genes. Our method predicts the functional categories of 12 unclassified Yeast genes from 12 clusters with 98.20 PPV. We also observed that even a small proportion of classified (annotated) genes can provide improvements in predicting true positive gene pairs. Moreover, for evaluating the results further, we merged the available annotations from Yeast GO-Slim process and MIPS for all the genes, and then split the genes into independent training and test sets. The training set is used to determine the weights, while the independent test set is used to compute the PPV and to evaluate the gene pairs and clustering results. The process is repeated 10 times and the cross-validation result is reported.

A. Data Sources and Similarity Extraction Techniques

Here we describe the different data sources and their respective similarity extraction techniques.

1) Phenotypic Profile: Recently, Brown et al. [1] presented a method for the global analysis of the function of genes in budding yeast. The method is based on hierarchical clustering of the quantitative sensitivity profiles of the 4756 strains with individual homozygous deletion of all nonessential genes, with each gene replaced by a cassette containing a 20-mer molecular barcode' unique for each deletion mutant. They showed the method to be superior than other global methods for identifying function of genes involved in various DNA repair, damage checkpoint pathways, and other interrogated functions. Analysis of the phenotypic profiles of the 51 diverse treatments places a total of 860 genes of unknown function in clusters with genes of known function. We use this complete phenotypic profile data for quantitative phenotypic profile similarity extraction with Pearson correlation [1].

Let $X = x_1, x_2, \cdots, x_k$ and $Y = y_1, y_2, \cdots, y_k$ be the phenotypic profiles of two genes obtained over a series of k different treatments. Using centered Pearson correlation, the similarity between genes X and Y is defined as

$$Pc_{X,Y} = \frac{1}{k}\sum_{i=1}^{k}\left(\frac{x_i - \overline{X}}{\sigma_X}\right)\left(\frac{y_i - \overline{Y}}{\sigma_Y}\right) \tag{2}$$

where $\overline{X}$ and σ_X are the mean and standard deviation of the gene X, respectively. σ_X is defined as

$$\sigma_X = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(x_i - \overline{X})^2}. \tag{3}$$

The Pearson correlation has value between -1 and 1, where 1 indicates a linear relationship between the two vectors.

When the phenotypic profile data set is downloaded from the website it is found that out of 51 treatments some treatments are missing for some genes (strains). For the subsequent analysis to be as informative as possible, it is essential that the missing values have to be estimated in order to analyze the available data and the estimates for the missing values are as accurate as possible. Currently, there is no state-of-the-art missing value estimation method for phenotypic profiles. Alternatively, missing values in phenotypic profiles can be estimated using the methods that are used for microarray gene expression. In this phenotypic profile data set, all the genes with more than 50% missing values are first eliminated from the dataset. Thereafter for the remaining genes missing values are estimated using LSimpute [17] software, a state-of-the-art statistical java based package to estimate missing values in gene expression data set. In LSimpute software, two basic methods based on least squares principle, one utilizing correlations between genes (LSimpute_gene) and the other utilizing correlations between arrays (LSimpute_array), are used to estimate missing values. A robust method (LSimpute_adaptive) using weighted averages of the estimates from LSimpute_gene and LSimpute_array for adaptive estimates is also available in LSimpute and used in this investigation.

Phenotypic profile data is succeptable to biases created during the PCR amplification reaction. The detailed procedure of generating and normalizing the data is available in Brown et al. [1]. In brief, each gene (strain) in phenotypic profile data is associated with four hybridization signals on the high-density oligonucleotide array generated in two separate PCR labeling reactions [1]. The data is then normalized by Brown et al. in the experimental array to that of the control array in order to eliminate any bias created during the PCR amplification reaction. The normalized data is downloaded from the supplimentary material of [1] and used in this investigation.

2) Gene Expression: We use the All Yeast [2], [18] data for gene expression similarity extraction. Brown et al. [1] have shown that even with 30 distinct biological conditions for gene expression, GO term ribosome biogenesis (GO:0007046) tends to dominate gene pairs implicated by coexpression. As we have already used phenotypic profiles, which implicate gene relationships over a broad range of biological processes, here we only use the widely studied All yeast data.

A number of measures in finding the microarray gene expression similarity can be used for gene annotation and grouping. The most popular and probably most simple measures for finding global similarity between genes are the Pearson correlation [9], a statistical measure of (linear) dependence between random variables, and the Euclidean distance [3]. We use centered Pearson correlation for extracting gene expression similarity as mentioned in the previous section.

To identify relationship among genes, involved in multiple biological functions or processes, many microarray experiments with different biological origins are conducted. These experiments with multiple microarray slides are sources of non-biological variation between slides such as dye biases, sample preparation or hybridization differences, scanner calibrations, slide printing variations, volume of initial RNA, etc. Some of these variabilities can be corrected by data normalization before analysis of the data. Normalization can be performed by removing saturated signals from microarray, background correction, low expression genes correction, etc. In cDNA microarray related investigations, many different methods are developed in order to compensate for dye-effects and other non-biological variations between arrays. The All Yeast dataset is downloaded from Stanford Microarray Database [19] with default parameters. The default parameters include all types of normalization applicable to that data and suggested by the experts.

Microarray experiments often produce multiple missing expression values, normally due to various experimental problems. As gene expression analysis generally requires a complete data matrix as input, the missing values have to be estimated in order to analyze the available data. Alternatively, genes with missing expression values can be removed until no missing values remain. However, for arrays with only a small number of missing values, it is desirable to estimate those values [17]. For the All Yeast data, we estimated the missing values using LSimpute_adaptive [17] in a similar fashion to phenotypic profiles, mentioned in Section II-A.1.

3) KEGG pathway: The pathway information for genes in KEGG [4] can be utilized as a reference for functional reconstruction. All the protein sequences, except Yeast proteins, corresponding to each pathway (121 pathways in the second level) are downloaded from PIR [15]. Profile vector for each protein in Yeast is computed by comparing its sequence across 121 pathway databases, using BLAST [20]. The method is similar to phylogenetic profile [7] construction, where, each pathway database is replaced by all proteins within a species. The pathway profiles of genes, computed using KEGG pathway databases, are denoted as KEGG profiles.

To find the similarity between two genes using KEGG profiles, we used the ratio of dot product value and OR value between two profiles. The similarity matrix has a highest similarity value of 1. Hence, the similarity values, obtained by all pair-wise comparison, have a dynamic range from 0 to 1 and its normalization is unnecessary. Note that, the genes, whose protein sequences are not available, are assigned a pathway profile similarity value of 0 w.r.t. all other genes (proteins).

4) Protein Sequence: Comparing the protein sequences presents an alternative prominent approach for gene annotation and analysis. Sequence-based comparative analysis also proved crucial for deciphering functions of genes and proteins. As the proteins are products of coding regions (open reading frames) of the genes, the integration of expression data similarity with protein sequence similarity for gene analysis could potentially provide new insights into the relation between gene functions. Protein similarity information is one of the major components of biological knowledge, which contains mostly known and validated protein (or gene) relations. Intuitively one can assume that all the protein relations arising from direct protein similarity search is available in the literature and will not help in predicting functions for unclassified genes in a widely studied organism like Yeast. As compared to direct protein similarity search, the field of searching gene/protein similarity through phylogenetic profiles (PP) [7], Rosetta Stone sequence (RS) [8], and transitive homology [21] are relatively new methods and with increasing number of fully sequenced genomes the search space of these methods are increasing rapidly. In this investigation, transitive homologues are used instead of PP and RS, for extracting protein similarity, as its accuracy is reported to be higher than PP and RS in literature [22], [23]. To detect transitive homologues by the third intermediate sequence, 37,66,477 protein sequences are downloaded from UniProt [24].

Transitive homology detection method [21]–[23] works by searching the query sequence against the database with a conservative threshold to find the closely homologous sequences and using these homologous sequences as seeds to search the database to find remotely homologous sequences with a less conservative threshold. The method has been shown to be close to the profile [7] based methods and better than a direct pairwise homology search [21]. Our findings are in agreement with [22] that, this homology transitivity can be used as the main source for gene pairing and predicting functions of unknown genes. To find the transitive homologues, homology comparisons are performed among target proteins and 37,66,477 proteins downloaded from UniProt [24], by using BLASTP in BLAST [20]. Before comparison all the yeast proteins are removed from the downloaded database. Let the similarity between two protein sequences A and B be $B_{A,B}$. The value $B_{A,B}$ is replaced by $B_{A,C} \times B_{C,B}$ if there exists a sequence C such that $B_{A,C} \times B_{C,B}$ is larger than the current value of $B_{A,B}$. This transformation takes advantage of the transitive homology of sequences A and B through the intermediate sequence C, assuming that sequences A and C and sequences B and C are independently homologous [23]. For example, consider the transitive homology between sequence a and sequence b through the third sequence c. The E-values between sequence a and sequence c, sequence c and sequence b, as well as sequence a and sequence b are 0.01, 0.005, and 20 respectively. The protein similarities $B_{a,c}$, $B_{c,b}$, and $B_{a,b}$ are 0.8, 0.9, and 0.2 respectively. The homology between sequence a and sequence b cannot be detected with their direct E-value. However, the value of $B_{a,b}$ is assigned to $0.8 \times 0.9 = 0.72$ because of the transitive sequence homology.

Instead of storing raw BLAST score as the similarity between two protein sequences, we use the metric of ProClust [25] where the metric value scales from 0 to 1. It is the ratio of the raw BLAST score of the sequence alignments to the raw BLAST score of one of those two sequences aligned to itself. The transitive protein similarity value also scales from 0 to 1 and its normalization is unnecessary. Here also the genes, whose protein sequences are not available, are assigned a transitive protein similarity value of 0 w.r.t. all other genes.

5) Protein-Protein Interaction: Protein-protein maps promise to reveal many aspects of the complex regulatory network underlying cellular function [10]. For this study, manually curated catalogues of known protein-protein interactions are downloaded from BioGRID [6] and binary interactions are used as the common unit of analysis. For a given pair of genes/proteins the similarity value is 1 or 0, indicating a interaction present or absent, respectively. Since the similarity value scales from 0 to 1, its normalization is unnecessary. The BioGRID database/catalogue includes more than 90000 interactions by combining results obtained from synthetic lethality, affinity capture, two-hybrid, epistatic miniarray profile, reconstituted complex, co-crystal structure, co-purification, dosage rescue, phenotypic enhancement, phenotypic suppression, synthetic growth defect, co-fractionation, biochemical activity, synthetic rescue, and protein-peptide based experiments. The related references are available in BioGRID.

B. Scoring the Similarities in a Common Framework

Our working hypothesis is that each set of data has an intrinsic error rate and a limited coverage but informs us to some extent about the tendency for genes to operate in the same cellular systems and biological processes in the cell. We can therefore construct a more accurate and extensive functional coupling between yeast genes across a broad set of data (experiments). The prerequisite of this strategy is that we have a unified scoring scheme for testing the heterogeneous data sets, even when the data sets are accompanied by their own intrinsic scoring schemes (such as Pearson Correlation for phenotypic profile and gene expression). This re-scoring by a single criterion allows us to directly measure the relative merit of each data set, and then to integrate the data sets with weights that reflect this merit. In this regard, the similarities arising from various heterogeneous data sources are separately re-scored, based on the common framework of Yeast GO-Slim process annotations of genes in the SGD database [16]. Genes/proteins that occur in the same process are presumed to be functionally linked. The proportion of true positive (TP) gene pairs at a particular similarity value (computed from a data source) can be used as a single criterion for re-scoring the similarity values, where TP gene pairs are defined as pairs of genes i and j, such that genes i and j have an overlapping (explicit or implicit) GO (Gene Ontology) term annotation. In [9] proportion of TP pairs (positive predictive value (PPV)) is defined as

$$PPV = \frac{no.\ of\ pairs\ predicted\ by\ method\ that\ share\ common\ GO\ term\ assignment}{total\ no.\ of\ pairs\ predicted\ by\ method}. \tag{4}$$

The hierarchical nature of GO and multiple inheritance in the GO structure can lead to evaluation problems if we consider only the particular GO term with which a gene is annotated [9]. To alleviate this problem, we consider the SGD Yeast GO-Slim process annotations, where every gene is annotated in the same level without any tree based structure. For every gene g, that has undergone Yeast GO-Slim process annotation, a vector

$$V(g) = (v_1, v_2, \cdots, v_j) \tag{5}$$

is used to represent its category (Yeast GO-slim process) status, where j is the number of categories. The value of v_j is 1 if gene g is in the jth category; otherwise is zero. Based on the information about categorization, the positive predictive value (PPV) at a given similarity value, can be defined as

$$PPV = \frac{\sum_{i=1}^{n}\sum_{m=1}^{j}(V(g_i)_m \times V(g_{ir})_m)}{n}, \tag{6}$$

where $\sum_{m=1}^{j}(V(g_i)_m \times V(g_{ir})_m) = 1$ if $\sum_{m=1}^{j}(V(g_i)_m \times V(g_{ir})_m) \geq 1$, g_i and g_{ir} form a gene-pair, n is the number of annotated gene pairs at a given similarity value, and $V(g_i)_m$ represents the m^{th} entry of vector $V(g_i)$.

The PPV can be interpreted as being proportional to the accuracy of the data sources and their ability to predict the cellular/biological processes involved at a given similarity value. In PPV a gene pair is considered as a predicted pair if both the genes in the pair are classified in Yeast GO-Slim process. According to Yeast GO-Slim process and MIPS, there are 6069 and 6131 annotated genes (ORFs) for yeast of which 4387 and 4737 genes, respectively, are classified to some biological or functional process and the remaining genes are unclassified.

Figure 1 compares the similarity values obtained from different data sources in terms of their PPV. The PPV for intermediate similarity values, that are not plotted in Fig. 1, are calculated from the slopes of the respective curves. The similarities extracted from protein-protein interactions are binary relations in our study. Therefore, PPV for protein-protein interactions has a constant value 0.69 at a similarity value of 1 and hence it is not shown in Fig. 1.

C. New Framework for Data Source Integration

As the similarities computed from different data sources are re-scored (see Section II-B) on a single criterion and common framework of Yeast GO-Slim process annotations, they are directly comparable and can be integrated even when the natures of experiments are distinct (e.g., comparing phenotypic profiles to protein-protein interactions). The PPV reflect the accuracy of similarity values, but do not provide any information about importance/weight of one data source in presence of the other data sources, in predicting gene pairs. Consequently, it will be more appropriate and better if

1) PPV of each data source, in presence of other data sources, is separately weighed by a factor and then integrated;
2) factors are dependent on the PPV of the integrated PPV of different data sources.

Such an attempt is made in this article with a new score where, PPV computed from phenotypic similarity (Pp), gene expression similarity (Pm), KEGG pathway profile similarity (Kp), protein similarity through transitive homologue (B), and protein-protein interaction information (I) between two genes X and Y are integrated through weights a, b, c, d, and e in a linear combination style. This score is referred to as $Biological\ Score$ (BS) and is defined as

$$BS_{X,Y} = \frac{a \times Pp_{X,Y} + b \times Pm_{X,Y} + c \times Kp_{X,Y} + d \times B_{X,Y} + e \times I_{X,Y}}{a+b+c+d+e} \tag{7}$$

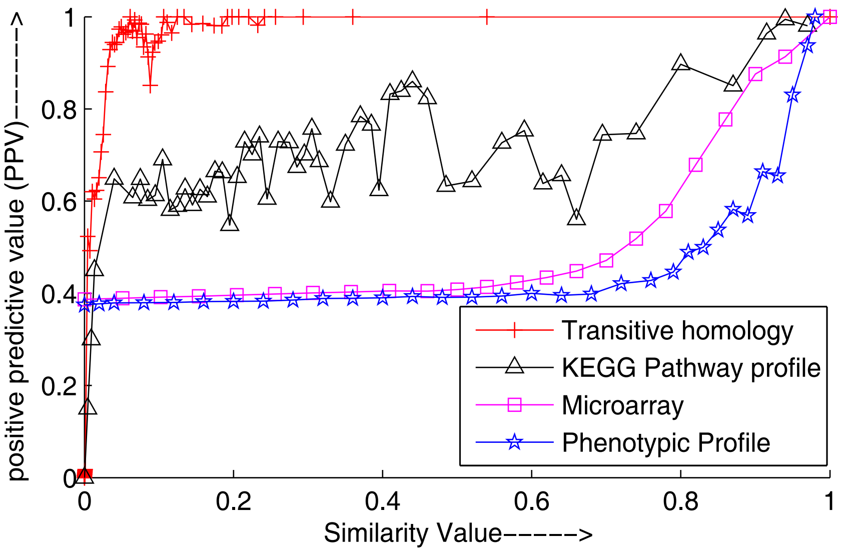


Fig. 1. Comparing the re-scored similarity values for different types of data sources to obtain equivalency in the common framework of Yeast GO-Slim process annotations. The positive predictive values (PPV) versus the similarity values are plotted for each data source.

where a, b, c, d, and e are varied within range 0 to α in steps of 1 to find a combination that maximizes the PPV for a user defined number of top gene pairs. Note that, the weights a, b, c, d, and e are assigned to the complete PPV matrices calculated from individual data sources. The following can be stated about the score:

1) $0 \leq BS_{X,Y} \leq 1$
2) $BS_{X,Y} = BS_{Y,X}$ (symmetric).

