

# 國立交通大學

生物資訊研究所

碩士論文

運用智慧型基因演算法最佳化微陣列資料  
分析 - 可語意解讀基因表現量分類器之設  
計暨基因網路模型之重建

Intelligent Genetic Algorithm for Microarray Data  
Analysis – Interpretable Gene Expression Classifier and  
Inference of Genetic Network

研究生：謝志宏

指導教授：何信瑩 教授

中華民國九十五年六月

運用智慧型基因演算法最佳化微陣列資料分析 –  
可語意解讀基因表現量分類器之設計暨  
基因網路模型之重建

Intelligent Genetic Algorithm for Microarray Data Analysis –  
Interpretable Gene Expression Classifier  
and Inference of Genetic Network

研究生：謝志宏

Student : Chih-Hung Hsieh

指導教授：何信瑩

Advisor : Shinn-Ying Ho



A Thesis Submitted to Institute of Bioinformatics  
National Chiao Tung University in partial Fulfillment of the Requirements  
for the Degree of Master in  
Bioinformatics

June 2006

Hsinchu, Taiwan, Republic of China

中華民國九十五年六月

# 運用智慧型基因演算法最佳化微陣列資料分析 – 可語意

## 解讀基因表現量分類器之設計暨基因網路模型之重建

學生：謝志宏

指導教授：何信瑩

國立交通大學生物資訊研究所

### 摘要

在癌症及疾病醫學診斷的研究之中，微陣列基因表現量資料分析可說是目前最重要的研究領域之一。基因表現量資料可提供有關基因、基因調控網路、及細胞內狀態之豐富資訊，藉由微陣列資料分析之技術，我們可以由基因表現量資料中，篩選出參與基因調控的重要基因，並且能夠重建出細胞內動態化的基因調控網路，進而探索並發現更多有關分子生物學、生物化學、生化工學及製藥學的重要新知識。進行微陣列資料分析其中兩個主要目的分別為：探索針對不同的細胞狀態中，各基因的表現情形分別為何？例如，在健康細胞及癌細胞中，各基因所分別表現之狀態；以及研究同一基因調控網路內之基因其彼此調控影響的關係。而上述微陣列資料分析的兩個主要議題，可以分別歸類為基因表現量分類問題及基因調控網路重建問題。

首先，當在面對基因表現量分類問題時，一個精準、只需少數基因資訊即可運作、並且可用自然語意解讀其學習結果之分類器，對於微陣列資料分析以及其後具經濟效益的醫學檢測，將有決定性之幫助。然而，許多常用於微陣列資料分析之分類器，例如：支持向量機(SVM)、類神經網路、 $k$  個最近鄰居分類法( $k$ -NN)以及羅吉斯回歸模型皆缺少良好之可用自然語意解讀的特性。因此，於此篇論文之中，對於基因表現量分類問題，我們提出一以精確且精簡之模糊分類規則為基礎，並且可以語意解讀之基因表現量分類器(iGEC)。iGEC 包含三個主要之最佳化設計目標分別為最大化分類辨識率、最少化所需分類規則、以及最少化分類所需基因數，並且採用一新式智慧型基因演算法 (IGA) 有效率地解決含有大量調控參數之 iGEC 最佳化設計問題。進一步，我們使用八組常用的基因表現資料來做效能評估。實驗結果顯示 iGEC 可有效產生一組精確、精簡、且語意

可解讀的模糊分類規則(平均一個類別只需 1.1 條模糊規則), 其中平均測試階段辨識率為 87.9%, 平均所需模糊分類規則數為 3.9, 平均分類所需基因數為 5.0。此外, 針對基因表現量分類問題, 根據上述的評量標準, iGEC 不但較現有之模糊規則分類器有更佳的表現, 對於某些不以分類規則為基礎的分類器, iGEC 同樣具有更精確之辨識率。

其次, 針對基因調控網路重建問題, 我們希望利用基因表現量資料, 藉由有效重建動態化的基因調控網路來發現更多有關分子生物學、生物化學等的重要知識。其中, S-system 基因網路模型不但適合用來描述生化網路系統, 更可用來分析調控網路內部動態變化之情形。然而要推算出一個含有  $N$  個基因的 S-system 基因網路模型就必須處理含有  $2N(N+1)$  個調控參數之非線性微分方程組, 此為一大量參數最佳化問題, 需耗費大量的計算成本。因此, 我們於此篇論文中, 提出一智慧型兩階段演化式演算法(iTEA), 有效率地由時間序列的基因表現量資料重建出 S-system 基因網路模型。為了處理如此大量的調控參數, iTEA 演算法主要可分為兩個分別採用 divide-and-conquer 策略之階段。首先將此最佳化問題分割為  $N$  個含有  $2(N+1)$  調控參數的子問題。於 iTEA 第一階段時, 使用以直交實驗設計 (OED) 為基礎之新式智慧型基因演算法 (IGA) 最佳化決定每一個子問題之解。再者, 為了處理基因表現量資料含有雜訊的問題, 於第二階段時, 結合  $N$  個子問題之解組成含有  $2N(N+1)$  個參數之 S-system 網路模型, 再利用另一以 OED 為基礎之新式退火演算法 (OSA) 做進一步的最佳化調整。我們利用單 CPU 電腦, 並且使用模擬產生不含及含有雜訊的基因表現量資料來對 iTEA 做效能評估。實驗結果顯示: (1) IGA 能夠有效地解決含有含有  $2(N+1)$  調控參數的子問題; (2) 相較於前人所採用 SPXGA 演算法, IGA 明顯具有更好的最佳化搜尋能力; (3) iTEA 能夠有效率地解決 S-system 基因調控網路模型的重建問題。

**關鍵詞:** 演化式演算法; 智慧型基因演算法; 直交實驗設計; Divide-and-conquer; 型樣識別; 模糊分類器; 基因表現量; 微陣列資料分析; 基因調控網路; 生化途徑識別; S-system 基因網路模型。

# Intelligent Genetic Algorithm for Microarray Data Analysis – Interpretable Gene Expression Classifier and Inference of Genetic Network

Student: Chih-Hung Hsieh

Advisor: Shinn-Ying Ho

Institute of Bioinformatics  
National Chiao Tung University

## Abstract

Microarray gene expression profiling technology is one of the most important research topics in cancer research or clinical diagnosis of disease. The gene expression data provide valuable information in the understanding of genes, biological networks, and cellular states. Through microarray techniques, we can find out the important genes which participate in the genetic regulation and rebuild cellular dynamic regulation networks from gene expression data to discover more delicate and substantial functions in molecular biology, biochemistry, bioengineering, and pharmaceuticals. One goal in analyzing expression data is to determine how genes are expressed as a result of certain cellular conditions (e.g., how genes are expressed in diseased and healthy cells). Another goal is to determine how the expression of any particular gene might affect the expression of other genes in the same genetic network. To achieve the two objectives of microarray data analysis mentioned above, two of the important issues in microarray data analysis are the gene expression classification and the genetic networks inference problem.

First, when dealing with the gene expression classification problem, an accurate classifier with linguistic interpretability using a small number of relevant genes is beneficial to microarray data analysis and development of inexpensive diagnostic tests. Several frequently used techniques for designing classifiers of microarray data, such as support vector machine, neural networks,  $k$ -nearest neighbor rule, and logistic regression model, suffer from low interpretabilities. This thesis proposes an interpretable gene expression classifier (named iGEC) with an accurate and compact fuzzy rule base for microarray data analysis. The design of iGEC has three objectives to be simultaneously optimized: maximal classification accuracy, minimal number of rules, and minimal number of used

genes. A novel intelligent genetic algorithm (IGA) is used to efficiently solve the design problem with a large number of tuning parameters. The performance of iGEC is evaluated using eight commonly-used data sets. It is shown that iGEC has an accurate, concise, and interpretable rule base (1.1 rules per class) on average in terms of test classification accuracy (87.9%), rule number (3.9), and used gene number (5.0). Moreover, iGEC not only has better performance than the existing fuzzy rule-based classifier in terms of the above-mentioned objectives, but also is more accurate than some existing non-rule-based classifiers.

Second, for the genetic networks inference problems, it is desirable to rebuild the relationships of regulation between genes from gene expression profiles. S-system model is suitable to characterize biochemical network systems and capable to analyze the regulatory system dynamics. However, inference of an S-system model of  $N$ -gene genetic networks has  $2N(N + 1)$  parameters in a set of non-linear differential equations to be optimized. This thesis proposes an intelligent two-stage evolutionary algorithm (iTEA) to efficiently infer the S-system models of genetic networks from time-series data of gene expression. To cope with curse of dimensionality, the proposed algorithm consists of two stages where each uses a divide-and-conquer strategy. The optimization problem is first decomposed into  $N$  subproblems having  $2(N + 1)$  parameters each. At the first stage, each subproblem is solved using the novel intelligent genetic algorithm (IGA) with intelligent crossover based on orthogonal experimental design (OED). At the second stage, the obtained  $N$  solutions to the  $N$  subproblems are combined and refined using an OED-based simulated annealing algorithm for handling noisy gene expression profiles. The effectiveness of iTEA is evaluated using simulated expression patterns with and without noise running on a single-processor PC. It is shown that 1) IGA is efficient enough to solve subproblems; 2) IGA is significantly superior to the existing method SPXGA; and 3) iTEA performs well in inferring S-system models for dynamic pathway identification.

**Keywords:** Evolutionary algorithm; Intelligent genetic algorithm; Orthogonal experimental design; Divide-and-conquer; Pattern recognition; Fuzzy classifier; Gene expression; Microarray data analysis; Genetic network; Pathway identification; S-system model.



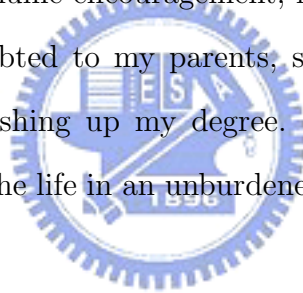
# Acknowledgements

First of all, I would like to express my sincere gratitude to my advisor, Professor Shinn-Ying Ho. I learned so much from him in every aspect. Without his guidance, work ethic, motivation and support, I would not be at this juncture in life.

I would like to thank all the professors who help me in every aspect during these two years.

Moreover, I would like to thank my labmates in the Intelligent Computing Lab. and all of my friends, for their genuine encouragement, kind help, and sweet memories.

Finally, I am deeply indebted to my parents, sister, and ewen for their love which supports me all the way finishing up my degree. Without them and their love, it is impossible for me to exploit the life in an unburdened way. For this, I am forever in their debt.