The proposed scoring framework for data source integration, in Eq. 7, is based on data source weighting where the re-scored similarity spaces, available from different data sources, are adaptively transformed using a set of weighting coefficients. Intuitively, more important similarity spaces should be assigned larger weights than less important ones, while irrelevant ones should be assigned zero weight. Although the proposed framework has some common working principle with feature weighting (FW) [26], it cannot be categorized as FW because what is computed using BS is the pair-wise gene similarities and not the set of features of any individual gene.

Estimation of Weights for Maximization of PPV: We maximize the PPV, using Yeast GO-Slim process annotations, for top gene pairs by varying the weights a, b, c, d, and e in the BS (Eq. 7). For each set of values of a, b, and c, the top gene pairs are identified with a gold standard cut-off value. Our gold standard cut-off value and gold standard of top gene pairs are determined from KEGG pathway profiles, which provides 26432 gene pairs with similarity value 1 and constant PPV of .81. These gene pairs are the most accurate of all, whereas the accuracy (PPV) of other data sources, as well as gene pairs below top 26432 for KEGG pathway profiles, vary considerably. We now use the following steps to estimate the weight factors a, b, c, d, and e in the $Biological\ Score$:

Step 1) All the factors are assigned an initial value of 1.
Step 2) BS values are calculated for all the gene pairs and sorted in descending order to identify the cut-off value above which the top 26432 gene pairs are available.
Step 3) PPV is calculated for the top 26432 gene pairs.
Step 4) The weight factors are now varied in steps of .1 and the steps from 2 to 3 are repeated to find a combination of weights for which the PPV is maximized.

Figure 2 shows how PPV, using Yeast GO-Slim process, varies for different values of weight factors ranging from 0 to 100, in steps of 1. The curves show instances where one weight factor is varied and the other weight factors are kept constant. Experiments are also conducted by excluding the KEGG pathway profile database and the corresponding curves are reffered to as $c = 0$.

D. Gene Function Prediction

For biological function prediction of each gene, a cluster comprising that gene and its K nearest neighbors is computed using the proposed score (BS). The function for each gene is predicted from the top K neighbors and selecting a gold standard BS cut-off value obtained from KEGG pathway profiles using MIPS October 2005 classification. The gene clustering method is denoted as *K-BS*, where each gene is considered once for its function prediction and allows its neighbor genes to be a member

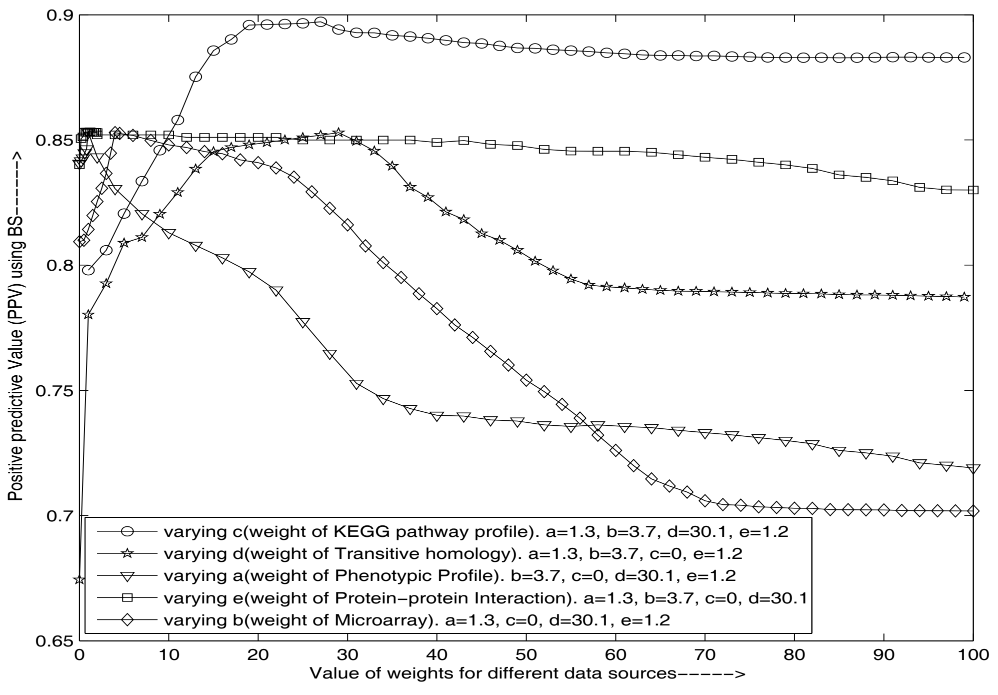


Fig. 2. Comparing the values of PPV using BS, by varying weights of PPV of different data sources for top 26432 gene pairs. When a particular weight is varied the other weights are kept constant at the values shown in the figure. The curves obtained with c=0 indicate that KEGG pathway profile is excluded in the integration process.

of multiple gene clusters. This clustering method, based on K nearest neighbors of each gene, is already used in previous related investigations of Marcotte et al. [3] and Troyanskaya et al. [9]. As Yeast GO-Slim process annotations was used for determining the weights of the data sources, 510 different MIPS (October 2005) functional categories are used to evaluate the biological significance of the clusters generated by our *K-BS*. One or several predominant functions are then assigned to each cluster and the target gene (the gene whose K nearest neighbors, using BS as a similarity value, are considered to form the cluster) by calculating the P-values for different functional categories. The probability (P-value) of observing at least m genes from a functional category within a cluster of size n is given by

$$P = 1 - \sum_{i=0}^{m-1} \frac{\begin{pmatrix} f \\ i \end{pmatrix} \cdot \begin{pmatrix} N-f \\ n-i \end{pmatrix}}{\begin{pmatrix} N \\ n \end{pmatrix}} \tag{8}$$

where f is the total number of genes within a functional category and N is the total number of genes within the genome (6131).

III. Results

As Yeast GO-Slim process was used for determining the weights of the data sources, MIPS annotation is now used to evaluate the performance of *BS*. Genes and their corresponding proteins are denoted by different symbols or identifiers in different data sources. data source integration requires that all genes/proteins are denoted according to a common naming scheme. We mapped genes from different resources to their MIPS identifier. Genes/proteins that could not be mapped to their MIPS identifier are eliminated. Our gold standard PPV of top gene pairs is now changed and determined from KEGG pathway profiles, which provides 26432 gene pairs with constant PPV of .8874, using top level classification of MIPS annotation. In this section, first we present the comparisons of our method with Lee et al.'s [11] probabilistic network and individual data sources in Section III-A. Influence of number of classified genes on the proposed scoring framework is demonstrated in Section III-B. In Section III-C various paremeters involved in the clustering method and the biological significance of some clusters are addressed. Finally, the performance of *BS* and some comparisons based on independent training (estimating weight factors) and test set with null intersection are presented in Section III-D.

A. Comparative Performance of Methods and Data Sources

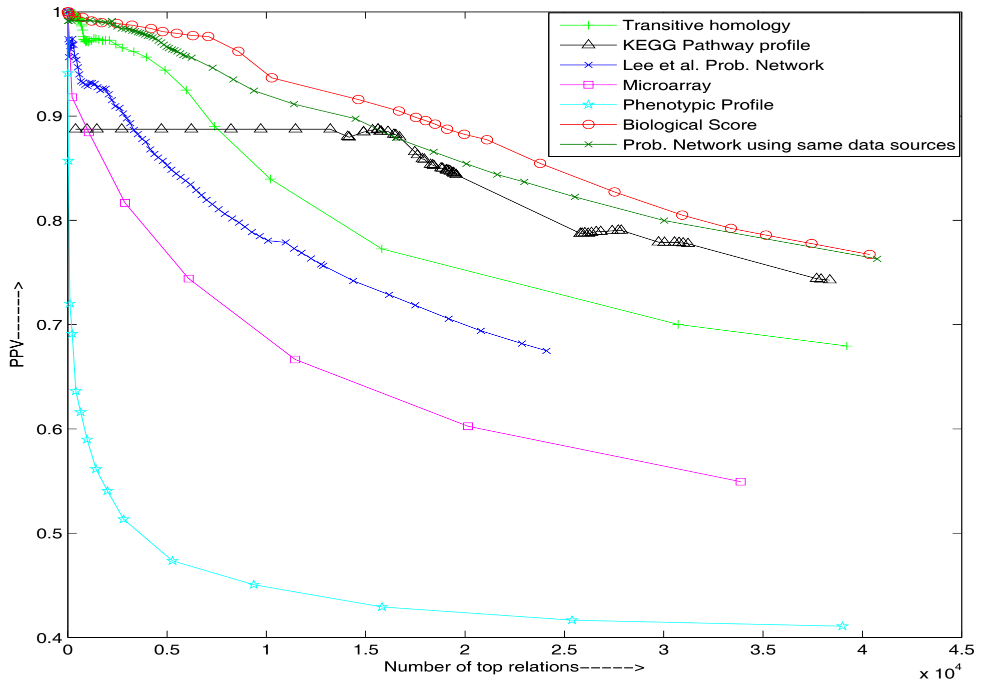


Fig. 3. Comparison between the $Biological\ Score$ (BS), Lee et al.'s Probabilistic Network, and individual data source in terms of PPV versus the number of top gene pairs. While, the available annotations (using vector $V(g)$ in Eq. 5) from Yeast GO-Slim process is used to train the weighting factors in BS and 'Probabilistic Network using same data sources', the available annotations from MIPS are used to evaluate (using PPV) the gene pairs of all the methods and data sources.

In order to demonstrate the power of data source integration, we compare the PPV of gene pairs identified by the BS (Proposed scoring framework for data source integration) with those identified by the individual data sources. Since the proposed method (BS) uses GO annotations for adapting its weights, it is not used for performing the comparisons. Rather, the MIPS annotation of classified genes is used (Fig. 3). We sorted the similarity values computed from $Biological\ Score$ (BS), phenotypic profiles, gene expression, KEGG profiles, and protein similarity from transitive homology in descending order, and drew a curve for top gene pairs verses PPV from the sorted data for each form of data source. In contrast, PPV for protein-protein interactions has a constant value of 0.69 and not shown in Fig.3. We found that the curve of BS is above the other curves. Moreover, the top 26432 gene pairs has an PPV greater than the gold standard KEGG pathway profiles. The gene pairs are also reasonably distinct from gene pairs of KEGG pathway profiles. It demonstrates that the proposed $Biological\ Score$ achieved higher PPV by combining similarities from multiple sources. Similarities supported by diverse forms of sources are more likely to be correct. This highlights the merit of data source integration. Figure 3 also compares the performance of BS and 'final log likelihood scores' of Lee et al.'s probabilistic network (downloaded from the website mentioned in [27]) in terms of PPV with MIPS annotation. The curve of Lee et al.'s probabilistic network is drawn from top 34,000 gene pairs, as mentioned in [11]. For a direct comparison between our method and the probabilistic network, we implemented the probabilistic network as described in Lee et al. using the same datasources as in Biological Score (BS) and plotted the respective curve in Fig. 3. From the figure it is clear that the top gene pairs identified in this investigation is better than any other existing network or data sources. The above statement is true not only for gold standard 24632 gene pairs but also for top 80000 gene pairs which can be used further for any gene network or gene function prediction. We found that beyond top 80000 gene pairs the performance of our method is gradually converging to the performance of probabilistic network (with same data sources) but, it does not hampers the superior performance of our method as only a fraction of top gene pairs are generally used [11] for gene function or network prediction. It is also evident from the results that the choice of data sources is a very important factor in data source integration. For example, protein homology and KEGG profile individually performs better than probabilistic network and considered as two important data sources in the proposed BS. The top $1,00,000$ gene pairs predicted by our method with PPV above 0.755 (not shown in the data) are available in http://www.isical.ac.in/~scc/Bioinformatics/AdS/toprelation.txt in tabular (tab delimited) form. The PPV computed from individual data source are also shown in the file.

B. Influence of Number of Classified Genes on Functional Annotation based Weighting

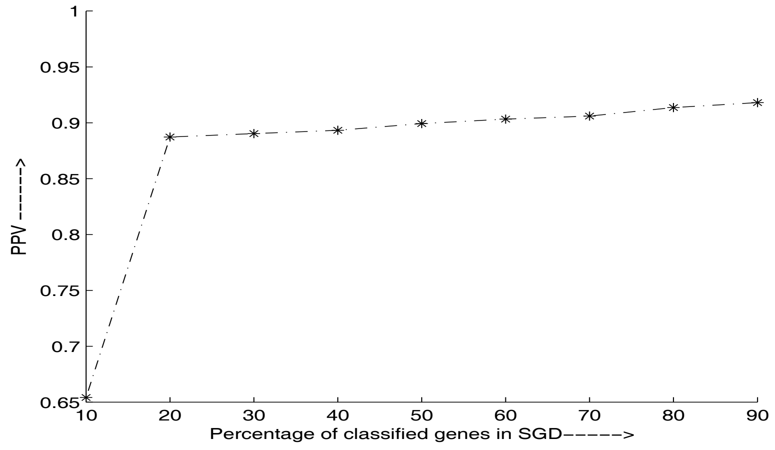


Fig. 4. Variation of PPV, using *BS*, with nine different percentages of classified genes.

Here we study how the increase in the number of classified genes in Yeast GO-Slim affects the PPV for the classified genes in MIPS for top 26432 gene pairs using *BS*. We found that even with 20% of classified genes the estimated values of a, b, c, d, and e, in maximizing PPV, differs by an amount of 1 than the estimated values with 90% of classified genes. Hence, the value of PPV also varies by an amount of 0.02 to 0.03 with classified genes ranging from 20% to 90%. Fig. 4 shows that the percentage of classified genes clearly has a limited contribution to the accuracy (PPV) of the *BS*. Thus *BS* may also be successfully used for organisms where the number of classified genes is as low as 20%.

C. Gene Function Prediction based on Clustering Results

Genes (open reading frames) are considered to be linked if they are among the 10 closest neighbors within a given distance or similarity cut-off [3]. The biological function for each gene is predicted from the cluster consisting the top 10 neighbors of that gene by selecting K to be at most 10 and BS cut-off value of 0.77. Above this cut-off value the gold standard PPV of 0.8874 is achieved for 36033 gene pairs using the MIPS October 2005 classification. We found several clusters to be significantly enriched with genes of a similar function. Clusters with P-values greater than 10^{-5} are not reported.

To predict a genes function from it's neighbor genes we use the following steps:

Step 1) 2507 clusters are identified with at-least three or more members by selecting $K = 10$ and with BS gold standard cut-off value 0.77.

Step 2) Out of these clusters, 1915 clusters are identified with functional enrichment in one or more categories and P-values less than 10^{-5}.

Step 3) From functionally enriched clusters we predict the functions of 1855 classified and 60 unclassified genes by assigning the function related with the smallest P-value. This ignores the possibility that a gene may be assigned more than one highly significant function, but in practice resulted in more accurate predictions than if multiple functions are allowed per cluster.

The functions of 1855 classified genes are predicted with 95.16 PPV. In general we can say that the possibility of 60 unclassified Yeast genes to match with the predicted functions is 95.16%. The functional enrichment, in one or more categories, for clusters intended for 60 unclassified yeast genes are available in tabular form (tab delimited file) at http://www.isical.ac.in/~scc/Bioinformatics/AdS/unclassifiedprediction.xls. The function with the smallest P-value in the table represents the predicted function for the unclassified gene, and the three values in the parenthesis denote the function related P-value, function related no. of genes in the cluster, and the function related no. of genes in the genome, respectively. The table also includes all the genes within each cluster, the PPV (between target gene and the neighbor gene) arising from various data sources, and the BS values. A table with similar format, containing the predicted functions of 1855 classified yeast genes is available at http://www.isical.ac.in/~scc/Bioinformatics/AdS/classifiedprediction.xls.

Out of 60 unclassified genes, YEL041W and YDR459C are now (April 2007) classified in MIPS, and our function predictions for these two genes are in agreement with present MIPS classification. YEL041w and its four neighbors YJR049C, YPL188W, YDR226W, and YER170W form a cluster. From the functional enrichment of the cluster we predict that YEL041w is related with the category 'phosphate metabolism' as the four remaining genes belong to this category. The prediction is right according to MIPS (April 2007) classification with p-value 1.42×10^{-6}. We further manually analyze the cluster and predict that the gene YEL041w may be related with category 'metabolism of vitamins, cofactors, and prosthetic groups' and 'homeostasis' since it's two top neighbor genes YJR049C and YPL188W is related with these categories. While the prediction of the category 'metabolism of vitamins, cofactors, and prosthetic groups' is a correct (MIPS 2007) one, the prediction 'homeostasis', may be a novel one for YEL041w. Moreover, according to MIPS, YEL041w has the highest similarity to YJR049C, which is related to homeostasis of metal ions (Na, K, Ca etc.).

The cluster containing gene YDR459C and its ten neighbor genes, YOL003C, YNL326C, YLR246W, YPR193C, YIR042C, YMR127C, YNL035C, YBL052C, YDR126W and YPR051W shows functional enrichment in categories 'protein modification' (8 out of 11, P-value 1.16×10^{-6}), 'modification with fatty acids (e.g. myristylation, palmitylation, farnesylation)' (4 out of 11, P-value 2.3×10^{-7}) and 'modification by acetylation, deacetylation' (4 out of 11, P-value 4.4×10^{-6}). We correctly predict that YDR459C is related to 'modification with fatty acids'. The hierarchical nature of MIPS annotation automatically ensures that YDR459C is related to 'protein modification', which is placed at one level higher than 'modification with fatty acids'. Although the remaining function, 'modification by acetylation, deacetylation', is also significant in terms of P-value, YDR459C is not related with this function and our results are in agreement with our approach of considering the function involving the lowest P-value. The cluster also contains three unclassified (MIPS 2007 classification) genes YDR126W, YIR042C, and YNL035C. Although the cluster is not intended to predict the function of these three genes we can assume that these genes may be related with the function 'protein modification'.

TABLE I
TOP 12 FUNCTION PREDICTIONS OF UNCLASSIFIED GENE AT BS CUT-OFF VALUE OF 0.77

Unclassified Gene	Functional category	P-value	Genes within cluster	Genes within category
YIL080W	ABC transporters	2.2204e-16	8	28
YLR057W	modification with sugar residues	2.2871e-14	8	67
YHR218W	DNA topology	0	9	52
YHR219W	DNA topology	0	10	52
YIL170W	C-compound and carbohydrate transport	1.3656e-14	8	63
YDR441C	purin nucleotide/nucleoside/nucleobase metabolism	6.7724e-15	8	58
YCL074W	TRANSPOSABLE ELEMENTS, VIRAL AND PLASMID PROTEINS	3.3307e-16	8	34
YBL112C	DNA topology	0	10	52
YLR464W	DNA topology	2.6645e-15	8	52
YMR010W	modification with sugar residues (e.g. glycosylation, deglycosylation)	0	9	67
YIL067C	vesicle fusion	2.2204e-16	9	32
YHR049W	metabolism of secondary products derived from glycine, L-serine and L-alanine	3.3307e-16	7	19

Our top predictions consist the function of 12 unclassified (MIPS 2007) and 417 classified genes at BS cut-off value 0.77, and P-value cut-off 1×10^{-13}. At these cut-off values, the functions of the classified genes are predicted with 98.20 PPV. Table I summarizes the top 12 predicted functions for 12 unclassified genes. For each of the predicted functions, the related p-values, no. of related genes in the cluster and the genome, is also shown in the table. Each of the clusters contain 11 genes and they are available in the table representing 60 clusters for function prediction of unclassified genes. since four of the 12 clusters show functional enrichment in a single category of 'DNA topology', we analyze these clusters manually. We observe that 15 classified (YBL113c, YDR545w, YEL077c, YER190w, YGR296w, YHL050c, YIL177c, YJL225c, YLL066c, YLL067c, YLR466w, YLR467w, YNL339c, YPL283c, and YPR204w), 4 unclassified (YHR218W, YHR219W, YBL112C, and YLR464W) and 2 recently deleted (YEL076C and YPR203W) genes are distributed in these clusters with 80% genes in common. We further perform clustering with *K-BS* by selecting $K = 20$ to find if these four clusters merge to form a single cluster. After clustering, all the 21 genes are found in the same cluster, which shows functional enrichment in categories 'CELL CYCLE AND DNA PROCESSING' (15 out of 19, P-value 1.53×10^{-10}), 'DNA processing' (15 out of 19, P-value 7.02×10^{-15}) and 'DNA topology' (15 out of 19, P-value 2.38×10^{-30}). Our analysis predicts that the four unclassified genes are very likely to be involved in the above mentioned processes. On examination of the literature for 4 unclassified genes, we find that their involvement in DNA processing and DNA topology is likely due to their relation to helicase-proteins [16], [28]. These proteins play important roles in various cellular processes including DNA replication, DNA repair, RNA processing, chromosomal segregation, and maintenance of chromosome stability. It has been well known that the amino acid sequences of these proteins contain several conserved motifs, and that the open reading frames (ORFs) which encode helicase-related proteins make up several gene families [28]. While YHR218W encodes helicase-like protein within the telomeric Y' element, YHR219W encodes protein that is similar to helicases and contains telomeric short Y' element [16]. YBL112C and YLR464W also contain helicase-encoding repetitive sequence and lies within TEL02L (subtelomeric region next to the telomeric repeats) and TEL12R, respectively.

D. Evaluation Based on Independent Training and Test Sets

The performance of the proposed integration method relies critically on Yeast GO-Slim process annotations in order to determine the weights of the data sources in the training process and its evaluation depends on MIPS annotation in the test

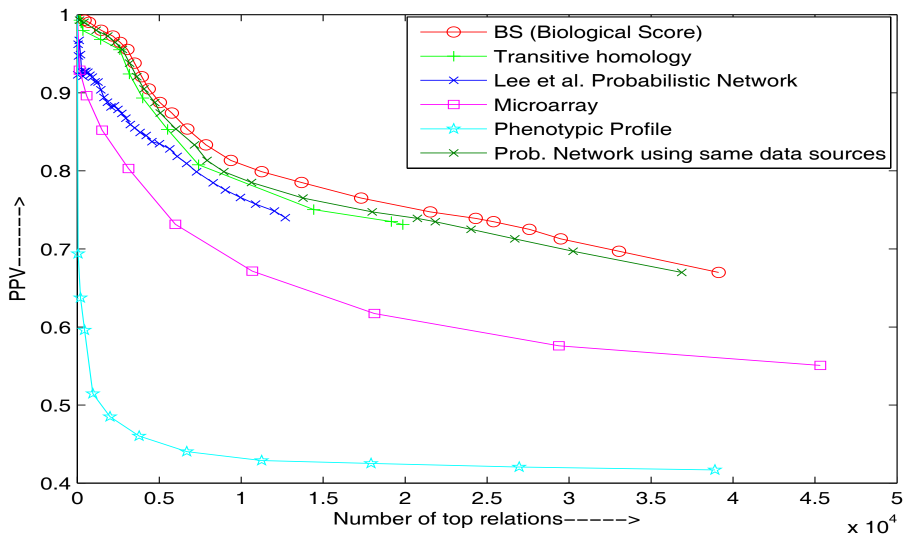


Fig. 5. Comparison between the $Biological\ Score$ (BS), Lee et al.'s Probabilistic Network, and individual data source in terms of PPV versus the number of top gene pairs. The available annotations (using vector $V(g)$ in Eq. 5) from Yeast GO-Slim process and MIPS are first merged for all the genes, and then the genes (with annotation vectors) are randomly splited into disjoint training and test sets. While, the training set is used to determine the weighting factors in BS and 'Probabilistic Network using same data sources', the test set is used to evaluate (using PPV) the gene pairs of all the methods and data sources.

process. But to perform a fair evaluation, the training and test set should be independent with null intersection. In this regard, we also experimented with an alternative method based on cross-validation. First, we merged the available annotations (using vector $V(g)$ in Eq. 5) from Yeast GO-Slim process and MIPS for all the genes, and then split the genes (with annotation vectors) into independent training and test sets. Because data are integrated using weights derived only from the training set, the performances measured on the remaining test benchmark are expected to be free from circular logic and memorization of the annotation set during the training procedure. Moreover, the KEGG pathway profile dataset is now excluded from the datasource integration procedure as pathway information is a bit redundant with functional annotations available in MIPS and Yeast GO-Slim process.

We randomly separated the set of 6,072 genes into 2 disjoint training and test subsets of 3,036 genes each. All links among genes within the same training subset are calculated and then used for training the weights. Similarly, all links among genes within the same test subset are calculated, with neither links nor genes shared between the training and test sets. The calculation of weights for data source integration and all other steps prior to the final assessment of *BS*, are performed using only the training set. The final assessment is performed on the independent test set. The cross-validation procedure is repeated 10 times and the performance of *BS* is evaluated without using KEGG pathway profile as a data source.

Fig. 5 shows the curves comparing *BS* and individual data sources in terms of PPV for top gene pairs, in one of the cross-validation procedures. Similar curves are obtained when the cross-validation procedure is repeated. The curves show that *BS* performs better than Lee et al.'s Probabilistic Network and individual data source. In clustering solutions, using *K-BS* and repeteting cross-validation procedure, on average 800 clusters are identified with functional enrichment in one or more categories by selecting K to be at most 10, BS cut-off value 0.77, and P-values less than 10^{-5}. From functionally enriched clusters, on average we predict the functions of 500 classified genes with 96.2 PPV and 300 unclassified genes by assigning the function related with the smallest P-value. In one of the cross-validation process (out of 10 repetitions), functions of 516 classified yeast genes are predicted with 97.1 PPV from 516 clusters. For the purpose of illustration, the predicted functions of 516 classified yeast genes are uploaded at http://www.isical.ac.in/~scc/Bioinformatics/AdS/classifiedpredictionreview.xls. Although, the KEGG pathway profile dataset is now excluded in the data source integration procedure using Eq. 7, the PPVs are higher (96.2) than that reported including KEGG pathway profile dataset (95.16) because, a part of Yeast GO-Slim process annotations, which on average have more genes in each functional process than pure MIPS annotations, are now included in the cluster evaluation procedure.

IV. Conclusion

In this study, we proposed a framework for data source integration that combines information from different sources to predict gene function. We employed functional annotation based weighting of data sources through annotations of classified genes to predict gene pairs for yeast from five data sources, namely, phenotypic profiles, gene expression data, KEGG profiles, protein-protein interaction and protein sequence similarity through transitive homologues. Functional categories of 60 unclassified

(MIPS October 2005) Yeast genes and 1855 classified genes are predicted with 95.16 PPV. Evaluation on the predicted gene pairs confirmed the validity and potential value of the proposed framework for gene function prediction.

Although a neighbor based clustering method needs a user defined neighbor number, from this investigation we find that *K-BS* is a highly accurate and efficient gene function annotation tool. The system integrates heterogeneous biological information in a functional annotation based weighting framework, leading to more biologically accurate gene groupings, which can be used for gene function prediction. The flexibility of the system allows for easy inclusion of other data sources by first benchmarking them, and then adaptively estimating the individual weights. Furthermore, we plan to examine our proposed framework on a larger test-bed by including similarities arising from gene-fusion and gene-order conservation based methods.

ACKNOWLEDGEMENT

We would like to acknowledge Dr. Martin Mnsterktter, CYGD coordinator at MIPS, for fixing the bugs in downloading functional categories, Dr. Maria C. Costanzo, Senior Scientific Curator of Saccharomyces Genome Database, for mapping the gene names from Yeast GO-Slim process annotations to ORFs and anonymous reviewers for their suggestions in improving the quality of research. Support of the Dept. of Science and Technology, Govt. of India to the Center for Soft Computing Research through its IRHPA scheme is acknowledged.

REFERENCES

[1] J. A. Brown, G. Sherlock, C. L. Myers, N. M. Burrows, C. Deng, H. I. Wu, K. E. McCann, O. G. Troyanskaya, and J. M. Brown, "Global analysis of gene function in yeast by quantitative phenotypic profiling," *Molecular System Biology*, vol. 2, no. 2006.0001, pp. 1–9, 2006.

[2] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein, "Cluster analysis and display of genome-wide expression patterns," *Proc. Natl. Acad. Sci. USA*, vol. 95, pp. 14863–14867, 1998.

[3] E. M. Marcotte, M. Pellegrini, M. J. Thompson, T. O. Yeates, and D. Eisenberg, "A combined algorithm for genome-wide prediction of protein function," *Nature*, vol. 402, pp. 83–86, 1999.

[4] M. Kanehisa, S. Goto, M. Hattori, K. F. Aoki-Kinoshita, M. Itoh, S. Kawashima, T. Katayama, M. Araki, and M. Hirakawa, "From genomics to chemical genomics: new developments in kegg," *Nucleic Acids Res.*, vol. 34, pp. D354–D357, 2006.

[5] L Salwinski, C. S. Miller, A. J. Smith, F. K. Pettit J. U. Bowie, and D. Eisenberg, "The database of interacting proteins," *Neuclic Acid Research*, vol. 32, pp. 449451, 2004.

[6] T. Reguly, A. Breitkreutz, L. Boucher, B. J. Breitkreutz, G. C. Hon, C. L. Myers, A. Parsons, H. Friesen, R. Oughtred, A. Tong, C. Stark, Y. Ho, D. Botstein, B. Andrews, C. Boone, O. G. Troyanskya, T. Ideker, K. Dolinski, N. N. Batada, and M. Tyers, "Comprehensive curation and analysis of global interaction networks in saccharomyces cerevisiae," *Journal of Biology*, vol. 5, no. 4, pp. 1–28, 2006.

[7] M. Pellegrini, E. M. Marcotte, M. J. Thompson, D. Eisenberg, and T. O. Yeates, "Assigning protein functions by comparative genome analysis: protein phylogenetic profiles," *Proc. Natl. Acad. Sci. USA*, vol. 96, pp. 4285–4288, 1999.

[8] E. M. Marcotte, M. Pellegrini, H. L. Ng, D. W. Rice, T. O. Yeates, and D. Eisenberg, "Detecting protein function and protein-protein interactions from genome sequences," *Science*, vol. 285, pp. 751–753, 1999.

[9] O. G. Troyanskaya, K. Dolinski, A. B. Owen, R. B. Altman, and D. Botstein, "A bayesian framework for combining heterogeneous data sources for gene function prediction (in saccharomyces cerevisiae)," *Proc. Natl. Acad. Sci. USA*, vol. 100, no. 14, pp. 8348–8353, 2003.

[10] C. V. Mering, R. Krause, B. Snel, M. Cornell, S. G. Oliver, S. Fields, and P. Bork, "Comparative assessment of large-scale data sets of protein-protein interactions," *Nature*, vol. 417, pp. 399–403, 2002.

[11] I. Lee, S. V. Date, A. T. Adai, and E. M. Marcotte, "A probabilistic functional network of yeast genes," *Science*, vol. 306, pp. 1555–1558, 2004.

[12] V. Spirin and L. A. Mirny, "Protein complexes and functional modules in molecular networks," *Proc. Natl Acad. Sci. USA*, vol. 100, no. 21, pp. 1212312128, 2003.

[13] DeLisi C Yanai I., "The society of genes: networks of functional links between genes from comparative genomics," *Genome Biology*, vol. 3, no. 11, pp. 1–64, 2002.

[14] Munich Information for Protein Sequences, "http://www.mips.com," .

[15] W. C. Barker et al., "The protein information resource (pir)," *Nucleic Acids Research*, vol. 28, no. 1, pp. 41–44, 2000.

[16] S. S. Dwight, M. A. Harris, K. Dolinski, C. A. Ball, G. Binkley, K. R. Christie, D. G. Fisk, L. Issel-Tarver, M. Schroeder, G. Sherlock, A. Sethuraman, S. Weng, D. Botstein, and J. M. Cherry, "Saccharomyces genome database (sgd) provides secondary gene annotation using the gene ontology (go)," *Nucleic Acids Research*, vol. 30, no. 1, pp. 69–72, 2002.

[17] T. H. B, B. Dysvik, and I. Jonassen, "Lsimpute: accurate estimation of missing values in microarray data with least squares methods," *Nucleic Acids Research*, vol. 32, no. 3: e34, pp. online, 2004.

[18] Website, "http://rana.lbl.gov/EisenData.htm," .

[19] G. Sherlock et al., "The stanford microarray database," *Nucleic Acids Research*, vol. 29, no. 1, pp. 152–155, 2001.

[20] S. F. Altschul, T. L. Madden, A. A. Schffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman, "Gapped blast and psi-blast: a new generation of protein database search programs," *Nucleic Acids Research*, vol. 25, no. 17, pp. 3389–3402, 1997.

[21] J. Park, K. Karplus, C. Barrett, R. Hughey, D. Haussler, T. Hubbard, and C. Chothia, "Sequence comparisons using multiple sequences detect three times as many remote homologues as pairwise methods," *J Mol Biol*, vol. 284, pp. 1201–1210, 1998.

[22] H. Xie, A. Wasserman, Z. Levine, A. Novik, V. Grebinskiy, Avi Shoshan, and Liat Mintz, "Large-scale protein annotation through gene ontology," *Genome Research*, vol. 12, pp. 785–794, 2002.

[23] Q. Ma, G. W. Chirn, R. Cai, J. D. Szustakowski, and N. Nirmala, "Clustering protein sequences with a novel metric transformed from sequence similarity scores and sequence alignments with neural networks," *BMC Bioinformatics*, vol. 6, no. 242, 2005.

[24] A. Bairoch, R. Apweiler, C. H. Wu, W. C. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, M. J. Martin, D. A. Natale, C. O'Donovan, N. Redaschi, and L. S. Yeh, "The universal protein resource (uniprot)," *Nucleic Acids Research*, vol. 33, pp. 154–159, 2005.