# Contents

<b>Abstract</b>	<b>i</b>
<b>Acknowledgements</b>	<b>vi</b>
<b>Table of Contents</b>	<b>vii</b>
<b>List of Figures</b>	<b>x</b>
<b>List of Tables</b>	<b>xii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Survey of the Related Works . . . . .	2
1.2.1 Gene Expression Classification Problems . . . . .	2
1.2.2 Genetic Network Inference Problems . . . . .	3
1.3 Sketch of the Thesis . . . . .	5
1.3.1 An Interpretable Gene Expression Classifier for Gene Expression Classification . . . . .	5
1.3.2 An Intelligent Two-stage Evolutionary Algorithm for Inference of Genetic Network . . . . .	6
1.4 Organization . . . . .	7
<b>2 Background</b>	<b>8</b>
2.1 Classifiers for Gene Expression Classification . . . . .	8
2.1.1 Neural Networks (NN) . . . . .	8
2.1.2 $k$ -nearest-neighbor Rule ( $k$ -NN) . . . . .	10



2.1.3	Support Vector Machine (SVM)	10
2.1.4	Fuzzy Rule-Base Classifier	11
2.2	Models of Genetic Network	14
2.2.1	Boolean Network Model	14
2.2.2	Bayesian Network Model	14
2.2.3	Linear Differential Network Model	15
2.2.4	S-system Network Model	15
2.3	Genetic Algorithm (GA)	16
2.3.1	Encoding Scheme and Fitness Function	17
2.3.2	Population Initialization	18
2.3.3	Selection	19
2.3.4	Crossover	21
2.3.5	Mutation	22
2.3.6	Termination Condition	23
<b>3</b>	<b>Intelligent Genetic Algorithm</b>	<b>24</b>
3.1	Concept of Orthogonal Experimental Design (OED)	24
3.2	Orthogonal Array	25
3.3	Factor Analysis	26
3.4	Intelligent Crossover	27
3.5	The Simple Intelligent Genetic Algorithm	27
<b>4</b>	<b>Interpretable Gene Expression Classifier</b>	<b>29</b>
4.1	The Proposed Interpretable Gene Expression Classifier (iGEC)	29
4.1.1	Flexible Generic Parameterized Membership Functions	29
4.1.2	Fuzzy Rule and Fuzzy Reasoning Method	31
4.1.3	Fitness Function and Chromosome Representation	32
4.1.4	The Used Intelligent Genetic Algorithm to Solve the Design Problem of iGEC	33
4.2	Experimental Results of iGEC	35

4.2.1	Implementation and Data Sets . . . . .	35
4.2.2	Experiment 1-Comparison between iGEC and the Vinterbo's Fuzzy Rule-based Classifier . . . . .	36
4.2.3	Experiment 2-Comparison between iGEC and Non-rule-based Classifiers . . . . .	43
4.3	Discussions of iGEC . . . . .	44
4.4	Conclusions for iGEC . . . . .	45
<b>5</b>	<b>Inference of Genetic Network</b>	<b>46</b>
5.1	The Investigated Problem . . . . .	46
5.1.1	Problem Statement . . . . .	46
5.1.2	Useful Techniques . . . . .	47
5.2	The Proposed Intelligent Two-stage Evolutionary Algorithm (iTEA) . . . . .	49
5.2.1	Orthogonal Experimental Design and Factor Analysis . . . . .	49
5.2.2	IGA for Solving Subproblems . . . . .	51
5.2.3	OSA for Refining the Combined Solution . . . . .	57
5.2.4	iTEA Using IGA and OSA . . . . .	60
5.3	Experimental Results of iTEA . . . . .	61
5.3.1	Experiment 1-Performance of IGA . . . . .	61
5.3.2	Experiment 2-Comparison between SPXGA and IGA . . . . .	64
5.3.3	Experiment 3-iTEA for noisy gene expression profiles . . . . .	65
5.4	Conclusions for iTEA . . . . .	69
<b>6</b>	<b>Conclusions</b>	<b>73</b>
6.1	iGEC for the gene expression classification problems . . . . .	73
6.2	iTEA for the genetic networks inference problems . . . . .	74
	<b>Bibliography</b>	<b>76</b>

# List of Figures

2.1	Illumination of the learning principle and testing behavior of SVM. . . . .	11
2.2	Illuminations of the three fuzzy partition methods. . . . .	13
2.3	Illuminations of the searching behaviors between genetic algorithm and traditional numerical method. . . . .	17
2.4	The flowchart of genetic algorithm. . . . .	18
2.5	An example of roulette wheel selection. . . . .	20
2.6	Illuminations of (a) one-point crossover, (b) multi-point crossover, and (c) uniform crossover. . . . .	22
2.7	An example of bit flip mutation. . . . .	23
4.1	Illuminations of FGPMF. . . . .	30
4.2	Examples of an antecedent fuzzy set $A_{ji}$ with linguistic values. . . . .	31
4.3	Chromosome representation. . . . .	33
4.4	The box plots of the statistical results of iGEC and the Vinterbo's classifier. . . . .	38
4.5	The 3D scatter plots of the statistical results of iGEC and the Vinterbo's classifier. . . . .	39
4.6	Fuzzy rules of the data set leukemia1 derived from iGEC. . . . .	40
4.7	The clustering result of 72 samples in data set leukemia1 using the three selected genes by the clustering algorithm EPCLUST. . . . .	42
5.1	Flowchart of the proposed two-stage evolutionary algorithm iTEA. . . . .	50
5.2	Flowchart of OSA with an intelligent generation mechanism. . . . .	58
5.3	The convergence comparison between IGA and SPXGA using 30 independent runs. . . . .	67

5.4 The distribution of 30 solutions with  $R = 1$  and one solution with  $R = 10$   
using IGA only without refinement of OSA. . . . . 71



# List of Tables

3.1	An Orthogonal Array of $L_8(2^7)$ . . . . .	26
4.1	The eight data sets for experiments of iGEC. . . . .	37
4.2	The statistical results of iGEC and the Vinterbo's classifier. . . . .	39
4.3	Selected genes for the leukemia1 data set example. . . . .	41
4.4	The test accuracies and numbers of used genes for iGEC and non-rule-based classifiers using 10-CV. . . . .	44
5.1	An Illustrative Example of Intelligent Crossover Using OA $L_8(2^7)$ . . . . .	54
5.2	The Contents of Parents and Children. . . . .	55
5.3	The S-system parameters of a small-scale target network with $N = 5$ . . . . .	62
5.4	15 Sets of initial gene expression levels of the target network with $N = 5$ . . . . .	62
5.5	The estimated S-system parameters sets with $N = 5$ . . . . .	63
5.6	S-system parameters of target network with $N = 10$ . . . . .	64
5.7	15 Sets of initial gene expression levels of the target network with $N = 10$ . . . . .	65
5.8	The obtained S-system parameters with $N = 10$ . . . . .	66
5.9	The fitness values for all subproblems with $N = 10$ . . . . .	66
5.10	T-test results for comparisons between IGA and SPXGA with various values of $N$ . . . . .	68
5.11	Performance of the proposed algorithm with OSA is performed 30 runs. . . . .	72

# Chapter 1

## Introduction

### 1.1 Motivation

Microarray gene expression profiling technology is one of the most important research topics in cancer research or clinical diagnosis of disease. The gene expression data provide valuable information in the understanding of genes, biological networks, and cellular states. Through microarray techniques, we can find out the important genes which participate in the genetic regulation and rebuild cellular dynamic regulation networks from gene expression data to discover more delicate and substantial functions in molecular biology, biochemistry, bioengineering, and pharmaceuticals. One goal in analyzing expression data is to determine how genes are expressed as a result of certain cellular conditions (e.g., how genes are expressed in diseased and healthy cells) [1]. Another goal is to determine how the expression of any particular gene might affect the expression of other genes in the same genetic network [2, 3, 4, 5].

To achieve the two objectives of microarray data analysis mentioned above, the most important issues in microarray data analysis are the gene expression classification and the genetic networks inference problem. In this thesis, we proposed two efficient algorithms to cope with these two important topics of microarray data analysis. Following is the introduction about the gene expression and the genetic networks inference problem, and the corresponding proposed algorithms to handle these two major problems in microarray data analysis.

## 1.2 Survey of the Related Works

### 1.2.1 Gene Expression Classification Problems

Given a large number of profiles contained thousands of genes in each experiment, we want to understand a global overview among lots of genes involved in the microarray experiments [6]. In such a case, gene expression classification was used to determine function for unknown genes [7], to look at expression programs for different systems in the cell [8] and for identifying sets of genes that are specifically involved in a certain type of cancer or other diseases [9]. Another major purpose in gene expression classification is effective data organization and visualization. It is thus not surprising that early work on gene expression analysis has focused on this level, and several classification algorithms have been suggested for gene expression data [10, 11].

The practical applications of microarray gene expression profiles include management of cancer and infectious diseases. There are many machine learning techniques, such as support vector machine (SVM), neural networks (NN),  $k$ -nearest neighbor rule ( $k$ -NN), and logistic regression have been used in gene expression data classification [12, 13]. However, due to the following three features about microarray data analysis, gene expression classification still remains difficult:

- 1) high dimensionality: there are thousands of genes (or features) in the microarray experiment;
- 2) few samples: compared with the number of genes, the number of samples was relatively few, usually fewer than one hundred;
- 3) given thousands of genes, only a small number of them show strong correlation with a certain phenotype [14].

Statnikov *et al.* investigated various classifiers which can handle data sets having multiple classes [12]. The results indicate that the multiclass SVM is the most effective classifier for tumor classification in terms of classification accuracy using large numbers of genes. However, given thousands of genes, only a small number of them show strong



correlation with a certain phenotype [14]. Unfortunately, it is intractable to identify the optimal subset from thousands of genes, while taking classification accuracy and linguistic interpretability into account.

Liu et al. proposed a feature selection method which combines top-ranked, test-statistic, and principle component analysis in conjunction with ensemble NN to design classifiers [15]. Zhou and Mao suggested a filter-like evaluation criterion, called LS Bound measure, derived from leave-one-out procedure of least squares support vector machines (LS-SVMs), which provides gene subsets leading to more accurate classification [16]. Liu et al. combined the entropy-based feature (gene) selection method using simulated annealing and  $k$ -NN classifier for cancer classification [17].

To advance the classification performance using a small number of genes, it is better to take both gene selection and classifier design into account simultaneously. Li et al. proposed a hybrid method of the genetic algorithm (GA)-based gene selection and  $k$ -NN classifier to assess the importance of genes for classification [18]. Ooi and Tan proposed a GA/MLH (maximal likelihood)-based method for the multicategory prediction of gene expression data [19].

An accurate classifier with linguistic interpretability is beneficial to microarray data analysis. However, the learning results of the above-mentioned classifiers cannot be summarized into human-interpretable forms for biologists and biomedical scientists [13]. Li et al. used a tree structure to classify the microarray samples [20]. Hvidsten et al. proposed learning rule-based models of biological process from gene expression time profiles using gene ontology [21]. Vinterbo et al. presented a rule-induction and filtering strategy to obtain an accurate, small, and interpretable fuzzy classifier using a grid partition of feature space, compared with the classifier of logistic regression [13]. However, the grid partition method often results in too many fuzzy rules for human to handle. And the adopted rule filtering strategies often cause the loss of accuracy.

### 1.2.2 Genetic Network Inference Problems

The goal of constructing genetic network models is to reveal the regulation rules behind the gene expression data. The genetic network may be used as instructions for further

biological experiments to discover more delicate and substantial functions in molecular biology, biochemistry, bioengineering, and pharmaceuticals. The traditional biological experiments mainly concentrate on small-scale or local reaction among parts of complex biological system behavior. When faced with large-scale genetic networks, the efficient method with increased computational efficiency is desirable.