[25] P. Pipenbacher, A. Schliep, S. Schneckener, A. Schonhuth, D. Schomburg, and R. Schrader, "Proclust: improved clustering of protein sequences with an extended graph-based approach," *Bioinformatics*, vol. 18, no. 2, pp. S182S191, 2002.

[26] D. Wettschereck, D. W. Aha, and T. Mohori, "A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms," *Artificial Intelligence Review*, vol. 11, no. 1-5, pp. 273–314, 1997.

[27] I. Lee, R. Narayanaswamy, and E. M. Marcotte, *Yeast Gene Analysis*, chapter Bioinformatic prediction of yeast gene function, Elsevier Press, Amsterdam, 2006.

[28] A. Shiratori, T. Shibata, M. Arisawa, F. Hanaoka, Y. Murakami, and T. Eki, "Systematic identification, classification, and characterization of the open reading frames which encode novel helicase-related proteins in saccharomyces cerevisiae by gene disruption and northern analysis," *Yeast*, vol. 15, no. 3, pp. 219–253, 1999.

Bioinformatics in Neurocomputing Framework

Shubhra Sankar Ray, Sanghamitra Bandyopadhyay, Pabitra Mitra and Sankar K. Pal
Machine Intelligence Unit, Indian Statistical Institute, Kolkata 700108
Email: {shubhra_r,sanghami,pabitra_r,sankar}@isical.ac.in

CODEC-04, pp. 94, Januray 1-3, 2004, kolkata

Abstract - **Bioinformatics is an interdisciplinary research area of biology and computer science. This article provides an overview of neural network applications in different bioinformatics tasks. The relevance of intelligent systems and neural networks to these problems is first mentioned. Different tasks like gene sequence analysis, gene finding, protein structure prediction and analysis, microarray analysis and gene regulatory network analysis are described along with some classical approaches. Different neural network based algorithms to address the aforesaid tasks are then presented. Finally some directions for future research are provided.**

Keywords: **biological data mining, gene sequence analysis, protein structure, microarray, gene regulatory network, multiplayer perceptron, self organizing map**

I. INTRODUCTION

Over the past few decades, major advances in the field of molecular biology, coupled with advances in genomic technologies, have led to an explosive growth in the biological information generated by the scientific community. This deluge of genomic information has, in turn, led to an absolute requirement for computerized databases to store, organize and index the data, and for specialized tools to view and analyze the data.

Bioinformatics can be viewed as the *use of computational methods to make biological discoveries* [1]. It is an interdisciplinary field involving biology, computer science, mathematics and statistics to analyze biological sequence data, genome content & arrangement, and to predict the function and structure of macromolecules. The ultimate goal of the field is to enable the discovery of new biological insights as well as to create a global perspective from which unifying principles in biology can be derived. There are three important sub-disciplines within bioinformatics:

a) The development of new algorithms and models to assess different relationships among the members of a large biological data set;
b) The analysis and interpretation of various types of data including nucleotide and amino acid sequences, protein domains, and protein structures; and
c) The development and implementation of tools that enable efficient access and management of different types of information

This article provides a survey of the various neural network based techniques that have been developed over the past few years for different bioinformatics tasks. First we describe the primary bioinformatics tasks along with their biological basis. Next different neural network based algorithms available to address them are explained. Finally, some conclusions and future research directions are presented.

II. BIOINFORMATICS TASKS

The different biological problems studied within the scope of bioinformatics can be broadly classified into two categories: genomics and proteomics which include genes, proteins, and amino acids. We describe below different tasks involved in their analysis along with their utility.

A. Gene Sequence Analysis

The evolutionary basis of sequence alignment is based on the principles of *similarity* and *homology* [3]. Similarity is a quantitative measure of the fraction of two genes which are identical in terms of observable quantities. Homology is the conclusion drawn from data that two genes share a common evolutionary history; no metric is associated with this. The tasks of sequence analysis are-

Sequence Alignment: An alignment is a mutual arrangement of two or more sequences, which exhibits where the sequences are similar, and where they differ. An optimal alignment is one that exhibits the most correspondences and the least differences. It is the alignment with the highest score but may or may not be biologically meaningful. Basically there are two types of alignment methods, global alignment and local alignment. Global alignment [4] maximizes the number of matches between the sequences along the entire length of the sequence. Local alignment [5] gives a highest scoring to local match between two sequences.

Pattern Searching: Pattern searching is search for a nucleic pattern in a nucleic acid sequence, in a set of sequences or in a databank. (e.g., INFOBIOGEN) [6]. It is the potential for uncovering evolutionary relationships and patterns between different forms of life. With the aid of nucleotide and protein sequences, it should be possible to find the ancestral ties between different organisms. So far, experience indicates that closely related organisms have similar sequences and that more distantly related organisms have more dissimilar sequences. Proteins that show a significant sequence conservation indicating a clear evolutionary relationship are said to be from the same *protein family*. By studying *protein folds* (distinct protein building blocks) and families, scientists are able to reconstruct the evolutionary relationship between two species and to estimate the time of divergence between two organisms since they last shared a common ancestor.

Gene Finding: In general DNA strand consists of a large sequence of nucleotides, or bases. For example there are more than 3 billions bases in human DNA sequences. Not all portions of the DNA sequences are *coding* and coding zones indicate that they are a template for a protein. In the human genome only 3%-5% of the sequence are coding, i.e., they constitute the gene. Due to the size of the database, manual searching of genes, which code for proteins, is not practical. Therefore automatic identification of the genes from the large DNA sequences is an important problem in bioinformatics [7].

B. *Protein Analysis*

Proteins are polypeptides, formed within cells as a linear chain of amino acids [8]. Within and outside of cells, proteins serve a myriad of functions, including structural roles (cytoskeleton), as catalysts (enzymes), transporter to ferry ions and molecules across membranes, and hormones to name just a few. There are twenty different amino acids that make up essentially all proteins on earth. Different tasks involved in protein analysis are as follows:

Multiple Sequence Alignment: Multiple amino acid sequence alignment techniques [1] are usually performed to fit one of the following scopes:
(a) determination of the consensus sequence of several aligned sequences; (b) help in the prediction of the secondary and tertiary structures of new sequences; and (c) preliminary step in molecular evolution analysis using phylogenetic methods for constructing phylogenetic trees.

In order to characterize protein families, one needs to identify shared regions of homology in a multiple sequence alignment; (this happens generally when a sequence search revealed homologies in several sequences) .The clustering method can do alignments automatically but are subjected to some restrictions. Manual and eye validations are necessary in some difficult cases. The most practical and widely used method in multiple sequence alignment is the hierarchical extensions of pairwise alignment methods, where the principal is that multiple alignments is achieved by successive application of pairwise methods.

Protein Motif Search: Protein motif search [7,8] allows search for a personal protein pattern in a sequence (personal sequence or entry of Gene Bank). Proteins are derived from a limited number of basic building blocks (domains). Evolution has shuffled these modules giving rise to a diverse repertoire of protein sequences, as a result of it proteins can share a global or local relationship. Protein motif search is a technique for searching sequence databases to uncover common domains/motifs of biological significance that categorize a protein into a family.

Structural Genomics: Structural genomics is the prediction of 3-dimensional structure of a protein from the primary amino acid sequence [9]. This is one of the most challenging tasks in bioinformatics..

The four levels of protein structure (Figure 1) are (a) *Primary structure* is the sequence of amino acids that compose the protein, (b) different regions of the sequence form local *secondary structures*, such as alpha helices and beta strands, (c) *Tertiary structure* is formed by packing secondary structural elements into one or several compact globular units called domains, and (d) Final protein may contain several polypeptide chains arranged in a *quaternary structure*.

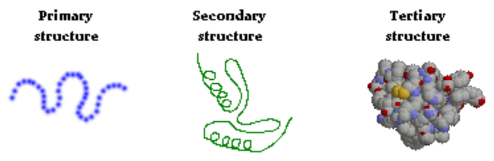


Figure 1: Different levels of protein structures

Sequence similarity methods predict secondary and tertiary structure based on homology to know proteins. Secondary structure predictions methods include Chou-Fasman [9], GOR, neural network, and nearest neighbor methods. Tertiary structure prediction methods include energy minimization, molecular dynamics, and stochastic searches of conformational space.

C. *Microarrays*

Microarray technology [10] makes use of the sequence resources created by the genome projects and other sequencing efforts to answer the question, what genes are expressed in a particular cell type of an organism, at a particular time, under particular conditions. Gene expression is the process by which a gene's coded information is converted into the structures present and operating in the cell. Expressed genes include those that are transcribed into mRNA and then translated into protein and those that are transcribed into RNA but not translated into protein (e.g., transfer and ribosomal RNAs). For instance, they allow comparison of gene expression between normal and diseased (e.g., cancerous) cells. There are several names for this technology - DNA microarrays, DNA arrays, DNA chips, gene chips, others.

Microarrays exploit the preferential binding of complementary single-stranded nucleic acid sequences. A microarray is typically a glass (or some other material) slide, on to which DNA molecules are attached at fixed locations (spots). There may be tens of thousands of spots on an array, each containing a huge number of identical DNA molecules (or fragments of identical molecules), of lengths from twenty to hundreds of nucleotides. (According to quick napkin calculations by Wilhelm Ansorge and Quackenbush in Heidelberg on 4 October, 2001, the number of DNA molecules in a microarray spot is 10^7-10^8). For gene expression studies, each of these molecules ideally should identify one gene or one exon in the genome, however, in practice this is not always so simple and may not even be generally possible due to families of similar genes in a genome. Microarrays that contain all of the about 6000 genes of the yeast genome have been available since 1997. The spots are either printed on the microarrays by a robot, or synthesized by photo-lithography (similarly as in computer chip productions) or by ink-jet printing.

There are different ways how microarrays can be used to measure the gene expression levels. One of the most popular micorarray applications allows to compare gene expression levels in two different samples, e.g., the same cell type in a healthy and diseased state. Conceptually, a gene expression database can be regarded as consisting of three parts – the gene expression data matrix, gene annotation and sample annotation.

Microarrays are already producing massive amounts of data. These data, like genome sequence data, can help us to gain insights into underlying biological processes only if they are carefully recorded and stored in databases, where they can be queried, compared and analyzed by different computer software programs. In many respects gene expression databases are inherently more complex than sequence databases (this does not mean that developing, maintaining and curating the sequence databases are any less challenging).

D. *Gene Regulatory Network Analysis*

Another important and interesting question in biology is how gene expression is switched on and off, i.e., how genes are regulated [1]. Since almost all cells in a particular organism have an identical genome, differences in gene expression and not the genome content are responsible for cell differentiation (how different cell types develop from a fertilized egg) during the life of the organism.

Gene regulation in eukaryotes, is not well understood, but there is evidence that an important role is played by a type of proteins called *transcription factors*. The transcription factors can attach (bind) to specific parts of the DNA, called transcription factor *binding sites* (i.e., specific, relatively short combinations of A, T, C or G), which are located in so-called *promoter* regions. Specific promoters are associated with particular genes and are generally not too far from the respective genes, though some regulatory effects can be located as far as 30,000 bases away, which makes the definition of the promoter difficult.

Transcription factors control gene expression by binding the gene's promoter and either activating (switching on) the gene's transcription, or repressing it (switching it off). Transcription factors are gene products themselves, and therefore in turn can be controlled by other transcription factors. Transcription factors can control many genes, and some (probably most) genes are controlled by combinations of transcription factors. Feedback loops are possible. Therefore we can talk about *gene regulation networks*. Understanding, describing and modelling such gene regulation networks are one of the most challenging problems in functional genomics. Microarrays and computational methods are playing a major role in attempts to reverse engineer gene networks from various observations. Note that in reality the gene regulation is likely to be a stochastic and not a deterministic process. Traditionally molecular biology has followed so-called reductionist approach mostly concentrating on a study of a single or very few genes in any particular research project. With genomes being sequenced, this is now changing into so-called systems approach.

III. ARTIFICIAL NEURAL NETWORK (ANN) ALGORITHMS IN BIOINFORMATICS

Neural network models try to emulate the biological neural network with electronic circuitry. NN models have been studied for many years with the hope of achieving human like performance, particularly in the field of pattern recognition. Recently, ANN have found a widespread use for classification tasks and function approximation in many fields of medicinal chemistry and bioinformatics. For these kinds of data analysis mainly two different types of networks are employed, "supervised" neural networks (SNN) and "unsupervised" neural networks (UNN). The main applications of SNN are function approximation, classification, pattern recognition and feature extraction, and prediction tasks. These networks require a set of molecular compounds with known activities to model structure-activity relationships. In an optimization procedure, these known "target activities" serve as a reference for SAR modeling. This principle coined the term "supervised" networks. Correspondingly, "unsupervised" networks can be applied to classification and feature extraction tasks even without prior knowledge of molecular activities or properties. The following sections describe different neural network applications to the bioinformatics tasks described previously.

A. Sequence Analysis

GenTHREADER is a neural network architecture that predicts similarity between gene sequences [11]. The effects of sequence alignment score and pairwise potential are the network outputs. Using GenTHREADER was successfully used in the following cases: ORF MG276 from Mycoplasma genitalium was predicted to share structure similarity with 1HGX;. MG276 shares a low sequence similarity (10% sequence identity) with 1HGX.

A back-propagation neural network can grossly approximate the score function of the popular BLAST family of genomic sequence alignment and scoring tools. The resultant neural network may provide a processing speed advantage over the BLAST tool, but may suffer somewhat in comparison to the accuracy of BLAST. Further study is necessary to determine whether a neural network with additional hidden units or structural complexity could be used to more closely approximate BLAST. However, closer approximation may also limit the speed performance advantages enjoyed by the neural network approach.

B Protein Analysis

The most successful techniques for prediction of the protein three-dimensional structure rely on aligning the sequence of a protein of unknown structure to a homologue of known structure. Such methods fail if there is no homologue in the structural database, or if the technique for searching the structural database is unable to identify homologues that are present.

Qian et al [12] used X-ray crystal structures of globular proteins available at that time to train a NN to predict the secondary structure of non-homologous proteins. Since every residue in a PDB entry can be associated to one of three secondary structures (HELIX, SHEET or neither: COIL) the authors were able to design a NN that had 21 input nodes (one for each residue including a null residue) and three output nodes coding for each of the three possible secondary structure assignments (HELIX, SHEET and COIL). It was easiest to restrict the input and output nodes to binary values (1 or 0) when loading the data onto the network during training. This explains why three output nodes are needed: HELIX was coded as 0,0,1 on the three output nodes; SHEET is coded as 0,1,0 and COIL is coded as 1,0,0. A similar binary coding scheme was used for the 20 input nodes for the 20 amino acids. Since the authors wished to consider a moving window of seven residues at a time, their input layer actually contained 20 x 7 nodes plus one node at each position for a null residue. Over 100 protein structures were used to train this network. After training, when the NN was queried with new data, a prediction accuracy of 64% was obtained.

Rost et al. [13] took advantage of the fact that a multiple sequence alignment contains more information about a protein than the primary sequence alone. Instead of using a single sequence as input into the network, they used a sequence profile that resulted from the multiple alignments. This resulted in a significant improvement in prediction accuracy to 71.4%. Recently, more radical changes to the design of NNs including bi-directional training and the use of the entire protein sequence as simultaneous input instead of a shifting window of fixed length has led to prediction accuracy above 71%.

The task of applying ANNs to the problem of protein structure prediction requires a certain number of input “nodes" and connect each one to every node in a hidden layer. Each node in the hidden layer is then connected to every node in the final output layer. The connection strength between each and every pair of nodes is initially assigned a random value and is then modified by the program itself during the training process. Each node will "decide" to send a signal to the nodes it is connected to based on evaluating its transfer function after all of its inputs and connection weights have been summed. Training proceeds by holding particular data (say from an entry in the Protein Data Bank) constant onto both the input and output nodes and iterating the network in a process that modifies the connection weights until the changes made to them approach zero. When such convergence is reached, the network is ready to receive new experimental data. Now the connection weights are not changed and the values of the hidden and output nodes are calculated. Selection of unbiased and normalized training data, however, is probably just as important as the network architecture in the design of a successful NN.

The prediction of protein secondary structure by use of carefully structured neural networks and multiple sequence alignments have been investigated by Riis and Krogh[14]. Separate networks are used for predicting the three secondary structures ff-helix, fi-strand and coil. The networks are designed using a priori knowledge of amino acid properties withrespect to the secondary structure and of the characteristic periodicity in ff-helices. Since these single-structure networks all have less than 600 adjustable weights over-fitting is avoided. To obtain a three-state prediction of ff-helix, fi-strand or coil, ensembles of single-structure networks are combined with another neural network. This method gives an overall prediction accuracy of 66.3% when using seven-fold cross-validation on a database of 126 non-homologous globular proteins. Applying the method to multiple sequence alignments of homologous proteins increases the prediction accuracy significantly to 71.3% [14].

C. Microarray

Clustering is commonly used in microarray experiments to identify groups of genes that share similar expression. Genes that are similarly expressed are often co-regulated and involved in the same cellular processes. Therefore, clustering suggests functional relationships between groups of genes. It may also help in identifying promoter sequence elements that are shared among genes. In addition, clustering can be used to analyze the effects of specific changes in experimental conditions and may reveal the full cellular responses triggered by those conditions.

Bayesian neural network is used with structural learning with forgetting for searching for optimal network size and structure for *microarray* data in order to capture the structural information of gene expressions [15,16]. The process of Bayesian learning starts with a Feed forward Neural Network (FFNN) and prior distribution for the network parameters. The prior distribution gives initial beliefs about the parameters before any data are observed. After new data are observed, the prior distribution is updated to the posterior distribution using Bayes rules. Multi-Layer Perceptron (MLP) is mainly considered as the network structure for Bayesian learning. Since the correlated data may include high levels of noise, efficient regularization techniques are required to improve the generalization performance. This involves network complexity adjustment and performance function modification. To do the latter, instead of the sum of squared error (SSE) on the training set, a cost function is automatically adjusted. PLANN (Plausible neural network) is another universal data analysis tool based upon artificial neural networks and is capable of plausible inference and incremental learning [17]. This tool has been applied to research data from molecular biological systems through the simultaneous analysis of gene expression data and other types of biological information.

D. Gene Regulatory Network

A novel clustering technique used for identifying gene regulatory networks is the adaptive double self-organizing map (ADSOM) [18]. It has a flexible topology and performs clustering and cluster visualization simultaneously, thereby requiring no a-priori knowledge about the number of clusters. ADSOM is developed based on a recently introduced technique known as double self-organizing map (DSOM). DSOM combines features of the popular self-organizing map (SOM) with two-dimensional position vectors, which serve as a visualization tool to decide how many clusters are needed. Although DSOM addresses the problem of identifying unknown number of clusters, its free parameters are difficult to control to guarantee correct results and convergence. ADSOM updates its free parameters during training and it allows convergence of its position vectors to a fairly consistent number of clusters provided that its initial number of nodes is greater than the expected number of clusters. The number of clusters can be identified by visually counting the clusters formed by the position vectors after training. The reliance of ADSOM in identifying the number of clusters is proven by applying it to publicly available gene expression data from multiple biological systems such as yeast, human, and mouse. ADSOM's performance in detecting number of clusters is compared with a model-based clustering method. It may be noted that gene regulatory network analysis is a very recent research area, and neural network applications to it are scarce.

IV. CONCLUSION AND SCOPE OF FUTURE RESEARCH

The rationale for applying computational approaches to facilitate the understanding of various biological processes includes:

- A more global perspective in experimental design; and
- The ability to capitalize on the emerging technology of database-mining - the process by which testable hypotheses are generated regarding the function or structure of a gene or protein of interest by identifying similar sequences in better characterized organisms.

Neural networks appear to be a very powerful artificial intelligence (AI) paradigm to handle these issues [19]. Other soft computing tools, like fuzzy set theory and genetic algorithms, integrated with ANN may also be based on the principles of Case Based Reasoning [20].

Even though the current approaches in biocomputing are very helpful in identifying patterns and functions of proteins and genes, they are still far from being perfect. They are not only time-consuming, requiring Unix workstations to run on, but might also lead to and assumptions due to necessary simplifications. It is therefore still mandatory to use biological reasoning and common sense in evaluating the results delivered by a biocomputing program. Also, for evaluation of the trustworthiness of the output of a program it is necessary to understand the mathematical / theoretical background of it to finally come up with a use- and senseful analysis.

REFERENCES

1. P. Baldi and S. Brunak, *Bioinformatics: The Machine Learning Approach*, MIT Press, Cambridge, MA, 1998.
2. R. B. Altman, A. Valencia, S. Miyano and S. Ranganathan, Challenges for intelligent systems in biology, *IEEE Intelligent Systems*, 16(6), pp. 14-20, 2001
3. H. Nash, D. Blair, J. Grefenstette, Proc. 2nd IEEE International Symposium on Bioinformatics and Bioengineering (BIBE'01), pp. 89, 2001, Bethesda, Maryland
4. S. B. Needleman and C. D. Wunsch,, *Journal of Molecular Biology*, 48, pp. 443-453, 1970.
5. T. F. Smith and M. S. Waterman, *Journal of Molecular Biology*, 147, pp. 195-197, 1981.
6. D. Gautheret, F. Major and R. Cedergren, Pattern searching/alignment with RNA primary and secondary structures: an effective descriptor for tRNA, *Comp. Appl. Biosc.* 6, pp. 325-331, 1990
7. J. W. Fickett, Finding genes by computer: The state of the art, Trends in Genetics, 12(8), pp. 316-320, 1996
8. Salzberg, S.L. , Searls, D.B., and Kasif, S. (Eds.) Computational Methods In Molecular Biology, North Holland: Elsevier Sciences, 1988
9. P. Chou, and G. Fasmann, "Prediction of the secondary structure of proteins from their amino acid sequence", Advances in Enzymology, 47, pp 45-148, 1978
10. J. Quackenbush, Computational analysis of microarray data, *National Review of Genetics*, 2, pp. 418-427, 2001
11. D. T. Jones, "GenTHREADER: An Efficient and Reliable Protein Fold Recognition.", Journal of Mol. Bio., 287, pp.797-815, 1999
12. N. Qian and T. J. Sejnowski, *Journal Molecular Biology,* 202**,** pp. 865-84, 1988
13. B. Rost and C. Sander, *Proc Natl Acad Sci U S A* 90, 7558-62, 1993
14. S.K. Riis and A. Krogh,"Improving Prediction of Protein Secondary Structure using Structured Neural Networks and Multiple Sequence Alignments", Journal of Computational Biology, 3, pp. 163-183, 1996
15. S. Liang, S. Fuhrman, and R. Somogyi. REVEAL: A general reverse engineering algorithm for inference of genetic network architectures, *In Pacific Symposium on Biocomputing,* 3, pp. 18-29, 1998.
16. Bayesian Neural Network for Microarray Data, Yulan Liang, E Olusegun Georgre, Arpad Kelemen Technical Report, Department of Mathematical Sciences , University of Memphis
17. PLANN Software, *PNN Technologies*, Pasadena, CA
18. H. Ressom, D. Wang, and P. Natarajan, Clustering gene expression data using adaptive double self-organizing map**,** Physiol. Genomics, 14, pp. 35–46, 2003
19. S. K. Pal, L. Polkowski and A. Skowron, Rough-neuro Computing: A way of computing with words, Springer, Berlin, 2003
20. S. K. Pal and S. C. K. Shiu, Foundations of Soft Case Based Reasoning, John Wiley, NY, 2004

New Operators of Genetic Algorithms for Traveling Salesman Problem

Shubhra Sankar Ray, Sanghamitra Bandyopadhyay and Sankar K. Pal
Machine Intelligence Unit
Indian Statistical Institute
Kolkata 700108
{shubhra_r, sanghami, sankar}@isical.ac.in

Abstract

This paper describes an application of genetic algorithm to the traveling salesman problem. New knowledge based multiple inversion operator and a neighborhood swapping operator are proposed. Experimental results on different benchmark data sets have been found to provide superior results as compared to some other existing methods.

Keywords: knowledge based multiple inversion, order crossover, knowledge based neighborhood swapping.

1. Introduction

The Traveling Salesman Problem (TSP) is one of the top ten problems, which has been addressed extensively by mathematicians and computer scientists. Its importance stems from the fact there is a plethora of fields in which it finds applications e.g., DNA fragment assembly, VLSI design. The classical formulation is stated as: Given a finite set of cities and the cost of traveling from city i to city j, if a traveling salesman were to visit each city exactly once and then return to the home city, which tour would incur the minimum cost? Formally, the TSP may be defined as follows [1]:

Let $\{1, 2, \ldots n\}$ be the labels of the n cities and $C = [c_{i,j}]$ be a $n \times n$ cost matrix where $c_{i,j}$ denotes the cost of traveling from city i to city j. The total cost A of a TSP tour is given by

$$A(n) = \sum_{i=1}^{n-1} C_{i,,i+1} + C_{n,1} \qquad (1)$$

The objective is to find a permutation of the n cities which has minimum cost. The TSP is a very well known NP-hard problem [2] and therefore any problem belonging to the NP-class can be formulated as a TSP problem.

Over decades, researchers have suggested a multitude of heuristic algorithms, including genetic algorithms (GAs) [3], for solving TSP [6]. In this article we propose some new operators, namely Knowledge Based Multiple Inversion (KBMI) and Knowledge Based Neighborhood Swapping (KBNS) along with a Modified Order Crossover for solving TSP. The experimental results obtained on TSP benchmarks have been found to be superior in terms of quality of solution when compared with other existing GAs [6].

2. Proposed GA for TSP

A new algorithm, called SWAP_GATSP, is described in this section for solving TSP using elitist GAs with new operators namely, Knowledge Based Multiple Inversion, Modified Order Crossover and Knowledge Based Swapping. The structure of the proposed SWAP_GATSP is presented below.

```
begin SWAP_GATSP
    Create initial population of tours randomly.
    while generation_count < k do
        /* k = max. no. of generations.*/
        begin
            KBMI
            Natural selection
            MOC
            KBNS
            Mutation
            Elitism
            Increment generation_count.
        end ;
        Output the best individual found.
end SWAP_TSP.
```

2.1. String representation and Cost function

In order to find the shortest tour for a given set of n cities using GAs, the path representation [6] is more natural for TSP and has been well studied. In this encoding, the string representation for a TSP tour is an array of n integers which is a permutation of {1, 2, …… n}. The objective is to find a string with minimum cost. In the following subsections the new genetic operators employed in the proposed GA are described.

2.2. Knowledge Based Multiple Inversion

In this process for each string the distance between every two consecutive cities is calculated from the cost matrix and the distances are sorted in descending order. A record is kept so that one can find which distance corresponds to which two cities.

Suppose for a string (1 2 3 4 5 6 7 8 9) the sorted distances are between cities

(1,2), (5,6), (3,4), (4,5), (9,1), (2,3), (8,9), (6,7) and (7,8).

Now the substring between the highest two distances (1,2) and (5,6) is inversed, resulting in the parent (P) and the child (C) as follows

P1 = (1 |2 3 4 5 |6 7 8 9)
and
C1 = (1 |5 4 3 2 |6 7 8 9)

This inversion procedure is repeated for pairs [(3,4) and (4,5)], [(9,1) and (2,3)], [(8,9) and (6,7)] and so on for string with higher no of cities with the condition that a inversion process will not take place if a substring for a pair overlaps any other substring of previous all the pair. Now the substring for the 2nd pair [(3,4) and (4,5)] overlaps the substring for the pair [(1,2) and (5,6)], so no inversion of string will take place for the pair [(3,4) and (4,5)] and the pair will be removed from the list. The resulting list then becomes

[(1,2) and (5,6)], [(9,1) and (2,3)] and [(8,9) and (6,7)]

The substring for the pair [(9,1) and (2,3)] now overlaps with the substring for the pair [(1,2) and (5,6)], again no inversion of string will take place. The resulting list is

[(1,2) and (5,6)] and [(8,9) and (6,7)]

Now for the pairs [(1,2) and (5,6)] and [(8,9) and (6,7)] there is no overlap between substrings. Therefore the modified child C2 obtained from C1 are as follows

C1 = (1 |5 4 3 2 |6 |7 8| 9)
and
C2 = (1 5 4 3 2 6 |8 7| 9)

Regarding the number of pairs (say, pa) to be taken for the 1st iteration and for number of cities within 100, we have found experimentally that pa's are 3, 3, 4, 5 and 7 for number of cities 24, 29, 48, 70 and 100 respectively. These values can be achieved by an equation of the form

$$pa=[(n+32)/20]$$

where n is the number of cities.

This is not kept constant over the generations, rather it is varied in cycles of appropriate intervals linearly from

1) [pa] to 0.0 for iteration 1 to [z/3]
2) 0.0 to [pa] for iteration [z/3] to [2×z/3]
3) [pa] to 0.0 for iteration [2×z/3] to [z]

where z is the total no. of iterations performed in one run.

This kind of variation helps in exploring the search space efficiently and prevents the GA from getting stuck in the local optima. Note that this is an upgraded version of Simple Inversion Mutation [6], which is discussed later. But, it can't be called mutation operator, as it is a decisive process not a random one.

2.3. Natural Selection

This operator is designed by a common method of natural selection in GA called the Roulette Wheel method [3]. The Roulette Wheel method simply chooses the strings in a statistical fashion based solely upon their relative (i.e., percentage) cost or fitness values. So, the natural selection operator in this GA randomly chooses strings from the current population with probability inversely proportional to their cost.

2.4. Crossover

Before presenting our new crossover strategy, a closely related existing method known as Order based crossover is described briefly. It has been observed to be one of the bests in terms of quality and speed, and yet is simple to implement.

Order Based Crossover (OBC). The order based crossover operator [7] selects at random several positions in one of the parent tours, and the order of the cities in the selected positions of this parent is imposed on the other parent to produce one child. The other child is generated in an analogous manner for the other parent.

Modified Order Crossover. A randomly chosen crossover point divides the parent strings in left and right substrings. The right substrings of the parents s1 and s2 are selected. After selection of cities the process is the same as the order crossover. Only difference is that instead of selecting random several positions in a parent tour all the positions to the right of the randomly chosen crossover point are selected.

For example with the following parents and crossover point

s1 = (1 2 3 4 |6 9 8 5 7)
and
s2 = (2 1 9 8 |5 6 3 7 4),

after position selection

s1 = (1 2 * * * 9 8 * *)
and
s2 = (2 1 * * * * 3 * 4)

we obtain the generated pair of children as

b1 = (1 2 5 6 3 9 8 7 4)
and
b2 = (2 1 6 9 8 5 3 7 4)

Clearly this method allows only the generation of valid strings.

2.5. Knowledge Based Neighborhood Swapping

This is a novel deterministic operation injected in the typical structure of elitist GA. This works on the set of all strings obtained from crossover. For a string s in the population a random index value i is generated ($1<i<S$, where S is string length). Now for city (i-1) and city (i+1) find the city that is nearest to both the cities. Let its index be j. In this search the cities (i-1) and (i+1) are excluded. Then in s, swap the city at ith position with that at j^{th} position. This operation is repeated for all the strings in the population.

Index j is obtained as follows:
1) calculate the distance from the cost matrix for a particular city (say c) as
$X(c)= \text{distance}((i-1), c) + \text{distance}((i+1), c)$
2) repeat step 1 for all the cities except city(i-1) and city(i+1)
3) find the city c for which X(c) is minimum and make j=c
Since it is a deterministic operator based on the cost matrix, it helps the stochastic environment of the working of GA to derive an extra boost in the positive direction. As this operator is applied only once on every string for a random city index, not on all the cities in that string, this operation is not expensive.

2.6. Mutation

For the TSP the simple inversion mutation (SIM) and insertion mutation (ISM) are the leading performers [6].
Here simple inversion mutation (SIM) is performed on each string as follows:
This operator selects randomly two cut points in the string, and it reverses the substring between these two cut points. For example consider the tour
(1 2 3 4 5 6 7 8)
and suppose that the first cut point is chosen randomly between 2nd city and 3rd city, and the second cut point between the 5th city and the 6th city. Then the resulting strings will be
P = (1 2 |3 4 5| 6 7 8)
C = (1 2 |5 4 3| 6 7 8)

The mutation probability it is not kept constant over the generations. Rather it is varied in cycles of appropriate intervals [1] (linearly from 0.06 to 0.003 where n is the number of cities).

3. Time complexity of proposed GA

The time complexity of the algorithm SWAP_GATSP is given by $O(k \bullet N \bullet n)$ where k is the number of generations, N is the population size and n is the data size or the number of cities.

4. Experimental Results

SWAP_GATSP was implemented in Matlab 5.1 on Pentium-4 (1.7 GHz) and the results were compared with those obtained from the survey of Larranaga [6] and [1]. Results are also compared with a public domain TSP solver based on GA by Michael Lalena [5] having the proclamation of being the fastest among known solvers.

Table 1 summarizes the final results obtained by running the Multiple Inversion GA on several symmetric TSP instances containing 24, 29, 48, 70 and 100 cities, taken from the TSPLIB [4]. The best results from 30 run are listed here. The number of populations is taken 10 for Grtschels24.tsp and bayg29.tsp. The population is 24 for 48 cities, 30 for 70 cities and 40 for 100 cities. Crossover probability was fixed at 0.85 across the generations. As discussed in Section 2.6, the mutation probability was varied linearly in with iteration, maximum being 0.06 and minimum 0.003. These values are experimentally obtained which gives very good results.

For Grtschels24.tsp the previous best results for other GAs were 1272 km [6]. For Grtschels48.tsp the previous best result of 5074 km was with ER crossover and SIM mutation [6]. These investigations were carried out with population size of 200, mutation probability 0.01 and 50000 iterations [6]. The proposed approach exceeds the previous best result as shown in Table 1 with less no. of population size and iteration. For st70.tsp our result is compared with GA of Lalena with best result of 895 km. As Lalenas software is not downloadable at this instant for some difficulties in his web site, we compared the result with that in [1] where Lalenas GA was downloaded at that time. In [1] it was stated that the best result for their algorithm is 776 km for st70.tsp with population size of 50 and 5000 iterations. GA with only order crossover and simple inversion mutation (OX-SIM) is implemented next as standard GA for TSP [6, 1] for all the problems and the results are compared with SWAP_GATSP.