Most of the mathematical algorithms and models proposed to describe biochemical networks include [22]: Boolean network model [23], Bayesian network [24, 25], and differential model or S-system model [26]. In Boolean network models, gene expression levels can be referred to two situations, true or false. These models have the advantage that they can be solved with less computing effort. But the drawback is that they can't quantify interaction intensity between genes and not adequate in analyzing cyclic network structure such as feedback regulatory loops. Bayesian network model is able to deal with linear, non-linear, and combinatorial problems also used to infer genetic networks. But similar to Boolean networks, it suffers from the same dilemma and only applicable to acyclic structures [22, 24]. To cope with the cyclic networks, some authors adopted the adapted dynamic Bayesian network [27, 28].

Another frequently used approach is to use differential equation models for analysis of gene expression. The most popular model can be referred to the S-system model which has been considered suitable to characterize biochemical network systems and capable to analyze the regulatory system dynamics [29, 30, 31, 32, 33, 34, 35, 26]. The S-system model is a set of non-linear differential equations as the following form:

$$\frac{dX_i(t)}{dt} = \alpha_i \prod_{j=1}^N X_j^{g_{ij}}(t-1) - \beta_i \prod_{j=1}^N X_j^{h_{ij}}(t-1) \quad (1.1)$$

where  $X_i(t)$  represents the expression level of gene  $i$  at time  $t$  and  $N$  is the number of genes in a genetic network.  $\alpha_i$  and  $\beta_i$  are rate constants which indicate the direction of mass flow and must be positive.  $g_{ij}$  and  $h_{ij}$  are kinetic orders which reflect the intensity of interaction from gene  $j$  to  $i$ . For inferring an S-system model, it is necessary to estimate all the  $2N(N+1)$  S-system parameters  $(\alpha_i, \beta_i, g_{ij}, h_{ij})$  from experimental time-series data of gene expression. Essentially, this reverse engineering problem is a large-scale parameter optimization problem (LPOP) which is time-consuming and intractable.

Genetic algorithm (GA) [36] plays an important role in solving the optimization problem of dynamic modeling of genetic networks using the S-system model [29, 30, 31, 33].

Kikuchi et al. used GA with simplex crossover (SPXGA) to improve the optimization ability for dynamic modeling of genetic networks from  $N = 2$  to 5 [29]. SPXGA successfully inferred the dynamics of a small genetic network using only time-series data of gene expression. When deal with a more complicated structure with a large number of genes (i.e.,  $N = 10$ ), it is hard to obtain a satisfactory solution in a limited amount of computation time. To infer large-scale genetic network models, Maki et al. proposed an efficient problem decomposition strategy to divide the inference problem into  $N$  separated small subproblems [32]. To reduce search time of the inference problem, Voit and Almeida proposed an approach to transforming the problem into several sets of decoupled algebraic equations, which can be processed efficiently in parallel or sequentially [26]. Kimura et al. used a cooperative coevolutionary algorithm with the problem decomposition strategy to efficiently infer large-scale S-system models with noisy time-series data [31]. However, the existing efficient evolutionary algorithms required parallel computing on a PC cluster for efficiently obtaining satisfactory solutions [29, 30, 31].

## 1.3 Sketch of the Thesis

### 1.3.1 An Interpretable Gene Expression Classifier for Gene Expression Classification

In this study, we propose an interpretable gene expression classifier (named iGEC) with an accurate and compact fuzzy rule base using a scatter partition of feature space for microarray data analysis. Because gene expression data have the property of natural clustering, fuzzy classifiers using a scatter partition of feature spaces often have a smaller number of rules than those using grid partition [37]. The design of iGEC has three objectives to be simultaneously optimized: maximal classification accuracy, minimal number of rules, and minimal number of used features. In designing iGEC, the flexible membership function optimization, rule filtering, and gene selection strategies are simultaneously optimized. A novel intelligent genetic algorithm (IGA) is used to efficiently solve the design problem with a large number of tuning parameters [38]. It is noted that the similar fuzzy

rule-based classifier to iGEC is averagely better than the C4.5 classifier using 11 machine learning data sets in terms of classification accuracy, rule number, and used feature number [37]. The performance of iGEC is evaluated using eight gene expression data sets. It is shown that iGEC has an accurate, concise, and interpretable rule base (1.13 rules per class averagely) in terms of averaged classification accuracy (87.89%), rule number (3.91), and used gene number (4.97). Moreover, iGEC not only has better performance than Vinterbo’s classifier in terms of the above-mentioned objectives, but also is more accurate than some non-rule-based classifiers using a large number of genes. Further, the proposed iGEC can be extended to an interpretable scoring fuzzy classifier (iSFC) which can effectively quantify the certainty grades of samples belonging to each class.

### 1.3.2 An Intelligent Two-stage Evolutionary Algorithm for Inference of Genetic Network

For the genetic network inference problems, we propose an intelligent two-stage evolutionary algorithm (iTEA) to infer S-system models of large-scale genetic networks from small-noise gene expression data using a single-CPU PC. iTEA consists of two stages where each uses a divide-and-conquer strategy. We solve the optimization problem by decomposing it into  $N$  subproblems having  $2(N+1)$  parameters each when the measurement noise is small. In stage 1, each subproblem is solved using the novel intelligent genetic algorithm (IGA) based on orthogonal experimental design (OED). In stage 2, the obtained  $N$  solutions to the  $N$  subproblems are combined and refined using a novel OED-based orthogonal simulated annealing algorithm (OSA) for handling noisy gene expression data. The effectiveness of iTEA is evaluated using simulated expression patterns with/without noise. It will be shown that 1) IGA is efficient enough to solve subproblems; 2) IGA is significantly superior to the existing method SPXGA [29]; and 3) iTEA performs well in inferring S-system models of large-scale genetic networks from small-noise gene expression data.

## 1.4 Organization

This monograph is divided into three parts. The first part (Chapter 3) is devoted to the intelligent genetic algorithm (IGA). The second part (Chapter 4) devoted to using IGA to design an interpretable fuzzy rule-base classifier for microarray gene expression classification. The third part (Chapter 5) is devoted to an intelligent two-stage evolutionary algorithm (iTEA) to solve genetic network inference problem. Finally, the detail organization is as follows.

Chapter 2 contains the introductions of several common used classifiers for gene expression classification, four kinds of genetic network models for describing genetic networks, and finally, the genetic algorithm which is one of the evolutionary algorithm is presented.

Chapter 3 presents the novel efficient intelligent genetic algorithm (IGA) using the efficient divide-and-conquer strategy and being good at solving the large-scale parameter optimization problem (LPOP) based on orthogonal experimental design (OED) and factor analysis.

Chapter 4 contains two major parts. One is how the designing problem of an interpretable gene expression classifier (iGEC) with accurate and compact fuzzy rule base to be transformed into an LPOP. The other is how to use IGA to optimize the design problem. Finally, the experimental results of iGEC on eight benchmark data sets and conclusions for iGEC are presented.

Chapter 5 proposes an intelligent two-stage evolutionary algorithm (iTEA) to optimize genetic network inference problem. The variant of intelligent genetic algorithm and another novel OED-based simulated annealing algorithm (OSA) used in each stages, and the combination of these two algorithms for iTEA are introduced in this chapter. In the last part of this chapter are the experimental results and conclusions for iTEA.

Chapter 6 concludes the thesis. It starts with the summary of the goals and the importances of gene expression classification and genetic network inference problems in microarray data analysis. Following are the results and future works of our two proposed optimization methods for the two topics mentioned above.

# Chapter 2

## Background

### 2.1 Classifiers for Gene Expression Classification

Several common machine learning methods, such as neural network,  $k$ -nearest-neighbor rule, support vector machine, and fuzzy rule-based classifier, have been used for gene expression classification. Each method has its own characteristics. Following are the brief introductions of these machine learning methods mentioned above.

#### 2.1.1 Neural Networks (NN)

Imitating the biological nervous systems, such as the brain, the neural network is a way of information processing or classification method which is inspired from the way of information processing of the neuron in biological nervous systems. Neural network is composed of a large number of highly interconnected processing nodes (neurons) working in unison to solve specific classification problems and there exists a weight value for a certain simple calculation in each link between two nodes. Following are the two common used variations of neural networks: backpropagation neural networks (BNN) and probabilistic neural networks (PNN).

##### **Backpropagation Neural Networks (BNN)**

Because of the easiness and effectiveness of their learning strategy, backpropagation neural networks (BNN) are one of the most common neural network structures and have been used in a wide range of machine learning applications, such as gene expression data classification problem.

The structure of the BNN is a network of nodes arranged in three layers—the input, hidden, and output layers. The input and output layers serve as nodes to buffer input and output for the model, respectively, and the hidden layer serves to provide a means for input relations to be represented in the output.

When presented with an input pattern, each input node takes the value of the corresponding attribute in the input pattern. During the training phase of the network, once a classification has been given, it is compared to the actual classification. This is then “backpropagated” through the network, which causes the hidden and output layer nodes to adjust their weights in response to any error in classification, if it occurs. The advantages and limitations of BNN are: 1) BNN performs well in prediction and classification; 2) Although, BNN is slow compared to other machine learning methods, such as support vector machines, it is reasonable for neural network; 3) The learning results are lack of explanation of what has been learned. [12, 39] applied this method to the gene expression data classification problem.

### **Probabilistic Neural Networks (PNN)**

Probabilistic neural networks (PNN) can be used for classification problems. Rather than the BNN directly fitting the training samples, PNN is interpreted as a function which approximates the probability density function (*pdf*) of the underlying training samples’ distribution.

During the test phase, when a sample forms an input vector is presented, the first layer computes distances from the input vector to the training input vectors, and produces a vector whose elements indicate how close the input is to a training input. The second layer (or pattern layer) sums these contributions for each class of inputs to produce as its net output a vector of probabilities. Finally, a compete transfer function on the output of the second layer picks the maximum of these probabilities, and makes the corresponding classified decision.

Not only because that PNN identifies the commonalities in the training examples and allows to perform classification of unseen samples, but also the learning rule of PNN is simple and requires only a single pass through the training data. The PNN offers the

following advantages [40]: 1) rapid training speed: the PNN is much faster than backpropagation; 2) guaranteed convergence to a Bayes classifier if enough training examples are provided, that is it approaches Bayes optimality; 3) enabling incremental training which is fast, that is additionally provided training examples can be incorporated without difficulties; 4) robustness to noisy examples. [12, 41, 42, 43] applied PNN to classify microarray samples.

### **2.1.2 $k$ -nearest-neighbor Rule ( $k$ -NN)**

The  $k$ -nearest-neighbor ( $k$ -NN) rule represents one of the most widely used classifiers in pattern recognition. The  $k$ -NN rule is based on the nearest neighbor algorithm which is a simple classification algorithm; a query data is classified according to the classification of the nearest neighbor from a database of known classifications, i.e. a reference dataset. By means of generalization the nearest neighbor algorithm, we obtained the so-called  $k$ -nearest neighbor algorithm, where the  $k$ -nearest samples are selected and the query data is assigned the class most frequently represented among them. A further extension is to weight the  $k$ -nearest samples with a certain power of the distance from the query data.

Although it is simple,  $k$ -NN can give competitive performance compared to many other methods. There are some applications of the nearest neighbor methods on bioinformatics, such as to predict protein secondary structure and to classify biological and medical data [44]. However, because of the small number of microarray samples, the  $k$ -nearest neighbor method often leads to the problem of overfitting and performs not very well on microarray data analysis. [12, 45] used the  $k$ -NN rule for gene expression classification.