Table 1

		Best results for different TSPs				
Problem	Optimal	Proposed	[5]	[1]	Best GA in [6]	OX-SIM
Grtschels 24	1272	1272 (500 iter)	----	----	1272	1272 (8,000 iter.)
bayg 29	1610	1610 (600 iter)	----	----	----	1620 (10,000 iter.)
Grtschels 48	5046	5046 (800 iter.)	----	----	5074	5097 (12,000 iter.)
St70	675	685 (2000 iter.)	895	776	----	888 (15,000 iter.)
KroA 100	21282	21504 (5000 iter.)	----	----	----	22,400 (25,000 iter.)

Table 2

Problem	Proposed GA	Best GA in [6]	OX-SIM
Grtschels24	1272	1274	1342
bayg29	1615	-----	1720
Grtschels48	5110	5154	5451
St70	710	-----	920
KroA100	21,900	-----	23,200

Table 2 shows the average results after 5000 iterations. Here OX-SIM is implemented, but the average results for best previous GA are taken from Larranaga [6].

The SWAP_GATSP and the GA with OX-SIM have been compared w.r.t. computation time. Both programs were run for 358 seconds for Grtschels48.tsp and the fitness of fittest string is plotted with iteration as shown in Fig 1. In 358 seconds the SWAP_GATSP has gone through 3600 iterations and the GA with OX-SIM run for 5000 iterations. The lower graph shows that the optimal cost of 5046 km is achieved within 800 iterations for the SWAP_GATSP whereas cost is 6773 km for GA with OX-SIM (shown in upper graph). Similar results are also found when the proposed method is compared with other GAs stated in [6].

5. Conclusion

The results obtained with the newly designed genetic operators in our algorithm are impressive, on practical data set. Larger benchmarks are to be tested next. This method can be easily adapted to solving the asymmetric TSP. Experiments on comparing those results with other existing solvers for asymmetric TSP also need to be performed. Application of the developed SWAP_GATSP to real life problems like DNA fragment assembly, an important issue in bioinformatics, should be studied. The authors are currently working in this direction.

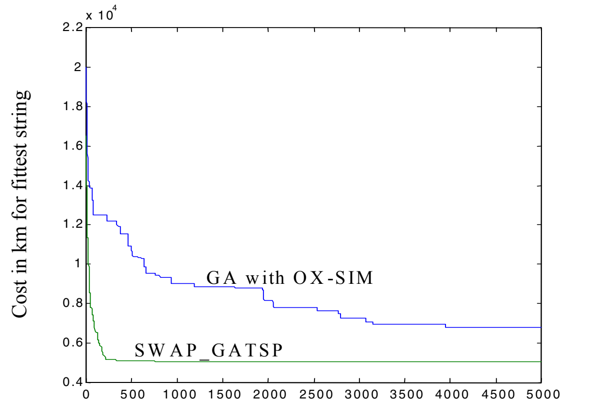


Figure 1:Cost of fittest string Vs. Iteration for Grtschels48.tsp

References

[1] Sur-Kolay S., Banerjee S., and Murthy C. A., "*Flavours of Traveling Salesman Problem in VLSI Design*", 1st Indian International Conference on Artificial Intelligence, *2003.*

[2] Garey, M. R., and Johnson, D. S.: *"Computers and Intractability: A Guide to the Theory of NP-completeness*", W. H. Freeman and Co., San Francisco, 1979.

[3] Goldberg, D. E.: "*Genetic Algorithm in Search, Optimization and Machine Learning",* Machine Learning. Addison-Wesley, New York, 1989.

[4] TSPLIB Homepage: http://www.iwr.uniheidelberg.de/groups/comopt/software/TSPLIB95/

[5] Lalena, M.: TSP solver. http://www.lalena.com/ai/tsp

[6] Larranaga, P., Kuijpers, C. M. H.,Murga, R. H., Inza, I., Dizdarevic, S.: "*Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators",* Artificial Intelligence Review. 13, 1999, 129-170.

[7] Syswerda, G, "*Schedule optimization using genetic algorithms, Handbook of Genetic Algorithms*", Van Nostrand Reinhold, New York, 1991, 332-349.

New Genetic Operators for Solving TSP: Application to Microarray Gene Ordering

Shubhra Sankar Ray, Sanghamitra Bandyopadhyay, and Sankar K. Pal

Machine Intelligence Unit, Indian Statistical Institute, Kolkata 700108
{shubhra_r, sanghami, sankar}@isical.ac.in
http://www.isical.ac.in/∼shubhra_r

Abstract. This paper deals with some new operators of genetic algorithms for solving the traveling salesman problem (TSP). These include a new operator called, "nearest fragment operator" based on the concept of nearest neighbor heuristic, and a modified version of order crossover operator. Superiority of these operators has been established on different benchmark data sets for symmetric TSP. Finally, the application of TSP with these operators to gene ordering from microarray data has been demonstrated.

1 Introduction

The Traveling Salesman Problem (TSP) has been used as one of the most important test-beds for new combinatorial optimization methods [1]. Its importance stems from the fact there is a plethora of fields in which it finds applications e.g., scheduling, vehicle routing, VLSI layout, microarray gene ordering and DNA fragment assembly. Over decades, researchers have suggested a multitude of heuristic algorithms, including genetic algorithms (GAs) [1, 2, 3]for solving TSP. The classical formulation of TSP is stated as: Given a finite set of cities and the cost of traveling from city i to city j, if a traveling salesman was to visit each city exactly once and then return to the home city, which tour would incur the minimum cost?

Let $1, 2, \cdots, n$ be the labels of the n cities and $C = [c_{i,j}]$ be an $n \times n$ cost matrix where $c_{i,j}$ denotes the cost of traveling from city i to city j. The Traveling Salesman Problem (TSP) is the problem of finding the shortest closed route among n cities, having as input the complete distance matrix among all cities. The total cost A of a TSP tour is given by

$$A(n) = \sum_{i=1}^{n-1} C_{i,i+1} + C_{n,1} \tag{1}$$

The objective is to find a permutation of the n cities, which has minimum cost.

The TSP, with some minor modifications, can be used to model the microarray gene ordering (MGO) problem. An optimal gene order provides a sequence of genes where the genes those are functionally related and similar are nearer in the ordering [4]. This functional relationship among genes is determined by gene expression levels from microarray by performing biological experiments [4]. Similarity between genes can be measured with Euclidean distance, Pearson correlation, absolute correlation, Spearman rank correlation, etc.

Two new genetic operators are proposed in this article for solving TSP. Consequently, the application of these operators are demonstrated for solving microarray gene ordering (MGO) problem efficiently.

2 Relevance of TSP in Microarray Gene Ordering

An optimal gene order, a minimum sum of distances between pairs of adjacent genes in a linear ordering $1, 2, \cdots, n$, can be formulated as [5]

$$F(n) = \sum_{i=1}^{n-1} C_{i,i+1}, \tag{2}$$

where n is the number of genes and $C_{i,i+1}$ is the distance between two genes i and $i+1$. In this study, the Euclidean distance is used to specify the distance $C_{i,i+1}$.

Let $X = x_1, x_2, \cdots, x_k$ and $Y = y_1, y_2, \cdots, y_k$ be the expression levels of the two genes in terms of log-transformed microarray gene expression data obtained over a series of k experiments. The Euclidean distance between X and Y is

$$C_{x,y} = \sqrt{\{x_1 - y_1\}^2 + \{x_2 - y_2\}^2 + \cdots + \{x_k - y_k\}^2}. \tag{3}$$

One can thus construct a matrix of inter-gene distances, which serves as a knowledge-base for mining gene order using GA. Using this matrix one can calculate the total distance between adjacent genes and find that permutation of genes for which the total distance is minimized. This is analogous to the traveling salesman problem.

3 GA with New Operators for TSP

In this section, two new operators of GAs for solving TSP are described. These are nearest fragment (NF) and modified order crossover (MOC). The genetic algorithm designed using these operators is referred to as FRAG_GA. The structure of the proposed FRAG_GA is provided in Fig. 1. Here path representation [1], linear normalized selection and elitism operators are utilized [2]. For TSP, simple inversion mutation (SIM) [1] is employed.

3.1 Nearest Fragment (NF) Heuristic

The nearest-neighbor (NN) heuristic for creating initial population, have the advantage that they only contain a few severe mistakes, while there are long segments connecting nodes with short edges. Therefore such tours can serve as good starting tours. In NN the main disadvantage is that, several cities are

```
begin FRAG_GA
    Create initial population with Nearest-Neighbor Heuristic
    while generation_count < $k$ do
        /* $k$ = max. number of generations. */
        begin
            Apply NF heuristic
            Linear normalized selection
            MOC
            Mutation
            Elitism
            Increment generation_count
        end
        Output the best individual found
end FRAG_GA
```

Fig. 1. The Pseudo-code for FRAG_GA

not considered during the course of the algorithm and have to be inserted at high costs in the end. This leads to severe mistakes in path construction. To overcome the disadvantages of the NN heuristics, we propose a new heuristic operator, called the Nearest Fragment (NF) operator, which is used in every generation (iteration) of GA with a predefined probability for every chromosome in the population as a subsequent tour improvement method. In this process, each string (chromosome in GA) is randomly sliced in *frag* fragments. The value of *frag* is chosen in terms of the total number of cities (n) for a particular TSP instance. For tour construction the first fragment is chosen randomly. From the last city of that fragment, the nearest city that is either a start or an end point of a not yet visited tour fragment is determined from the cost matrix. The fragment containing the nearest city is connected to the selected fragment, with or without inversion depending on whether the nearest city is the last city of a fragment or not respectively. The process is repeated until all fragments have been reconnected.

3.2 Modified Order Crossover (MOC)

Order crossover [6] has been observed to be one of the best in terms of quality and speed, and yet is simple to implement for solving TSP using GA [1, 2, 3]. In order crossover the length of a substring is chosen randomly. Thus on an average, the length is equal to $n/2$. This can lead to a marked increase in the computational time, which can be reduced if the length of the substring for performing crossover can be fixed to a small value. However, no study has been reported in the literature for determining an appropriate value of the length of a substring for performing order crossover. Such an attempt is made in this article where it is found that a substring length $'y'$ for MOC provides good results for TSP if $y = \max\{2, \alpha\}$, where $n/9 \leq \alpha \leq n/7$ (n is the total number of cities). Unlike order crossover, where the substring length is randomly chosen, in MOC it is predefined at y. For example, for a 10 city problem the value of α

is predefined at 1.25, therefore $y = 2$. The rest of the process in MOC is same as order crossover.

4 Experimental Results

FRAG_GA is implemented in Matlab 5.2 on Pentium-4 (1.7 GHz). The experiment has two parts. In the first part we have compared FRAG_GA with SWAP_GATSP [3], and OX_SIM (standard GA with order crossover and simple inversion mutation) [1] for solving benchmark TSP instances like Grtschels24, kroA100, d198, ts225, pcb442 and rat783 [7]. In the second part for biological microarray gene ordering, Cell Cycle cdc15, Cell Cycle and Yeast Complexes datasets are chosen [8]. The three data sets consists of about 782, 803 and 979 genes respectively, which are cell cycle regulated in Saccharomyces cerevisiae, with different number of experiments (24, 59 and 79 respectively) [4]. Each dataset is classified into five groups termed G1, S, S/G2, G2/M, and M/G1 by Spellman et. al. [4]. Throughout the experiments the population size is taken to be 10 for smaller problems (<100 cities/genes), while for larger problems (≥100 cities/genes) this is set equal to 20. Crossover probability is fixed at 0.85 and mutation probability is fixed at 0.015 across the generations.

For the nearest fragment (NF) operator, each string (chromosome in GA) is randomly sliced in *frag* fragments, where $frag \cong n/8$. Probability of NF operator was set to be 0.4 for n greater than 100 and 0.5, otherwise. The value of substring length for the modified order crossover operator (MOC) is kept in the range $n/7$ to $n/9$. All these values were obtained after extensive experiments, which are omitted here for the lack of space.

Table 1 summarizes the best results and average results obtained by running the FRAG_GA, SWAP_GATSP and OX_SIM on the aforesaid TSP instances.

Table 1. Cost values using FRAG_GA, SWAP_GATSP and OX_SIM for different TSP instances

		Best Results			Average Results		
Problem	Optimal	FRAG_GA	SWAP_ GATSP	OX_SIM	FRAG_GA	SWAP_ GATSP	OX_SIM
Grtschels 24	1272	1272 (130)	1272 (500)	1272 (8,000)	1272 (1000)	1272 (2000)	1322 (15000)
KroA 100	21282	21282 (800)	21504 (5000)	22,400 (25,000)	21,350 (2000)	21,900 (5000)	22670 (30000)
d198	15780	15834 (3000)	15992 (7000)	16,720 (25,000)	15964 (3500)	16,132 (10000)	18200 (40000)
Ts 225	126643	126730 (3000)	127012 (7000)	135800 (25,000)	126890 (3500)	128532 (10000)	138283 (40000)
Pcb 442	50778	51104 (8000)	52620 (15000)	53402 (40,000)	51930 (10000)	53,820 (20000)	59740 (65000)
Rat 783	8806	9007 (15000)	9732 (30000)	10810 (70,000)	9442 (20000)	10110 (40000)	11520 (100000)

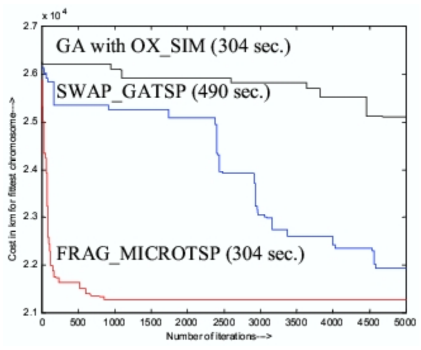


Fig. 2. Variation of cost of the best string with number of iteration for kroa100.tsp

For each problem the iteration in which the result is obtained is mentioned in columns 3-8 within parentheses. In SWAP_GATSP and OX_SIM the number of populations is taken to be 10 for 24 and 29 cities, 24 for 48 and 51 cities, 30 for 70 and 76 cities, and 40 for number of cities greater than or equal to 100 [3]. As can be seen from Table 1 FRAG_GA is superior in terms of quality of solution when compared with other existing GAs [1,3].

Fig. 2 shows a comparison of FRAG_GA, SWAP_GATSP and OX_SIM when the fitness value of the fittest string is plotted with iteration. The three programs were run for 5000 iterations for kroa100.tsp with population 20. At any iteration, the FRAG_GA has the lowest tour cost. It took 304 seconds, 490 seconds and 304 seconds by FRAG_GA, SWAP_GATSP and OX_SIM respectively for executing 5000 iterations. Moreover, only FRAG_GA is seen to converge at around 800 iterations at the optimal cost value of 21,282 km. On the other hand, the cost is 21912 km for SWAP_GATSP and 25103 km for OX_SIM even after 5000 iterations.

The performance of FRAG_GA on microarray datasets is evaluated with a biological score (not used as fitness function of GA), defined by [5]

$S(n) = \sum_{i=1}^{n-1} C_{i,,i+1}$ where $C_{i,,i+1} = 1$, if gene i and $i+1$ are in the same group

$= 0$, if gene i and $i+1$ are not in the same group.

Using this, a solution of gene ordering has a higher score when more genes within the same group are aligned next to each other. Table 2 compares the performance of our FRAG_GA with other GA based methods in terms of S value. It is clear that FRAG_GA and NNGA [9] are comparable and they both

Table 2. Comparison of FRAG_GA with other algorithms in terms of best score

Algorithms	Cell cycle cdc15	Cell cycle	Yeast complexes
FRAG_GA	537	635	384
NNGA	539	634	384
FCGA	521	627	—-

dominate FCGA [5]. Note that FRAG_GA is a conventional GA, while NNGA (hybrid GA) uses exhaustive local search methods [10], which provides the key contribution to optimality (not the GA itself). The main reason behind the good results obtained by FRAG_GA is that, biological solutions of microarray gene ordering lie in more than one sub optimal point (in terms of gene expression distance) rather than one optimal point.

5 Conclusion

A new "nearest fragment operator" (NF) and "modified version of order crossover operator" (MOC) of GAs are described along with their implementation for solving both symmetric TSP and microarray gene ordering problem. Appropriate number of fragments and appropriate substring length in terms of the number of cities are determined for NF and MOC respectively, and then applied on TSP and microarray data. It appears that NF operator is able to augment the search space quickly and thus obtains much better results compared to other heuristics. Moreover, MOC requires shorter computation time; thereby balancing the overhead corresponding to the NF operator.

Acknowledgement

This work is supported by the grant no. 22(0346)/02/$EMR-II$ of the Council of Scientific and Industrial Research (CSIR), New Delhi.