### **2.1.3 Support Vector Machine (SVM)**

The support vector machines (SVM) are learning machine based on the statistical theory proposed by Vapnik [46]. Not like the most of machine learning methods which minimize the classification error, the objective of SVM is to maximize the upper bound of the error rate under a certain probability such that SVM can make the classification precisely.

The main idea of the SVM is that: given a set of training data samples under a non-linearly separable low-dimension. The SVM non-linearly maps their low dimensional



input space into a high dimensional feature space. In this high dimensional feature space, SVM finds a linear hyperplane and use this to make the classification. The corresponding classification using the optimal hyperplane has the two properties: 1) leaving the largest possible part of training samples of the same class on the same side; 2) maximizing the distance of the each class from this hyper plane. If SVM can find the optimal linearly separable hyperplane which minimize the probability of misclassifying the training samples, the unseen test samples will be well classified, too. Figure 2.1 demonstrates the illumination of the learning principle and testing behavior of SVM. In the left diagram with small separating margin, the unknown test sample,  $x$ , would be classified to class 2, however, according to the distances between sample  $x$  and each class, sample  $x$  should belong to class 1. Therefore, the right diagram with large separating margin will provides better test accuracy when coping with the unknown test samples.

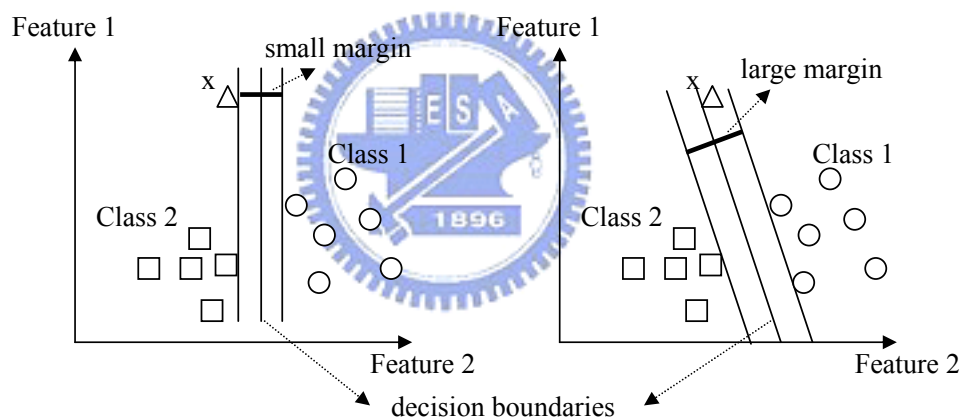


Figure 2.1. Illumination of the learning principle and testing behavior of SVM.

The SVM is the most popularly used on the microarray data analysis [12, 45, 47, 48, 49]. Because of the characteristic that it is not easily to be overfitting when the number of training samples is small, the SVM performs well on the gene expression data classification problem.

#### 2.1.4 Fuzzy Rule-Based Classifier

There are many machine learning techniques, such as support vector machine (SVM), neural networks (NN),  $k$ -nearest neighbor ( $k$ -NN), and logistic regression have been used

in gene expression data classification [12, 13]. However, the learning results of the above-mentioned classifiers cannot be summarized into human-interpretable forms for biologists and biomedical scientists [13]. Fuzzy Rule-Base Classifier not only provides the human-interpretable if-then learning rules but also adopts the “fuzzy” concept to describe the continuous feature value rather than “crisp” one.

The form of a fuzzy if-then rule is:

$$R : \text{If } \dots \text{ then Class is } \dots$$

The most distinguishing property of fuzzy logic is that it deals with fuzzy propositions, that is, propositions which contain fuzzy variables and fuzzy values, for example, “the gene  $X_i$  is up-regulated.” or “the gene  $X_j$  is down-regulated.”. However, the truth values for fuzzy propositions are not binary value, i.e. TRUE/FALSE only, as is the case in propositional boolean logic, but include all the possibilities of certainty grade between two extreme values.

In fuzzy systems, there are three major fuzzy partition methods for the feature space of a membership functions: grid partition, tree partition and scatter partition. In Ho et al. 's work [37], they are briefly described as follows. Figure 2.2 is the brief illuminations of the three partition method.

### **Grid Partition**

Grid partition is the most commonly used fuzzy partition approach. There may be  $p^n$  fuzzy rules in the case of  $p$  fuzzy sets on each axis of an  $n - D$  feature space using grid partition. A major advantage of grid partition is that fuzzy rules obtained from fixed linguistic fuzzy grids are always linguistically interpretable. However, the grid partition method often results in too many fuzzy rules for human to handle. And the adopted rule filtering strategies often cause the loss of accuracy.

### **Tree partition**

Tree partition results from a series of guillotine cuts. A guillotine cut is made entirely across the subspace to be partitioned, and each of the regions thus produced can then

be subjected to independent guillotine cutting. Tree partition can significantly relieve the problem of rule explosion and accelerate classification, but its application to high-dimensional problems faces practical problems [50].

### Scatter Partition

Scatter partition uses multi-dimensional antecedent fuzzy sets. From the viewpoint of classification performance, scatter partition may be the most effective approach to designing high-dimensional fuzzy classifiers [51]. Scatter partition usually generates fewer fuzzy regions than the grid and tree partitions owing to the natural clustering property of training patterns. However, scatter partition of high-dimensional feature spaces is difficult, and thus some learning or automatic evolutionary procedures become necessary [50].

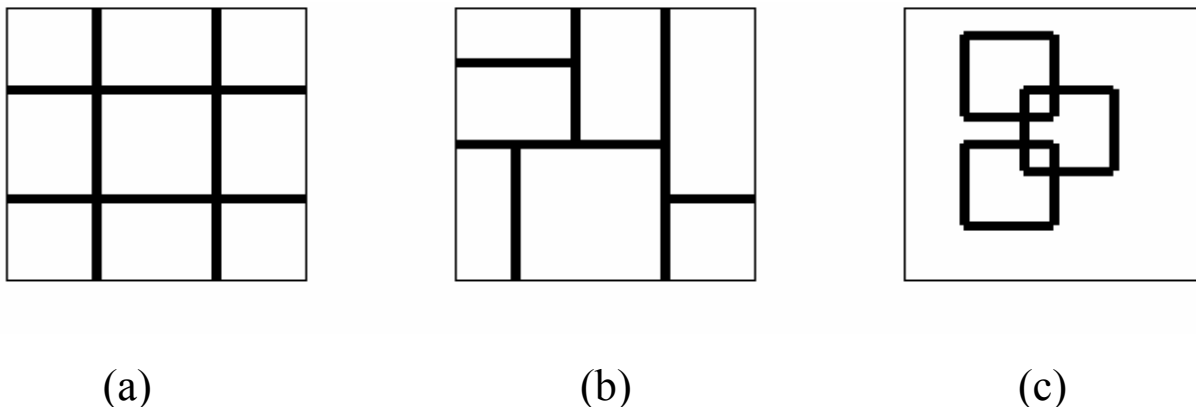


Figure 2.2. Illustrations of the three fuzzy partition methods: (a) grid partition; (b) tree partition; (c) scatter partition.

Each fuzzy partition method forms a corresponding membership functions representing the fuzzy concept mentioned above. Recently, Vinterbo et al. proposed a small and interpretable fuzzy rule-based classifier using a grid partition of feature space for gene expression classification [13]. Because of the continuous and noisy gene expression data of microarray experiments, the fuzzy classifier often makes the classification result more precisely.

## 2.2 Models of Genetic Network

The first issue of the inference problem of genetic network is which genetic work model is going to be adopted and to describe the interaction among genes. Based on the nature of the regulation interaction, reverse engineering algorithms for genetic network modeling, in general, can be classified into three categories: by Boolean rules, by stochastic formulas and theory, and by differential equations. In Wu et al. 's work [52], they summarized the most frequently used models among the four major categories mentioned above: 1) boolean network model, 2) Bayesian network model, 3) linear differential network model, and 4) S-system network model. Following are the brief descriptions about these four models.

### 2.2.1 Boolean Network Model

Boolean network model is the simplest and the most computationally effective model system that can give some insight into the overall behavior of large genetic networks [53]. In Boolean network models, the interaction between genes can be referred to two situations, true or false (on or off) and the state is determined by a Boolean function of the states of some other genes. These simple models have the advantage that they can be solved with less computing effort. But the drawback is that they can't quantify interaction intensity between genes and not adequate in analyzing cyclic network structure such as feedback regulatory loops. [54, 55, 56] adopted the Boolean network model to describe the regulation of genetic network.

### 2.2.2 Bayesian Network Model

Bayesian network model which is able to deal with linear, non-linear, and combinatorial problems is also used to describe genetic networks. Rather than the only two extreme states adopted in Boolean network model, Bayesian network use the stochastic method to model the causality between genes which can quantify degrees of the interactions among networks. There are a set of nodes and a set of edges, which together constitute a directed acyclic graph in a bayesian network. The nodes in the graph represent random variables,

while the edges indicate the existence of direct causal connections between the linked nodes and the strengths of these connections are expressed in terms of conditional probabilities. But similar to Boolean networks, it suffers from the same dilemma and only applicable to acyclic structures [22, 24]. To cope with the cyclic networks and dynamic modeling the gene regulation, some authors adopted the adapted dynamic Bayesian network [27, 28].

### 2.2.3 Linear Differential Network Model

The linear differential model is one of the simplest ways to dynamically model the interactions between genes. Linear differential models assume that the change of each gene at one time point is determined by a weighted sum of the expressions of all genes at the previous one time point. The mathematical formulism of the linear differential model for a continuous-time system with  $N$  genes is described as follows [23]:

$$\frac{dX_i(t)}{dt} = \sum_{j=1}^N w_{i,j} \times X_j(t-1) + b_i, i = 1, \dots, N, \quad (2.1)$$

where  $X_i(t)$  is the expression level of the  $i_{th}$  gene at time  $t$ ,  $N$  indicates the number of genes in this genetic network, and  $b_i$  is a bias term indicating whether gene  $i$  is expressed or not in the absence of regulatory inputs.

The linear differential model is very simple such that it can provide the chance to researchers for finding out the most significant information without taking too complex computational cost. However, there is a major drawback when using the linear differential model: the assumption of linear gene-regulation relationship is unrealistic. To cope with the nonlinear complex systems, such as gene expression networks and metabolic pathways, we need a more general, non-linear, and representative model. [57, 58] applied the linear differential equations to modeling the gene regulation relationship.

### 2.2.4 S-system Network Model

Another frequently used approach is to use non-linear differential equation models for analysis of gene expression. The most popular model can be referred to the S-system model which has been considered suitable to characterize biochemical network systems and capable to analyze the regulatory system dynamics [29, 30, 31, 32, 33, 34, 35, 26].

The S-system model is a set of non-linear differential equations as the following form:

$$\frac{dX_i(t)}{dt} = \alpha_i \prod_{j=1}^N X_j^{g_{ij}}(t-1) - \beta_i \prod_{j=1}^N X_j^{h_{ij}}(t-1) \quad (2.2)$$

where  $X_i(t)$  represents the expression level of gene  $i$  at time  $t$  and  $N$  is the number of genes in a genetic network.  $\alpha_i$  and  $\beta_i$  are rate constants which indicate the direction of mass flow and must be positive.  $g_{ij}$  and  $h_{ij}$  are kinetic orders which reflect the intensity of interaction from gene  $j$  to  $i$ . For inferring an S-system model, it is necessary to estimate all the  $2N(N+1)$  S-system parameters  $(\alpha_i, \beta_i, g_{ij}, h_{ij})$  from experimental time-series data of gene expression.