References

1. Larranaga, P., Kuijpers, C., Murga, R., Inza, I., Dizdarevic, S.: Genetic algorithms for the traveling salesman problem: A review of representations and operators. Artificial Intell. Rev. **13** (1999) 129–170
2. Goldberg, D.E.: Genetic Algorithm in Search, Optimization and Machine Learning. Machine Learning, Addison-Wesley, New York (1989)
3. Ray, S.S., Bandyopadhyay, S., Pal, S.K.: New operators of genetic algorithms for traveling salesman problem. Volume 2., Cambridge, UK, ICPR-04 (2004) 497–500
4. Spellman, P.T., Sherlock, G., Zhang, M.Q., Iyer, V.R., Anders, K., Eisen, M.B., Brown, P.O., Botstein, D., Futcher, B.: Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisia by microarray hybridization. Molecular Biology Cell **9** (1998) 3273–3297
5. Tsai, H.K., Yang, J.M., Kao, C.Y.: Applying Genetic Algorithms To Finding The Optimal Gene Order In Displaying The Microarray Data. GECCO (2002) 610–617
6. Davis, L.: Applying adapting algorithms to epistatic domains. Proc. Int. Joint Conf. Artificial Intelligence (Quebec, canada, 1985)
7. TSPLIB: (http://www.iwr.uniheidelberg.de/groups/comopt/software/TSPLIB95/)
8. (http://www.psrg.lcs.mit.edu/clustering/ismb01/optimal.html)
9. Lee, S.K., Kim, Y.H., Moon, B.R.: Finding the Optimal Gene Order in Displaying Microarray Data. GECCO (2003) 2215–2226
10. Lin, S., Kernighan, B.W.: An effective heuristic for the traveling salesman problem. Operation Research **21** (1973) 498–516

Gene Ordering in Partitive Clustering using Microarray Expressions

Shubhra Sankar Ray[1], Sanghamitra Bandyopadhyay[2], and Sankar K. Pal[1]
[1]Center for Soft Computing Research: A National Facility, [2]Machine Intelligence Unit,
Indian Statistical Institute,
Email: {shubhra_r, sanghami, sankar}@isical.ac.in

Abstract

Motivation: A central step in the analysis of gene expression data is the identification of groups of genes that exhibit similar expression patterns. Clustering and ordering the genes using gene expression data into homogeneous groups was shown to be useful in functional annotation, tissue classification, regulatory motif identification, and other applications. Although there is a rich literature on gene ordering in hierarchical clustering framework for gene expression analysis, to the best knowledge of the author, there is no work addressing and evaluating the importance of gene ordering in partitive clustering framework. Outside the framework of hierarchical clustering, different gene ordering algorithms are applied on the whole data set, and the domain of partitive clustering is still unexplored with gene ordering approaches.

Results: A new hybrid method for ordering genes in each of the clusters of partitive clustering solution using microarray gene expressions is proposed. Two existing ordering algorithms used for optimally ordering cities in Travelling Salesman Problem (namely, Concorde [1] and FRAG_GA [2]), are hybridized individually with Self Organizing MAP to show the importance of gene ordering in partitive clustering framework. We validated our hybrid approach using Yeast and Fibroblast data and showed that our approach improves the result quality of partitive clustering solution, by identifying subclusters within big clusters, minimization of summation of gene expression distances and the maximization of biological gene ordering using MIPS categorization. Moreover, the new hybrid approach, finds comparable or sometimes superior biological gene order than those obtained by optimal leaf ordering in hierarchical clustering solution [3].

Reference:

[1] D. Applegate, R. Bixby, V. Chvátal, and William Cook, "Concorde Package. [Online]," ww.tsp.gatech.edu/concorde/downloads/codes/src/ co031219.tgz, 2003.

[2] Shubhra Sankar Ray, Sanghamitra Bandyopadhyay and Sankar K. Pal, "Genetic Operators for Combinatorial Optimization in TSP and Microarray Gene Ordering", *Applied Intelligence* (published online: epub ahead of print), 2006

[3] Z. Bar-Joseph, D. K. Gifford, and T. S. Jaakkola, "Fast optimal leaf ordering for hierarchical clustering," *Bioinformatics*, vol.-17, pp- 22-29, 2001.

New Distance Measure for Microarray Gene Expressions using Linear Dynamic Range of Photo Multiplier Tube

Shubhra Sankar Ray
Center for
Soft Computing Research
Indian Statistical Institute
Kolkata, India
shubhra_r@isical.ac.in

Sanghamitra Bandyopadhyay
Machine Intelligence Unit
Indian Statistical Institute
Kolkata, India
sanghami@isical.ac.in

Sankar K. Pal
Center for
Soft Computing Research
Indian Statistical Institute
Kolkata, India
sankar@isical.ac.in

Abstract

This paper deals with a new distance measure for genes using their microarray expressions. The distance measure is called, "Maxrange *distance", where an experiment specific normalization factor is incorporated in the computation of the distance. The normalization factor is dependent on the linear dynamic range of the photo multiplier tube (PMT) for scanning fluorescence intensities of the gene expression values. Superiority of this distance measure in the microarray gene ordering problem has been extensively established on widely studied microarray data sets by performing statistical tests.*

1 Introduction

The recent advances in DNA array technologies have resulted in a significant increase in the amount of genomic data [3, 2]. The most powerful and commonly used technique is that involving microarray, which has enabled the monitoring of the expression levels of more than thousands of genes simultaneously. Due to the large quantity of information available from microarray it is necessary to find an appropriate distance measure for genes and to employ a process of classification of the data in order to obtain initial conclusions about the genes.

The present article deals with the tasks of measuring the distance between genes and evaluating their biological ordering in clustering framework. The widely used measures for finding the global similarity (where all the gene expression values present in the gene are taken into consideration) between genes are the Pearson correlation [3, 2] and the Euclidean distance [8]. In computing the similarity, all the above mentioned measures do not assign appropriate weights to gene expressions obtained from different types of experiments, where the expressions differ by orders of magnitude from one type to another. Consequently, gene expression values in lower dynamic range do get dominated by those with higher dynamic range. A new similarity measure between genes, called "*Maxrange* distance" is defined in this article, where gene expression (for a particular type of experiment) distance between two genes are first normalized with a factor dependent on the linear dynamic range of photo multiplier tube (used for scanning fluorescence intensities of that experiment), and then summed to find a global distance.

Superiority of the proposed *Maxrange* distance measure over the related measures is established by using them on four different algorithms.

2 Gene Ordering Methods

Cluster analysis, ordering, and display of gene expression patterns are considered to be useful tools to detect genes that are co-expressed or implicated in similar cellular functions [3, 2]. Hierarchical clustering approaches (single, complete and average linkage) [3, 1] group gene expressions into trees of clusters. They start with singleton sets and merge all genes until all nodes belong to only one set. Hierarchical clustering does not determine unique clusters. Thus the user has to determine which of the subtrees are clusters and which subtrees are only a part of a bigger cluster. So in the framework of hierarchical clustering a gene ordering algorithm helps the user to identify clusters, and subclusters in big clusters, by means of visual inspection of the clustered gene expression data [1]. Moreover, genes that are adjacent in a linear ordering are often functionally co-regulated and involved in the same cellular process [2, 3] and biological analysis is often done in the context of this linear ordering [1].

Ideally, one would like to obtain a linear order of all genes that puts similar genes close to each other; such that for any two consecutive genes the distance between them is small. An optimal gene order can be obtained by minimizing the summation of gene expression distances (or maximizing summation of gene expression similarities) between pairs of adjacent genes in a linear ordering $1, 2, \cdots, n$. This can be formulated as [2]

$$F(n) = \sum_{i=1}^{n-1} C_{i,i+1}, \tag{1}$$

where n is the number of genes and $C_{i,i+1}$ is the distance/similarity between two genes i and $i+1$ obtained from distance/similarity matrix.

Though hierarchical clustering provides good gene order [3] by grouping co-regulated genes, there is still much room in improving gene order. A hybrid method (first clustering then ordering) for ordering genes for a hierarchical clustering solution is proposed in [1] where dynamic programming is applied to flip internal nodes to reorder the leaves in a hierarchical solution.

3 Materials and Methods

3.1 Preliminaries of Microarray Technology

In general, microarray data can be represented by a real valued matrix; each row represents a gene and each column (or a set of columns) represents a condition, or experiment. In cDNA (clone DNA) microarray-based investigations, RNA from experimental samples (taken at selected times during the process) is labeled during reverse transcription with the red-fluorescent dye Cy5 and is mixed with a reference sample labeled in parallel with the green-fluorescent dye Cy3 [3]. After hybridization and appropriate washing steps, separate images/spots are acquired for each fluor, and fluorescence intensity ratios are obtained for all target elements. If R (red) and G (green) are the spot-specific, quantitated, fluorescent intensities of the target and reference expression signals respectively, relative gene expression is defined as the log ratio $M = log_2 \frac{R}{G}$. For microarray data table each cell represents the M value at the corresponding target element [3] obtained from the gene under that experimental condition.

Fluorescence is currently the predominant method for microarray signal detection [5]. A critical component of a fluorescence scanner is the photomultiplier tube (PMT), in which fluorescent photons produce electrons that are amplified by the PMT voltage, also referred to as the PMT gain. For many microarray scanners, the PMT gain is an easily adjustable parameter, and the calibration curve (i.e., the curve showing the relationship between dye concentration and fluorescence intensity) depends on the gain setting [5]. This PMT gain is also varied for different types of experiments of different biological origin. DNA microarray measurements normally assume a linear relationship between detected fluorescent signal and the concentration of the fluorescent dye. Each PMT has its own linear dynamic range within which signal intensity increases linearly with the increase of fluorescent dye concentration [5]. This linear dynamic range also fixes the dynamic range of the recorded microarray data (log ratio values) within which the data values are most reliable and used as the normalization factor in the proposed distance measure to remove variations of biological origin. For example, in Cell Cycle related experiments, for dye Cy5, PMT gain at 960 volts fixes the intensity range from x1 to x2, and for dye Cy3, PMT gain at 760 volts fixes the intensity range from y1 to y2. So the linear dynamic range of PMT fixes the linear dynamic range of the data from $log_2 \frac{x1}{y1}$ to $log_2 \frac{x2}{y2}$. Note that, this dynamic range is available either from the supplementary information (website) of the article/data (Yeast datas), or upon request to the authors (Herpes data) and not from the datasets, and hence is not sensitive to outliers. The proposed dynamic range based normalization belongs to the category of between-slide or multiple-slide normalization with two other members median absolute deviation (MAD) and variance regularization. The MAD and variance regularization are dynamic range estimators (not the real one) and are also implemented for the purpose of comparison. However, the results obtained were similar to without any normalization.

3.2 Description of Data Sets

For gene ordering, data sets like Cell Cycle [4], Yeast Complex [3, 1], All Yeast [3], and Herpes [7] are chosen. Table 1 shows the name of the data sets, number of genes in each dataset, number of gene categories, name of experiment types and number of experiments performed under each type, and finally the total number of experiments performed for a particular dataset. The dynamic range of expression values of each experiment is shown within parenthesis. The dynamic range of available data represents log ratios of -1.2 to 1.2 for the cell-cycle experiments, -3.0 to 3.0 for sporulation, -1.5 to 1.5 for the shock experiments, -2.0 to 2.0 for the diauxic shift, and -13.0 to 13.0 for Herpes data. The first three data sets of Saccharomyces cerevisiae consists of about 652, 979 and 6221 genes, and 184, 79 and 80 microarray experiments respectively. The genes in the three data sets are classified according to MIPS [6] categorization into 16, 16, and 18 groups respectively. Herpes virus genes are broadly assigned to five functional groups and available in [7].

Table 1. Summary for different microarray data sets

Dataset	No. of genes	Category	Experiments performed				Total
Cell Cycle	652	MIPS 16	Cell Cycle (-1.2 to 1.2) 93	sporulation (-3.0 to 3.0) 9	shock (-1.5 to 1.5) 56	diauxic shift (-2.0 to 2.0) 26	184
Yeast Complex	979	MIPS 16	Cell Cycle (-1.2 to 1.2) 18+14+15	sporulation (-3.0 to 3.0) 7+4	shock (-1.5 to 1.5) 6+4+4	diauxic shift (-2.0 to 2.0) 7	79
All Yeast	6221	MIPS 18	Cell Cycle (-1.2 to 1.2) 60	sporulation (-3.0 to 3.0) 13	diauxic shift (-2.0 to 2.0) 7		80
Herpes	106	GeneBank 5	No KSHV (-13.0 to 13.0) 1	-TPA (-13.0 to 13.0) 7	TPA (-13.0 to 13.0) 13		21

3.3 New Distance Measure

A natural basis for organizing gene expression data is to group together genes with similar patterns of expression. The first step to this end is to adopt a mathematical description of distance. A number of measures of distance in the behavior of two genes can be used, such as the Manhattan distance [8], Euclidean distance [8], Pearson Correlation distance [2]. These distance measures usually take the same normalization factor (like standard deviation for Pearson correlation) for a gene. This normalization factor is independent of the type of experiment and performs global normalization to all the expression values for a particular gene; thus loosing useful local information. But, a closer look at the gene expression data reveals that the dynamic range of expression values differs with the type of experiment, and remains the same for all the genes in the dataset. So, using the same normalization factor is undesirable for all types of experiments, where expression values differ by orders of magnitude from one kind of experiment to another. Consequently, it may be appropriate and better if the normalization is performed

- separately for the different types of experiment with different normalizing factors; thereby preserving the local information
- keeping the same set of normalization factors for all the genes in the dataset.

Such an attempt is made in this article where two new distance measures are developed using Manhattan distance and Euclidean distance respectively (to avoid over sensitivity to three fold changes), in which the normalization is dependent on the type of experiment. This, in turn, results in equal weighting of distance values for different experiment types. The normalization factor is chosen as the linear dynamic range of data values obtained from photo multiplier tube, for a particular type of experiment.

Let
$X = x_1^{e_1}, \cdots, x_{i_1}^{e_1}, x_1^{e_2}, \cdots, x_{i_2}^{e_2}, \cdots, x_1^{e_m}, \cdots, x_{i_m}^{e_m}$ and
$Y = y_1^{e_1}, \cdots, y_{i_1}^{e_1}, y_1^{e_2}, \cdots, y_{i_2}^{e_2}, \cdots, y_1^{e_m}, \cdots, y_{i_m}^{e_m}$
be the expression levels of the two genes in terms of log-transformed microarray gene expression data obtained over a series of m different types of experiment $(e_1, e_2, \cdots e_m)$ consisting of $i_1 + i_2 + \cdots + i_m$ experiments in total. Using Manhattan distance the *Maxrange* distance between X and Y is defined as

$$\textit{Maxrange-M}_{X,Y} = \frac{1}{m}\sum_{r=1}^{m}\frac{1}{i_r} \times \frac{\sum_{j=1}^{i_r}|x_j^{e_r} - y_j^{e_r}|}{Max_{e_r} - Min_{e_r}} \quad (2)$$

where, Max_{e_r} and Min_{e_r} are the maximum and minimum $log_2(R/G)$ values obtained from the linear dynamic range of the photo multiplier tube (or radioactive probe) for an experiment of type e_r.

Using the Euclidean distance the *Maxrange* distance between X and Y is defined as

$$\textit{Maxrange-E}_{X,Y} = \frac{1}{m}\sum_{r=1}^{m}\frac{1}{i_r} \times \frac{\sqrt{\sum_{j=1}^{i_r}(x_j^{e_r} - y_j^{e_r})^2}}{Max_{e_r} - Min_{e_r}} \quad (3)$$

Throughout the literature we have used *Maxrange-M* and *Maxrange-E* for representing *Maxrange* distance measure using Manhattan and Euclidean distance respectively.

4 Biological Interpretation

A biological score, that is different from the similarity/distance measures, is used to evaluate the final gene ordering. Each gene that has undergone MIPS categorization can belong to one or more category, while there are many unclassified genes also (no category). A vector $V(g) = (v_1, v_2, \cdots, v_j)$ is used to represent the category status of each gene g, where j is the number of categories. The value of v_j is 1 if gene g is in the jth category; otherwise is zero. Based on the information about categorization, the score of a gene order for multiple class genes is defined as [9]

$$S(n) = \sum_{i=1}^{N-1} G\left(g_i, g_{i+1}\right), \tag{4}$$

where N is the number of genes, g_i and g_{i+1} are the adjacent genes and $G\left(g_i, g_{i+1}\right)$ is defined as

$$G\left(g_i, g_{i+1}\right) = \sum_{k=1}^{j} V(g_i)_k V(g_{i+1})_k, \tag{5}$$

where $V(g_i)_k$ represents the k^{th} entry of vector $V(g_i)$. Note that, $S(n)$ can also be used as scoring function for single class genes like Herpes genes. Using scoring function $S(n)$, a gene ordering would have a higher score when more genes within the same group are aligned next to each other. So higher values of $S(n)$ are better and can be used to evaluate the goodness of a particular gene order. Note that, although these scoring functions provide a good quantitative index for gene ordering, using $S(n)$ as the similarity measure in ordering is not practical, since the information about gene categories is unknown for most of the genes in the real world.