The S-system models have the ability not only to describe a non-linear gene regulation system but also to cope with the cyclic networks and the dynamic regulation between genes. However, the reverse engineering problem of this general and representative model is a large-scale parameter optimization problem with  $2N(N+1)$  parameters which is time-consuming and intractable. Genetic algorithm (GA) [36] plays an important role in solving the optimization problem of dynamic modeling of genetic networks using the S-system model [29, 30, 31, 33].

## 2.3 Genetic Algorithm (GA)

Recently, genetic algorithm (GA) proposed by J. H. Holland in 1970 has become the one of the most popular optimization methods [36]. GA has the advantages that it provides the robust solution quality and that although GA does not need the additional domain knowledge to search the solution space, however, applying appropriate prior knowledge leads to better performance. The main difference between GA and traditional numerical methods is that: 1) GA adopts the coding strategy to transform the candidate solution to “individual chromosome” consisting of a group of parameters; 2) with the population of chromosomes and the specific operators to exchange the information between chromosomes during searching, GA can efficiently search for the optimal solutions in the search space with high probability to finding out the global optima. Figure 2.3 shows the illuminations of the searching models of GA and traditional numerical methods. The qualities

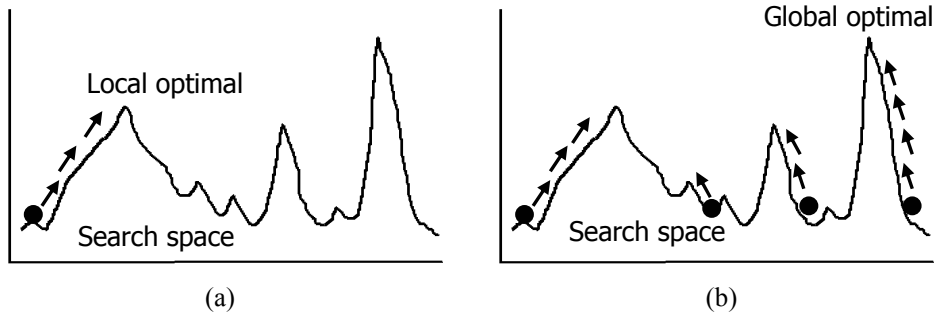


Figure 2.3. Illuminations of the searching behaviors between genetic algorithm and traditional numerical method; (a) traditional numerical method; (b) genetic algorithm.

of the solutions using traditional numerical methods highly depends on the initial given value such that it is easy to fall into local optima.

GA consists of three basic operators: 1) Selection: attempting to apply pressure upon the population in a manner similar to that of natural selection found in biological systems; 2) Crossover: allowing solutions to exchange information in a way similar to that used by natural organism undergoing sexual reproduction; 3) Mutation: used to randomly change (flip) the value of single parameter (bit) within the individual chromosome. Figure 2.4 is the flowchart of GA. In this flowchart, the lighter the color of gene is, the better the value of the gene contains. Following are the brief introductions about the major issues in GA: encoding scheme and fitness function, population initialization, selection, crossover, mutation, and termination condition.

### 2.3.1 Encoding Scheme and Fitness Function

The first stage of building a genetic algorithm is to decide on a genetic representation of a candidate solution to the original problem. This involves defining and arranging each parameter within the individual chromosome and the mapping approach from individual chromosomes and the corresponding candidate solutions to problems being solved.

After deciding on the representation of chromosomes is to design an appropriate fitness function. The fitness functions (or objective functions) are used to quantify each candidate solution mapped from one chromosome, and often, they can be maximized or minimized. Because of the selection operator based on the fitness values a lot, the performance of genetic algorithms usually highly depends on the convenience of the adopted

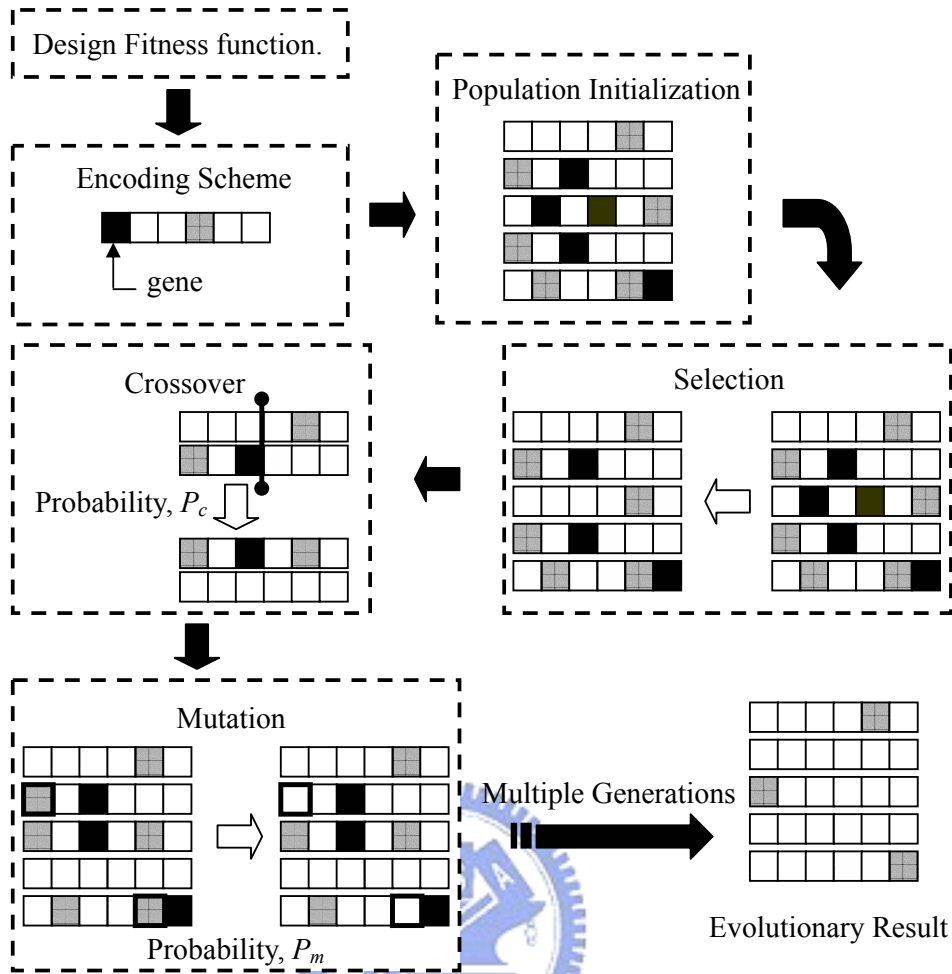


Figure 2.4. The flowchart of genetic algorithm. The lighter the color of gene is, the better the value of the gene contains.

fitness functions.

Following is a simple example for encoding scheme and fitness function design. If we want to maximize the following equation  $f(x)$ :

$$f(x) = x^2; \text{ for integer } x \text{ and } 0 \leq x \leq 4095. \quad (2.3)$$

We can just use  $f(x)$  as the fitness function to be maximized, and adopt the binary representation strategy to encode the value of  $x$  such that “110101100100” implies  $x = 3428$  while “010100001100” represents  $x = 1292$ .

### 2.3.2 Population Initialization

One of the characteristics of genetic algorithms is doing parallel search in the solution space with a set of candidate solutions. This set of candidate solutions is called a “popu-



lation”. To achieve the objective of searching the solution space globally, the chromosomes of populations usually are randomly initialized such that each chromosome will be scattered over the solution space uniformly. However, if there are constraints on solutions to the problem being solved, how to guarantee all initial chromosomes feasible is an important issue to be considered.

### 2.3.3 Selection

Selection attempts to apply pressure upon the population in a manner similar to that of natural selection found in biological systems. Poorer performing individuals are weeded out and better (fitter) performing ones have a greater chance of promoting the information they contain within the next generation. The typical selection operators can be classified to two categories, parent selection and survivor selection. Both of them are to distinguish among individuals based on their qualities, however parent selection is responsible to allow the fitter individuals to become parents of the next generation, while survivor selection is called after having created the offspring of the selected parents and decide which individual will exist in the next generation. Because of the selection operator, GA can guarantee that after iterated generations, the average quality of the entire population will be improved with a high probability. Following, we will introduce the most common used methods for parent selection: roulette wheel selection and binary tournament selection, and for survivor selection: ranking selection.

#### Roulette Wheel Selection

With this approach, the probability of selection for one individual is based on the proportion of its fitness to the sum of fitness of entire population. Given the fitness value of the  $i_{th}$  individual,  $f_i$ , and the size of population is  $N_{pop}$ , the probability of the  $i_{th}$  individual being selected is:

$$p_i = \frac{f_i}{\sum_{j=1}^{N_{pop}} f_j}. \quad (2.4)$$

Suppose that there are four individual in the population, and their fitness values and the corresponding selected probabilities are showed in Figure 2.5. For example, in selection, first, randomly generate a real number in  $[0, 1]$ . If the real number is in  $[0, 0.1]$

then child 1,  $C_1$  is selected; if the random value is in  $(0.1, 0.3]$  then child 2,  $C_2$  is selected. Repeat the steps mentioned above, until the number of individuals in the mating pool is equal to the size of population in the previous generation.

Individuals	$C_1$	$C_2$	$C_3$	$C_4$
Fitness	10	20	30	40
Selected Probability	0.1	0.2	0.3	0.4

(a)

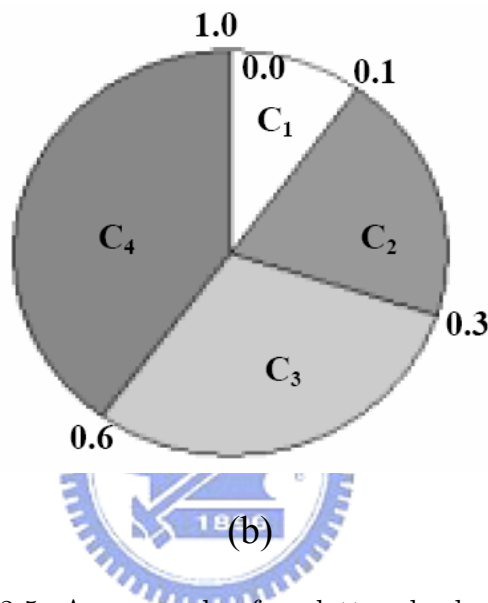


Figure 2.5. An example of roulette wheel selection.

### Binary Tournament Selection

The main idea of binary tournament selection is that when doing parent selection, repeat to randomly picking up two individual and place the fitter one to the mating pool, until the number of individuals in the mating pool is equal to the size of population in the previous generation. Compared with roulette wheel selection, in the later period of evolutionary computing, binary tournament selection has a better ability to distinguish the fitter one from two individuals. That is because that in the later period of evolutionary computing, the fitness values of all individual in the population converged such that because of the closed fitness values between individuals, when using roulette wheel selection, it is more difficult to distinguish the better one by two almost the same probabilities. However, even though the fitness values of population have converged, by means of judging which

one of two has the better fitness, binary tournament selection can still select the better one successfully.

### Ranking Selection

Ranking selection is the simplest approach to survivor selection. Rank selection method replaces the worst  $P_s \times N_{pop}$  individuals with the best  $P_s \times N_{pop}$  individuals to form a new population, where  $P_s$  is a selection probability and  $N_{pop}$  is the size of population. Although it is simple, ranking selection have the advantage that it can efficiently speed up the convergence of the entire population and improve the average quality of entire population a lot. [38] adopted ranking selection in their selection operator of GA.

### 2.3.4 Crossover

The major advantage of genetic algorithm is that with the population of chromosomes (candidate solutions) and the specific operators, crossover, each individual in the population can efficiently searching the solution space concurrently. As the name indicate, crossover or recombination allowing two parent individuals to exchange their parameters or information in a way similar to that used by natural organism undergoing sexual reproduction. With a probabilistic parameter,  $P_c$ , controlling whether the selected pairs of individuals doing crossover or not, we can mate two individuals with different but desirable features to produce the offspring that combines both of those features. Cooperating with the selection operator, once the better or fitter offspring are generated, they have the higher probability to survive after selection such that the average fitness of population is successfully improved. The most used variations of crossover operator are: one-point crossover, multi-point crossover, and uniform crossover.

#### One-point Crossover

Before doing one-point crossover, Randomly generate a cut point, then exchange the all parameters of the two parent behind the position of cut point. Figure 2.6(a) shows the behavior of one-point crossover.

## Multi-point Crossover

First, randomly generate multiple cut points. After the positions of multiple cut points are determined, randomly determine whether the parameters of parents between all pairs of successive cut points to be exchange or not. Figure 2.6(b) shows the behavior of multi-point crossover.

## Uniform Crossover

Before doing uniform crossover, randomly generate a binary bit string with length being the same as the number of parameters in the individual chromosome. This binary bit string is used as a mask. If a bit value is one, it means that the corresponding parameter should be exchanged, while zero bit implies that the corresponding parameters will not to be exchanged. Figure 2.6(c) shows the behavior of uniform crossover.

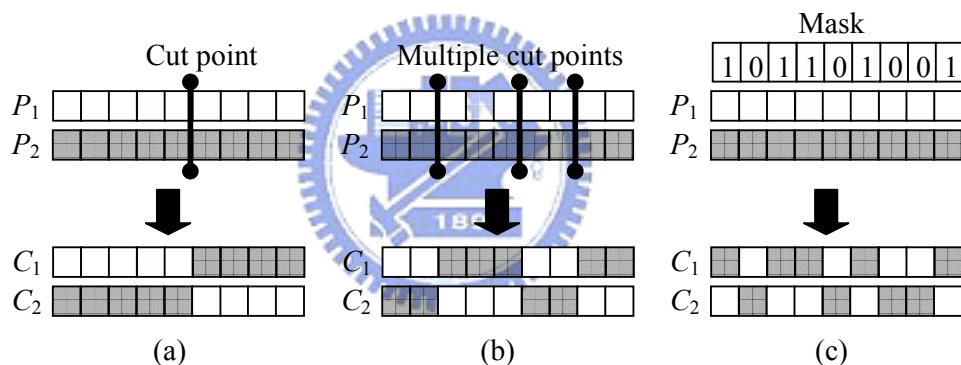


Figure 2.6. Illustrations of (a) one-point crossover, (b) multi-point crossover, and (c) uniform crossover.

## 2.3.5 Mutation

Mutation operators randomly change (flip) the value of single parameter (bit) within individual chromosome. When doing mutation operation, each parameter or bit in a single individual chromosome is determined whether its value is changed or not based on a probabilistic parameter  $P_m$ . Because of the experiences of medical science, usually, the mutation brings harmful effects to individuals such that we often set  $P_m$  with a small value. However, the mutation operators still have the significant importance during evolutionary computing. According to the selection and crossover operator, the average

quality of population will be improved during iterated generations. However, in the last period of evolutionary computing, the fitness values among populations converge and all information contained in the individuals is almost the same. Without producing some new information or parameter values, the entire candidate solutions of population will be trapped into local optima. In this situation, the mutation operator can bring the new information to the entire population such that the population may jump the local optima and find out the global ones.

The mostly used methods of mutation operations are bit flip mutation for binary bit string or randomly generating the perturbing value for each real-valued parameter. The bit flip mutation for binary bit string is that when doing mutation, each bit in the individual have the probability  $P_m$  to flip its value, such as change 1 to 0 or reverse 0 to 1. Figure 2.7 shows the behavior of bit flip mutation. The other commonly used mutation for real-valued parameters is described below. With a probability  $P_m$ , assume a real-valued parameter  $x$  is to be mutated. A perturbation  $x'$  of  $x$  is generated by the Cauchy-Lorentz probability distribution [59]. The mutated value of  $x$  is  $x + x'$  or  $x - x'$ , determined randomly.

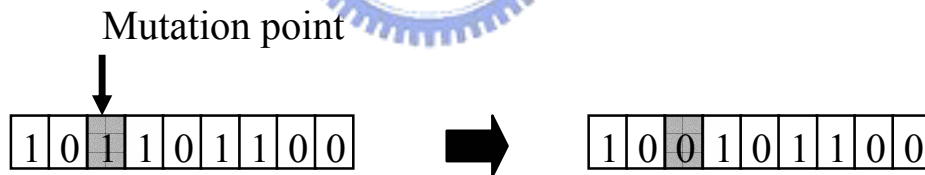


Figure 2.7. An example of bit flip mutation.

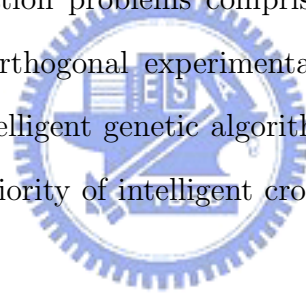
### 2.3.6 Termination Condition

The termination conditions are the criterions that we terminate the evolutionary search or computing of genetic algorithm. The commonly used termination conditions may be: 1) the average or best fitness values is improved to a default value; 2) The number of generations or fitness evaluation is up to a upper bound set in advance; 3) The best fitness is still not improved after a number of generations; 4) other criterions designed by the users.

# Chapter 3

## Intelligent Genetic Algorithm

The used intelligent genetic algorithm (IGA) is a specific variant of the intelligent evolutionary algorithm [38] to solve the large-scale parameter optimization problems (LPOP). The main difference between IGA and the traditional GA [36] is an efficient intelligent crossover operation. The intelligent crossover is based on orthogonal experimental design to solve intractable optimization problems comprising lots of design parameters. The following sections describe orthogonal experimental design, factor analysis, intelligent crossover, and the simple intelligent genetic algorithm. The merits of orthogonal experimental design and the superiority of intelligent crossover can be further referred to [37] and [38].



### 3.1 Concept of Orthogonal Experimental Design (OED)

An efficient way to study the effect of several factors simultaneously is to use OED with both orthogonal array (OA) and factor analysis [60, 61, 62]. The factors are the variables (parameters), which affect response variables, and a setting (or a discriminative value) of a factor is regarded as a level of the factor. OED utilizes properties of fractional factorial experiments to efficiently determine the best combination of factor levels to use in design problems.

OA is a fractional factorial array, which assures a balanced comparison of levels of any factor. OA is an array of numbers arranged in rows and columns where each row represents the levels of factors in each combination, and each column represents a specific factor that can be changed from each combination. The term “main effect” designates

the effect on response variables that one can trace to a design parameter [62]. The array is called orthogonal because all columns can be evaluated independently of one another, and the main effect of one factor does not bother the estimation of the main effect of another factor. Factor analysis using the orthogonal array's tabulation of experimental results can evaluate the effects of individual factors on the evaluation function, rank the most effective factors, and determine the best level for each factor such that the evaluation function is optimized.

OED can provide near-optimal quality characteristics for a specific objective. Furthermore, there is a large saving in the experimental effort. OED specifies the procedure of drawing a representative sample of experiments with the intention of reaching a sound decision [62]. Therefore, OED using OA and factor analysis is regarded as a systematic reasoning method.

## 3.2 Orthogonal Array

In this study, the two-level and three-level OAs are used for IGA and OSA [63], respectively. The two-level OAs used in IGA are described below. Let there be  $\alpha$  factors, with two levels each. The total number of level combinations is  $2^\alpha$  for a complete factorial experiment. To use an OA of  $\alpha$  factors, we obtain an integer  $M = 2^{\lceil \log_2(\alpha+1) \rceil}$  where the bracket represents an upper ceiling operation, build an OA  $L_M(2^{M-1})$  with  $M$  rows and  $M - 1$  columns, use the first  $\alpha$  columns, and ignore the other  $M - \alpha - 1$  columns. OA can reduce the number of level combinations for factor analysis. For instance, Table 3.1 shows an OA  $L_8(2^7)$ . The number of OA combinations required to analyze all individual factors is only  $M = O(\alpha)$ , where  $\alpha + 1 \leq M \leq 2\alpha$ .

OSA uses three-level OAs where each factor has three levels. The total number of level combinations for  $\alpha$  factors is  $3^\alpha$  for a complete factorial experiment. To use a three-level OA of  $\alpha$  factors, we obtain an integer  $M = 3^{\lceil \log_3(2\alpha+1) \rceil}$ , build an OA  $L_M(3^{(M-1)/2})$  with  $M$  rows and  $(M - 1)/2$  columns, use the first  $\alpha$  columns, and ignore the other  $(M - 1)/2 - \alpha$  columns. The number of OA combinations required to analyze all individual factors is only  $M = O(\alpha)$ , where  $2\alpha + 1 \leq M \leq 6\alpha - 3$ .

Table 3.1. An Orthogonal Array of  $L_8(2^7)$ .

Experiment no.	Factor $d$							Fitness values
	$V_1$	$V_2$	$V_3$	$V_4$	$V_5$	$V_6$	$V_7$	
1	1	1	1	1	1	1	1	$y_1$
2	1	1	1	2	2	2	2	$y_2$
3	1	2	2	1	1	2	2	$y_3$
4	1	2	2	2	2	1	1	$y_4$
5	2	1	2	1	2	1	2	$y_5$
6	2	1	2	2	1	2	1	$y_6$
7	2	2	1	1	2	2	1	$y_7$
8	2	2	1	2	1	1	2	$y_8$

Algorithm of constructing the two- and three-level OAs can be found in [63]. After proper tabulation of experimental results, the summarized data are analyzed using factor analysis to determine the relative level effects of factors.

### 3.3 Factor Analysis

Consider the OA  $L_M(2^{M-1})$  or  $L_M(3^{(M-1)/2})$  is used. Let  $y_t$  denote a function value of the combination  $t$ , where  $t = 1, \dots, M$ . Define the main effect of factor  $d$  with level  $k$  as  $S_{dk}$  where  $d = 1, \dots, \alpha$ :

$$S_{dk} = \sum_{t=1}^M y_t W_t, \quad (3.1)$$

where  $W_t = 1$  if the level of factor  $d$  of combination  $t$  is  $k$ ; otherwise,  $W_t = 0$ . Consider that the objective function is to be minimized. For the two-level OA, level 1 of factor  $d$  makes a better contribution to the objective function than level 2 of factor  $d$  does when  $S_{d1} < S_{d2}$ . If  $S_{d1} > S_{d2}$ , level 2 is better. If  $S_{d1} = S_{d2}$ , levels 1 and 2 have the same contribution. The main effect reveals the individual effect of a factor. The most effective factor  $d$  has the largest main effect difference  $MED_d = |S_{d1} - S_{d2}|$ .