5 Experimental Results

Algorithms of gene ordering and clustering are implemented using mex files in Matlab 7 on Sun Fire V 890 (1.2 GHz and 8 GB RAM). The codes for single, average and complete linkage and Bar-Joseph et al.'s [1] method are downloaded from [10]. Performance of the proposed *Maxrange-M* and *Maxrange-E* distance are compared with Pearson correlation, Euclidean distance, and Manhattan distance.

5.1 Comparative Performance of Distance Measures

Table 2 compares the performance of our proposed measure with those of the other measures in terms of the $S1$ value (Eq. 4). Three distance measures are considered, namely, *Maxrange-M*, Pearson and Euclidean. The biological scores corresponding to Manhattan Distance are found to be comparable to those for Pearson Correlation, and hence omitted here. The percentages of improvement over the lowest biological score (in terms of $S1$ value) in a particular data set are shown within parenthesis, and defined as:

$$PI_{i,j} = \frac{d_{i,j} - min_i(d_{i,j})}{min_i(d_{i,j})} \times 100 \tag{6}$$

where, $d_{i,j}$ indicates the biological score ($S1$ value) in ith row and jth column of the result matrix in Table 2, and $min_i(d_{i,j})$ indicates the minimum biological score in column j for all i. These PI values in Table 2 are used in the next section for conducting t-tests.

Though in most of the cases *Maxrange-E* distance is found to be superior to Euclidian distance and inferior to *Maxrange-M*, for All Yeast data, it performs better ($S(n)$=2441) than *Maxrange-M* ($S(n)$=2341) for average linkage algorithm. When the microarray data sets contain experiments with data value of same dynamic range, like Herpes, then *Maxrange-M* provides identical results with Manhattan distance for all widely used ordering algorithms. However the superiority of *Maxrange-M* is evident when different types of experiments are present in a particular microarray data. For example, superior results are obtained with *Maxrange-M* for most of the available algorithms for the Cell Cycle, Yeast complex and All Yeast data sets (shown in first row for each algorithm in Table 2).

5.2 Statistical Analysis of *Maxrange-M* Distance Measure

To statistically compare the performance of *Maxrange-M* distance with Pearson Correlation in case of ordering algorithms, t-tests are performed with the PI (Eq. 6) values shown within parenthesis in Table 2, using the equation

$$t = \frac{\overline{PI_1} - \overline{PI_2}}{\sqrt{\frac{VariancePI_1}{n_1} + \frac{VariancePI_2}{n_2}}}. \tag{7}$$

where, $\overline{PI_1}$ and $VariancePI_1$ are the mean and the variance of all the available PI values for *Maxrange-M* distance in Table 2. PI_2 is used for Pearson Correlation and $n_1 = n_2 = 16$, as there are 16 PI values available in total from Table 2 for each of the distance measures with 4 datasets and 4 algorithm. So, the degrees of freedom for t-test are $16 \times 2 - 2 = 30$. Similarly, t-test is also performed for *Maxrange-M* distance and Euclidean distance. The two t values and related p values are shown in Table 3. The alternative hypothesis (H_1), that the average of 'percentages of improvement over the lowest biological score' for the *Maxrange-M* distance is better than the related one (Pearson or Euclidean), is used in the calculation of t-statistics. The final conclusion, once the test has been carried out, is always given in terms of the null hypothesis (H_0), that there is no difference between the averages of 'percentages of improvement over the lowest biological score' for the two distance measures. After finding the p values (from t-table)

Table 2. Biological Score ($S(n)$) and Percentage of Improvement (PI) value (within parenthesis) for different distance measures and algorithms

		Data Sets			
Distance	Algorithm	Cell cycle	Yeast complexes	All Yeast	Herpes
Maxrange-M	Bar-Joseph	423 (17.83)	1074 (26.50)	2371 (22.85)	43 (19.44)
	Average Linkage	415 (15.60)	1040 (22.50)	2341 (21.30)	39 (8.33)
	Complete Linkage	407 (13.37)	1043 (22.85)	2305 (19.43)	38 (5.56)
	Single Linkage	382 (6.41)	903 (6.36)	1970 (2.07)	41 (13.89)
Pearson	Bar-Joseph	381 (6.13)	1024 (20.61)	2350 (21.76)	38 (5.56)
	Average Linkage	385 (7.24)	987 (16.25)	2292 (18.76)	38 (5.56)
	Complete Linkage	393 (9.47)	955 (12.49)	2301 (19.22)	36 (0.00)
	Single Linkage	359 (0.00)	902 (6.24)	1973 (2.23)	39 (8.33)
Euclidean	Bar-Joseph	421 (17.27)	1013 (19.32)	2346 (21.55)	40 (11.11)
	Average Linkage	403 (12.26)	1011 (19.08)	2431 (25.96)	39 (8.33)
	Complete Linkage	403 (12.26)	999 (17.67)	2269 (17.56)	37 (2.78)
	Single Linkage	361 (0.56)	849 (0.00)	1930 (0.00)	36 (0.00)

Table 3. Results of t-test for different pairs of distance measures

	Pairs of distance measure	
	Maxrange-M & Pearson	*Maxrange-M* & Euclidean
t	2.0134	1.2709
p	$0.027 > p$	$0.107 > p$

for corresponding t values, we reject the null hypothesis for both the cases with significance level 0.027 and 0.107 respectively, which suggests that there is strong evidence against the null hypothesis in favor of the alternative.

6 Conclusion

A new measure called *Maxrange*, for evaluating the distance between genes, is used for efficiently ordering the genes in terms of their expression values for microarray datasets. The available measures for gene distance, like Manhattan Distance, Euclidean distance, and Pearson correlation, use only one normalization factor (1, 1, and standard deviation respectively) for all types of experiments, although the expression values may differ by orders of magnitude from one kind of experiment to another. As a consequence, the distance between genes may not be properly reflected in these measures for microarray data having different types of experiments. In contrast, normalization is performed separately with different normalizing factors for the different types of experiment in our *Maxrange-M* and *Maxrange-E* distance. This makes it, suitable for both single type and multiple type of experiments and, promising for microarray gene expression related experiments.

References

[1] Z. Bar-Joseph, D. K. Gifford, and T. S. Jaakkola. Fast optimal leaf ordering for hierarchical clustering. *Bioinformatics*, 17:22–29, 2001.

[2] T. Biedl, B. Brejov, E. D. Demaine, A. M. Hamel, and T. Vinar. Optimal arrangement of leaves in the tree representing hierarchical clustering of gene expression data. Technical Report 2001-2014, Dept. Computer Sci., Univ. Waterloo, 2001.

[3] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc. National Academy of Sciences*, 95:14863–14867, 1998.

[4] G. S. et al. The stanford microarray database. *Nucleic Acids Research*, 29(1):152–155, 2001.

[5] L. S. et al. Microarray scanner calibration curves: characteristics and implications. *BMC Bioinformatics*, 6((Suppl2):S11):1–14, 2005.

[6] M. I. for Protein Sequences. http://www.mips.com.

[7] R. G. Jenner, M. M. Alb, C. Boshoff, and P. Kellam. Kaposi's sarcoma-associated herpesvirus latent and lytic gene expression as revealed by dna arrays. *Journal of Virology*, 75(2):891–902, 2001.

[8] E. F. Krause. Taxicab Geometry: An Adventure in Non-Euclidean Geometry. 1986.

[9] H. K. Tsai, J. M. Yang, Y. F. Tsai, and C. Y. Kao. An Evolutionary Approach for Gene Expression Patterns. *IEEE Trans. on Info. Tech. in Biomedicine*, 8(2):69–78, 2004.

[10] D. Venet. MatArray: a Matlab toolbox for microarray data. *Bioinformatics*, 19(5):659–660, 2003.

Predicting Gene Function in Yeast through Adaptive Weighting of Multi-Source Information

Shubhra Sankar Ray [1,*], Sanghamitra Bandyopadhyay [2], Sankar K. Pal [1]

1. Center for Soft Computing Research, Indian Statistical Institute, Kolkata, India
2. Machine Intelligence Unit, Indian Statistical Institute, Kolkata, India
*E-mail: shubhra_r@isical.ac.in

Background

The value of combining informations obtained from different methods, for gene function predictions, has been illustrated by several studies [1, 2, 3, 4]. We propose a new scoring framework, called *Adaptive Score* (*AdS*), for predicting the function of a few unclassified Yeast genes. We mainly focus on phenotypic profiles [5], microarray gene expression (All Yeast data [6]), KEGG pathway database [7], protein sequence similarity through transitive homologues, and protein-protein interactions from BioGRID [8] as data-sources. We use the Pearson correlation for similarity extraction from phenotypic profile and gene expression data. All the protein sequences, except Yeast proteins, corresponding to each KEGG pathway (121 pathways in the second level) are downloaded from PIR to extract profile similarity between two Yeast proteins. Profile vector for each protein in Yeast is computed by comparing its sequence across 121 pathway databases, using BLAST. The method is similar to phylogenetic profile [9] construction. To find the similarity between two genes using KEGG profiles, we used the ratio of dot product value and OR value between two profiles. To detect similarity between two proteins sequences through transitive homologues, 37,66,477 protein sequences are downloaded from UniProt and compared with target proteins by using BLAST [10], the metric of ProClust [11], and the method described in [12]. For protein-protein interaction study, manually curated catalogues of known interactions are downloaded from BioGRID [8] and binary interactions are used as the common unit of analysis.

Benchmarking

The similarities arising from various data-sources are separately benchmarked, based on the super GO-Slim process annotations of genes in the Saccharomyces Genome Database (SGD). The proportion of true positives (TP) gene-pairs at a particular similarity value (computed from a data-source) can be used as a benchmarking method [1], where TP gene-pairs are defined as pairs of genes i and j, such that genes i and j have an overlapping (explicit or implicit) super GO-Slim process term annotation. Figure 1.a compares the similarity values obtained from different data-sources in terms of their $proportionTP$. The $proportionTP$ values for intermediate similarity values are calculated from the slopes of the respective curves. The $proportionTP$ for protein-protein interactions has a constant value 0.69 at a similarity value of 1 and hence it is not shown in Fig. 1.a.

New Scoring Framework

The $proportionTP$ values reflect the accuracy of similarity values, but do not provide any information about importance/weight of one data-source in presence of the other data-sources, in predicting gene-pairs. Consequently, it will be more appropriate if *proportionTP* values of each data-source, in presence of other data-sources, is weighed by a different factor and then integrated; and the factors are dependent on the *proportionTP* of the integrated *proportionTP* values of different data-sources. In this investigation we propose a new score where, *proportionTP* values (computed from different data-sources) between two genes X and Y are added through weights a, b, c, d, and e in a linear combination style. This score is referred to as *Adaptive Score* (*AdS*) and is defined as

$$AdS_{X,Y} = \frac{a \times Pp_{X,Y} + b \times Pm_{X,Y} + c \times Kp_{X,Y} + d \times B_{X,Y} + e \times I_{X,Y}}{a+b+c+d+e} \tag{1}$$

where a, b, c, d, and e are varied within range 0 to α in steps of 1 to find a combination that maximizes the *proportionTP* (using super GO-Slim process) for a user defined cut-off of top gene-pairs. The weights a, b, c, d, and e are assigned to the complete *proportionTP* matrices calculated from individual data-sources. Our gold standard cut-off (user defined) of top gene-pairs is determined from KEGG pathway profiles, which provides 26432 gene-pairs with similarity value 1 and gold standard constant *proportionTP* value of .81. These gene-pairs are the most accurate of all, whereas the accuracy (*proportionTP*) of other data-sources, as well as gene-pairs below top 26432 for KEGG pathway profiles, vary considerably.

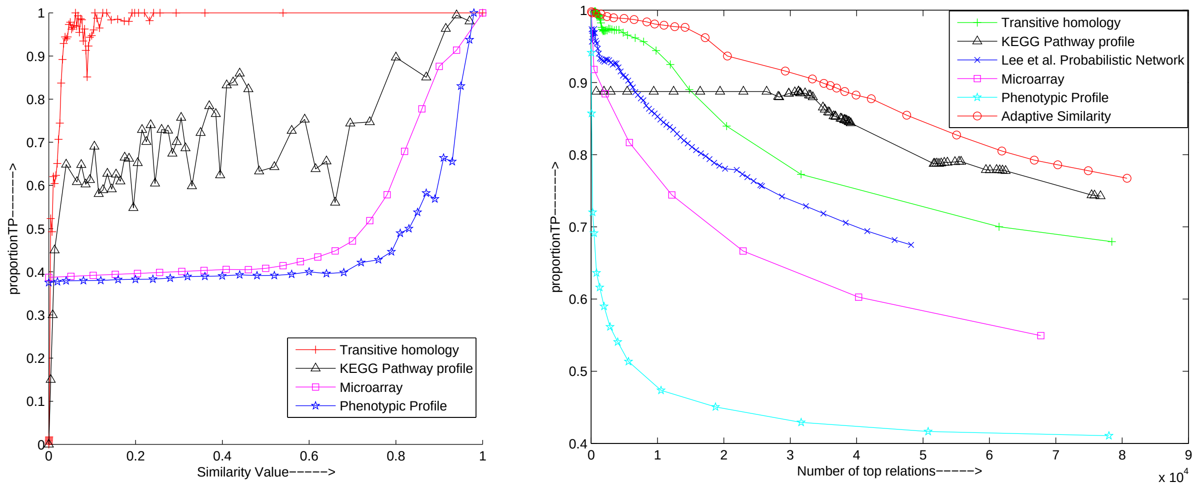
Evaluation

As super GO-Slim process was used for determining the weights of the data-sources, top level classification of MIPS October 2005 annotation is now used to evaluate the performance of *AdS*. We sorted the similarity values computed from different data-sources in descending order, and drew a curve for top gene-pairs verses *proportionTP* from the sorted data for each form of data-source (Fig. 1.b). In contrast, *proportionTP* for protein-protein interactions has a constant value of 0.69 and not shown in Fig.1.b. Figure 1.b also compares the performance of *AdS* and 'final log likelihood scores' of Lee et al.'s [3] probabilistic network (downloaded from the website mentioned in [13]) in terms of *proportionTP* with MIPS annotation. From the figure it is clear that the gene-pairs identified in this investigation is better than any other existing network or data-sources. The top $1,00,000$ gene-pairs predicted by our method with *proportionTP* values above 0.755 are available at http://www.isical.ac.in/˜scc/Bioinformatics/AdS/toprelation.txt in tabular form. The *proportionTP* values computed from individual data-source are also shown in the file.

Results and Discussions

Genes are considered to be linked if they are among the 10 closest neighbors within a given distance or similarity cut-off [2]. Genes are clustered with a method based on the K Nearest Neighbors (KNN) algorithm [14] by selecting $K = 10$ and using *AdS* with gold standard cut-off value 0.77. The method is denoted as *KNN-AdS*. The biological significance of the clusters generated by our *KNN-AdS* is evaluated with 400 different MIPS functional categories. Clusters with P-values greater than 10^{-5} are not reported. 2507 clusters are identified with at-least three or more members. Out of these clusters, 1915 clusters are identified with functional enrichment in one or more categories. From functionally enriched clusters we predict the functions of 1855 classified genes (with 95.16% accuracy) and 60 unclassified genes by assigning the function related with the smallest P-value. The functional enrichment, in one or more categories, for clusters intended for 60 unclassified yeast genes are available in tabular form at http://www.isical.ac.in/~scc/Bioinformatics/AdS/unclassifiedprediction.xls. The function with the smallest P-value in the table represents the predicted function for the unclassified gene, and the three values in the parenthesis denote the function related P-value, no. of genes in the cluster, no. of genes in the genome, respectively. The table also includes all the genes within each cluster, the $proportionTP$ values arising from various data-sources, and the AdS values. A table containing the predicted functions of 1855 classified yeast genes is available at http://www.isical.ac.in/~scc/Bioinformatics/AdS/classifiedprediction.xls. Out of 60 unclassified genes, YEL041W and YDR459C are now (April 2007) classified in MIPS, and our function predictions for these two genes are in agreement with present MIPS classification. YEL041w is related with the category 'phosphate metabolism' (4 out of 5 genes, p-value 1.42×10^{-6}). YDR459C is related with category 'modification with fatty acids' (4 out of 11, P-value 2.3×10^{-7}). Our top predictions consist the function of 12 unclassified (MIPS 2007) and 417 classified genes at P-value cut-off 1×10^{-13} and with 98.20% accuracy. The related Table for top 12 predictions and discussions are available at http://www.isical.ac.in/~scc/Bioinformatics/AdS/top12predictions.pdf.

Figures



a) b)

Figure 1: a) $proportionTP$ Vs. similarity values for different types of data-sources. b) Comparing *Adaptive Score* and individual data-source in terms of $proportionTP$ versus the number of top gene-pairs.

References

At http://www.isical.ac.in/~scc/Bioinformatics/AdS/top12predictions.pdf the references are available.