For the three-level OA, the level  $k$  of factor  $d$  makes the best contribution to the objective function than the other two levels of factor  $d$  do when  $S_{dk} = \min\{S_{d1}, S_{d2}, S_{d3}\}$ . On the contrary, if the objective function is to be maximized, the level  $k$  is the best one when  $S_{dk} = \max\{S_{d1}, S_{d2}, S_{d3}\}$ . The most effective factor has the largest one of main effect differences  $MED_d = \max\{S_{d1}, S_{d2}, S_{d3}\} - \min\{S_{d1}, S_{d2}, S_{d3}\}$ . After the better one of two/three levels of each factor is determined, a reasoned combination consisting of  $\alpha$



factors with the better/best levels can be easily derived.

### 3.4 Intelligent Crossover

All parameters are encoded into a chromosome using binary codes or real values. Like traditional GAs, two parents  $P_1$  and  $P_2$  produce two children  $C_1$  and  $C_2$  in one crossover operation. Let all encoded parameters be randomly assigned into  $\alpha$  groups where each group is treated as a factor. The following steps describe the intelligent crossover operation.

Step 1: Use the first  $\alpha$  columns of an OA  $L_M(2^{M-1})$ .

Step 2: Let levels 1 and 2 of factor  $d$  represent the  $d_{th}$  groups of parameters coming from parents  $P_1$  and  $P_2$ , respectively.

Step 3: Evaluate the fitness values  $y_t$  for experiment  $t$  where  $t = 2, \dots, M$ . The value  $y_1$  is the fitness value of  $P_1$ .

Step 4: Compute the main effect  $S_{dk}$  where  $d = 1, \dots, \alpha$  and  $k = 1, 2$ .

Step 5: Determine the better one of two levels of each factor.

Step 6: The chromosome of  $C_1$  is formed using the combination of the better genes from the derived corresponding parents.

Step 7: The chromosome of  $C_2$  is formed similarly as  $C_1$ , except that the factor with the smallest main effect difference adopts the other level.

Step 8: The best two individuals among  $P_1$ ,  $P_2$ ,  $C_1$ ,  $C_2$ , and  $M - 1$  combinations of OA are used as the final children  $C_1$  and  $C_2$  for elitist strategy.

One intelligent crossover operation takes  $M + 1$  fitness evaluations, where  $\alpha + 1 \leq M \leq 2\alpha$ , to explore the search space of  $2^\alpha$  combinations.

### 3.5 The Simple Intelligent Genetic Algorithm

The used IGA is given as follows:

Step 1: Randomly generate an initial population with  $N_{pop}$  individuals.

- Step 2: Evaluate fitness values of all individuals. Let  $I_{best}$  be the best individual in the population.
- Step 3: Use the simple ranking selection that replaces the worst  $P_s \times N_{pop}$  individuals with the best  $P_s \times N_{pop}$  individuals to form a new population, where  $P_s$  is a selection probability.
- Step 4: Randomly select  $P_c \times N_{pop}$  individuals including  $I_{best}$ , where  $P_c$  is a crossover probability. Perform intelligent crossover operations for all selected pairs of parents.
- Step 5: Apply a conventional bit flip mutation for binary bit string or mutation of randomly generating the perturbing value for each real-valued parameter to the population using a mutation probability  $P_m$ . To prevent the best fitness value from deteriorating, mutation is not applied to the best individual.
- Step 6: Termination test: If a pre-specified termination condition is satisfied, stop the algorithm. Otherwise, go to Step 2.



## Chapter 4

# Interpretable Gene Expression Classifier

### 4.1 The Proposed Interpretable Gene Expression Classifier (iGEC)

This section proposes an interpretable gene expression classifier (named iGEC) with an accurate and compact fuzzy rule base using a scatter partition of feature space for microarray data analysis. The design of iGEC has three objectives to be simultaneously optimized: maximal classification accuracy, minimal number of rules, and minimal number of used genes. The novel intelligent genetic algorithm introduced in Chapter 3 is used to efficiently solve the design problem with a large number of tuning parameters.

The performance of iGEC is evaluated using eight data sets and high performance of iGEC mainly arises from two aspects. One is to simultaneously optimize all parameters in the design of iGEC where all the elements of the fuzzy classifier design have been moved in parameters of a large parameter optimization problem. The other is to use an efficient optimization algorithm IGA which uses a divide-and-conquer strategy to effectively solve these optimization problems.

#### 4.1.1 Flexible Generic Parameterized Membership Functions

The classifier design of iGEC uses flexible generic parameterized fuzzy regions which can be determined by flexible generic parameterized membership functions (FGPMFs) and a hyperbox-type fuzzy partition of feature space. Each fuzzy region corresponds to a parameterized fuzzy rule. In this study, each value of gene expression is normalized into

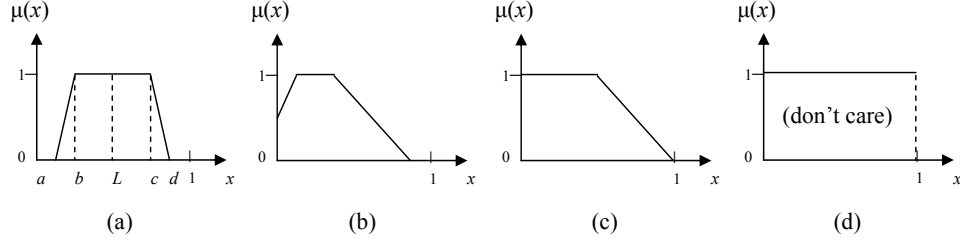


Figure 4.1. Illuminations of FGPMF. (a)  $a > 0$  and  $d < 1$ . (b)  $a < 0 < b$ . (c)  $b \leq 0$ . (d)  $b \leq 0$  and  $c \geq 1$ .

a real number in the unit interval  $[0, 1]$ . An FGPMF with a single fuzzy set is defined as:

$$\mu(x) = \begin{cases} 0 & \text{if } x \leq a \text{ or } x \geq d, \\ (x - a)/(b - a) & \text{if } a < x < b, \\ (d - x)/(d - c) & \text{if } c < x < d, \\ 1 & \text{if } b \leq x \leq c, \end{cases} \quad (4.1)$$

where  $x \in [0, 1]$  and  $a \leq b \leq c \leq d$ . The variables  $a$ ,  $b$ ,  $c$ , and  $d$  determining the shape of a trapezoidal fuzzy set are the parameters to be optimized. It is well recognized that confining genetic searches within feasible regions is often much more reliable than penalty approaches for handling constrained problems [64]. Therefore, five parameters  $V^1, V^2, \dots, V^5 \in [0, 1]$  without constraints instead of  $a$ ,  $b$ ,  $c$ , and  $d$  are encoded into a chromosome for facilitating IGA. Let an additional variable  $L = V^1$  where  $b \leq L \leq C$  which determines the location of the fuzzy set characterizing the occurrence of training patterns. When  $V^i$  are obtained, variables  $a$ ,  $b$ ,  $c$ , and  $d$  can be derived as follows:

$$\begin{aligned} a &= L - (V^2 + V^3), \\ b &= L - V^3, \\ c &= L + V^4, \\ d &= L + (V^4 + V^5). \end{aligned} \quad (4.2)$$

This transformation can always make the derived values of  $a$ ,  $b$ ,  $c$ , and  $d$  feasible and reduce interactions among encoded parameters of chromosomes. Some illuminations of FGPMF are shown in Figure 4.1.

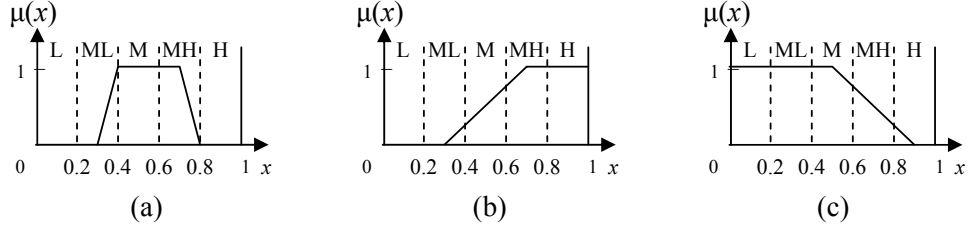


Figure 4.2. Examples of an antecedent fuzzy set  $A_{ji}$  with linguistic values (L: low, ML: medium low, M: medium, MH: medium high, H: high). (a)  $A_{ji}$  represents  $\{ML, M, MH\}$ . (b)  $A_{ji}$  represents  $\{ML, M, MH, H\}$ , i.e., not Low. (c)  $A_{ji}$  represents  $\{L, ML, M, MH, H\}$  or ALL.

#### 4.1.2 Fuzzy Rule and Fuzzy Reasoning Method

The following fuzzy if-then rules for  $n$ -dimensional pattern classification problems are used in the design of iGEC:

$$R_j : \text{If } x_1 \text{ is } A_{j1} \text{ and } \dots \text{ and } x_n \text{ is } A_{jn} \text{ then Class } CL_j \text{ with } CF_j, j = 1, \dots, N.$$

where  $R_j$  is a rule label,  $x_i$  denotes a gene variable,  $A_{ji}$  is an antecedent fuzzy set,  $C$  is a number of classes,  $CL_j \in 1, \dots, C$  denotes a consequent class label,  $CF_j$  is a certainty grade of this rule in the unit interval  $[0, 1]$ , and  $N$  is a number of initial fuzzy rules in the training phase.

To enhance interpretability of fuzzy rules, linguistic variables in fuzzy rules can be used. Each variable  $x_i$  has a linguistic set  $U = \{L, ML, M, MH, H\}$ . Each linguistic value of  $x_i$  equally represents  $1/5$  of the domain  $[0, 1]$ . Following the quantization criterion, we can consider genes to be regulated according to a qualitative level. For example,  $x_i$  is Low for down-regulated genes;  $x_i$  is Medium for neutral genes; and  $x_i$  is High for up-regulated genes. An antecedent fuzzy set  $A_{ji} \in A_u$  where  $A_u$  denotes a set of subsets of  $U$ . Examples of linguistic antecedent fuzzy sets are shown in Figure 4.2.

In the training phase, all the variables  $CL_j$  and  $CF_j$  are treated as parametric genes encoded in chromosomes and their near-optimal values are obtained using IGA. The following fuzzy reasoning method is adopted to determine the class of an input pattern  $x_p = (x_{p1}, x_{p2}, \dots, x_{pn})$  based on voting using multiple fuzzy if-then rules:

Step 1: Calculate score  $S_{Classv}(v = 1, \dots, C)$  for each class as follows:

$$\begin{aligned} S_{Classv} &= \sum_{\substack{R_j \in FC \\ CL_j = Classv}} \mu_j(x_p) \cdot CF_j, \\ \mu_j(x_p) &= \prod_{i=1}^n \mu_{ji}(x_{pi}), \end{aligned} \quad (4.3)$$

where  $FC$  denotes the fuzzy classifier, the scalar value and represents the membership function of the antecedent fuzzy set  $A_{ji}$ .

Step 2: Classify  $x_p$  as the class with a maximal value of  $S_{Classv}$ .

### 4.1.3 Fitness Function and Chromosome Representation

We define the fitness function of designing iGEC using IGA as follows:

$$\max Fit(FC) = NCP - W_r \cdot N_r - W_f \cdot N_f, \quad (4.4)$$

where  $W_r$  and  $W_f$  are positive weights. This fitness function is used to simultaneously optimize the following three objectives: to maximize the number  $NCP$  of the correctly classified training patterns and to minimize the number  $N_r$  of fuzzy rules and the number  $N_f$  of used features (genes). The weights should be specified based on the designer's preference. In this study, we used  $W_r = 0.1$  and  $W_f = 0.001$ .

A chromosome consists of control genes for selecting useful genes and significant fuzzy rules, and parametric genes for encoding the membership functions and fuzzy rules. The control genes comprise two types of parameters. One is parameter  $r_j$ ,  $j = 1, \dots, N$ , represented by one bit for eliminating unnecessary fuzzy rules. If  $r_j = 0$ , the fuzzy rule  $R_j$  is excluded from the rule base. Otherwise,  $R_j$  is included. The other is parameter  $f_i$ ,  $i = 1, \dots, n$ , represented by one bit for eliminating useless genes. If  $f_i = 0$ , the gene  $x_i$  is excluded from the classifier. Otherwise,  $x_i$  is included. The parametric genes consist of three types:

- 1)  $V_{ji}^k \in [0, 1]$ ,  $k = 1, \dots, 5$ , for determining the antecedent fuzzy set  $A_{ji}$ ;
- 2)  $CL_j$  for determining the consequent class label of rule  $R_j$ ; and
- 3)  $CF_j \in [0, 1]$  for determining the certainty grade of rule  $R_j$ ;

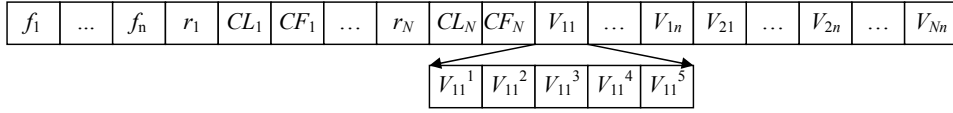


Figure 4.3. Chromosome representation.

where  $j = 1, \dots, N$  and  $i = 1, \dots, n$ . A rule base with  $N$  fuzzy rules is represented as an individual, as shown in Figure 4.3. The number of encoding parameters to be optimized is equal to  $N_p = n + 3N + 5Nn$ . A chromosome representation uses a binary string for encoding control and parametric genes. There are eight bits for encoding one of parameters  $V_{ji}^k$  and  $CF_j$ . Since each fuzzy region defines a fuzzy rule, the initial setting of  $N$  is independent of  $n$  but dependent on the number of fuzzy regions. Generally,  $N$  is set to the maximal number of possible fuzzy regions. In this study,  $N = 3C$ . The design of an efficient fuzzy classifier is formulated as a large-scale parameter optimization problem (LPOP). Once the near-optimal solution is found, an accurate classifier with a compact fuzzy rule base can be obtained.

#### 4.1.4 The Used Intelligent Genetic Algorithm to Solve the Design Problem of iGEC

Here we use the simple intelligent genetic algorithm (IGA) which is a specific variant of the intelligent evolutionary algorithm [38] to solve the design problem of iGEC. The main difference between IGA and the traditional GA [36] is an efficient intelligent crossover operation. The intelligent crossover is based on orthogonal experimental design to solve intractable optimization problems comprising lots of design parameters. The intelligent crossover is presented while the merits of orthogonal experimental design and the superiority of intelligent crossover can be further referred to [37] and [38].

##### Orthogonal Experimental Design

The two-level orthogonal arrays (OAs) used in IGA are described below. Let there be  $\alpha$  factors, with two levels each. The total number of level combinations is  $2^\alpha$  for a complete factorial experiment. To use an OA of  $\alpha$  factors, we obtain an integer  $M = 2^{\lceil \log_2(\alpha+1) \rceil}$  where the bracket represents an upper ceiling operation, build an OA  $L_M(2^{M-1})$  with

$M$  rows and  $M - 1$  columns, use the first  $\alpha$  columns, and ignore the other  $M - \alpha - 1$  columns. OA can reduce the number of level combinations for factor analysis. The number of OA combinations required to analyze all individual factors is only  $M = O(\alpha)$ , where  $\alpha + 1 \leq M \leq 2\alpha$ .

After proper tabulation of experimental results, the summarized data are analyzed using factor analysis to determine the relative effects of levels of various factors as follows. Let  $y_t$  denote a objective function value of the combination  $t$ , where  $t = 1, \dots, M$ . Define the main effect of factor  $d$  with level  $k$  as  $S_{dk}$  where  $d = 1, \dots, \alpha$ :

$$S_{dk} = \sum_{t=1}^M y_t W_t, \quad (4.5)$$

where  $W_t = 1$  if the level of factor  $d$  of combination  $t$  is  $k$ ; otherwise,  $W_t = 0$ . Consider that the objective function is to be maximized. For the two-level OA, level 1 of factor  $d$  makes a better contribution to the objective function than level 2 of factor  $d$  does when  $S_{d1} > S_{d2}$ . If  $S_{d1} < S_{d2}$ , level 2 is better. If  $S_{d1} = S_{d2}$ , levels 1 and 2 have the same contribution. The main effect reveals the individual effect of a factor. The most effective factor  $d$  has the largest main effect difference  $MED_d = |S_{d1} - S_{d2}|$ . After the better one of two levels of each factor is determined, an efficient combination consisting of all factors with the better levels can be easily derived.

### Intelligent Crossover

All parameters are encoded into a chromosome using binary codes. Like traditional GAs, two parents  $P_1$  and  $P_2$  produce two children  $C_1$  and  $C_2$  in one crossover operation. Let all encoded parameters be randomly assigned into  $\alpha$  groups where each group is treated as a factor. The following steps describe the intelligent crossover operation:

Step 1: Use the first  $\alpha$  columns of an OA  $L_M(2^{M-1})$ .

Step 2: Let levels 1 and 2 of factor  $d$  represent the  $d_{th}$  groups of parameters coming from parents  $P_1$  and  $P_2$ , respectively.

Step 3: Evaluate the fitness values  $y_t$  for experiment  $t$  where  $t = 2, \dots, M$ . The value  $y_1$  is the fitness value of  $P_1$ .

Step 4: Compute the main effect  $S_{dk}$  where  $d = 1, \dots, \alpha$  and  $k = 1, 2$ .



- Step 5: Determine the better one of two levels of each factor.
- Step 6: The chromosome of  $C_1$  is formed using the combination of the better genes from the derived corresponding parents.
- Step 7: The chromosome of  $C_2$  is formed similarly as  $C_1$ , except that the factor with the smallest main effect difference adopts the other level.
- Step 8: The best two individuals among  $P_1$ ,  $P_2$ ,  $C_1$ ,  $C_2$ , and  $M - 1$  combinations of OA are used as the final children  $C_1$  and  $C_2$  for elitist strategy.

## The Used Intelligent Genetic Algorithm

The used IGA is given as follows:

- Step 1: Randomly generate an initial population with  $N_{pop}$  individuals.
- Step 2: Evaluate fitness values of all individuals. Let  $I_{best}$  be the best individual in the population.
- Step 3: Use the simple ranking selection that replaces the worst  $P_s \times N_{pop}$  individuals with the best  $P_s \times N_{pop}$  individuals to form a new population, where  $P_s$  is a selection probability.
- Step 4: Randomly select  $P_c \times N_{pop}$  individuals including  $I_{best}$ , where  $P_c$  is a crossover probability. Perform intelligent crossover operations for all selected pairs of parents.
- Step 5: Apply a conventional bit-inverse mutation operator to the population using a mutation probability  $P_m$ . To prevent the best fitness value from deteriorating, mutation is not applied to the best individual.
- Step 6: Termination test: If a pre-specified termination condition is satisfied, stop the algorithm. Otherwise, go to Step 2.

## 4.2 Experimental Results of iGEC

### 4.2.1 Implementation and Data Sets

The parameter settings of IGA from [37] are  $N_{pop} = 20$ ,  $P_c = 0.7$ ,  $P_s = 1 - P_c$ ,  $P_m = 0.01$ , and  $\alpha = 15$ . Because the search space of optimal design of iGEC is proportional to the

number  $N_p$  of parameters to be optimized, the stopping condition is suggested to use a fixed number  $100 \times N_p$  of fitness evaluations [37] for the following two reasons: 1) for future comparisons with other methods based on the same computation cost; and 2) satisfactory solutions can be obtained which are not sensitive to the number of evaluations used. Of course, if the number of evaluations is increased, the results may be slightly improved. Because of the non-deterministic characteristic of GA, all the experimental results are the average values of 30 independent runs. For each run, a ten-fold cross validation (10-CV) is adopted. Note that the algorithm proposed by [13] is deterministic that the results are the same for all independent runs.

For comparison, we adopted the same Wilcoxon rank sum test with [13] as a non-parametric feature pre-selection method. In this study, we pre-selected  $n = 10, 15, 20$  and 100 representative genes to evaluate the performance of iGEC. Considering the test accuracy as well as the numbers of rules and genes,  $n = 15$  (slightly better) is suggested as the default setting of iGEC in this study. If the number  $C$  of classes is further increased (e.g.,  $C > 10$ ), the number  $n$  is suggested to be proportionally increased.

Table 4.1 shows the eight data sets from [12], which are available from <http://www.gems-systems.org>. The following experiments are designed to evaluate the proposed method using comparisons with some existing rule and non-rule based classifiers. The first comparison is made between iGEC and the Vinterbo's fuzzy rule-based classifier [13] and the second one between iGEC and the non-rule-based classifiers in [12].

#### **4.2.2 Experiment 1-Comparison between iGEC and the Vinterbo's Fuzzy Rule-based Classifier**

For comparisons, we conducted two evaluations on the Vinterbo's method using different numbers of pre-selected genes. One is to use 200 pre-selected genes (V200), which is the same with that in [13]. The other is to use 15 genes (V15), which is the same with that of the proposed method. Table 4.2 shows the statistical results (mean and standard deviation) of iGEC and the Vinterbo's classifier in terms of training accuracy, test accuracy, number of rules, number of genes, and rule number per class. The results of the Vinterbo's classifier were obtained by running the same program provided by Vinterbo

Table 4.1. The eight data sets from [12].

No.	Data Set	Descriptions	# of classes	# of samples	# of genes	$N_p$	Reference
1	brain tumor1	5 human brain tumor types	5	90	5920	1185	[65]
2	brain tumor2	4 malignant glioma types	4	50	10367	951	[66]
3	DLBCL	Diffuse large b-cell lymphomas and follicular lymphomas	2	77	5469	483	[67]
4	leukemia1	Acute myelogenous leukemia (AML), Acute lymphoblastic leukemia (ALL) B-cell, and ALL T-cell	3	72	5327	717	[68]
5	leukemia2	AML, ALL, and mixed-lineage leukemia (MLL)	3	72	11225	717	[69]
6	lung cancer	4 lung cancer types and normal tissues	5	203	12600	1185	[70]
7	prostate tumor	Prostate tumor and normal tissue	2	102	10509	483	[71]
8	SRBCT	Small, round blue cell tumors of childhood	4	83	2308	951	[72]

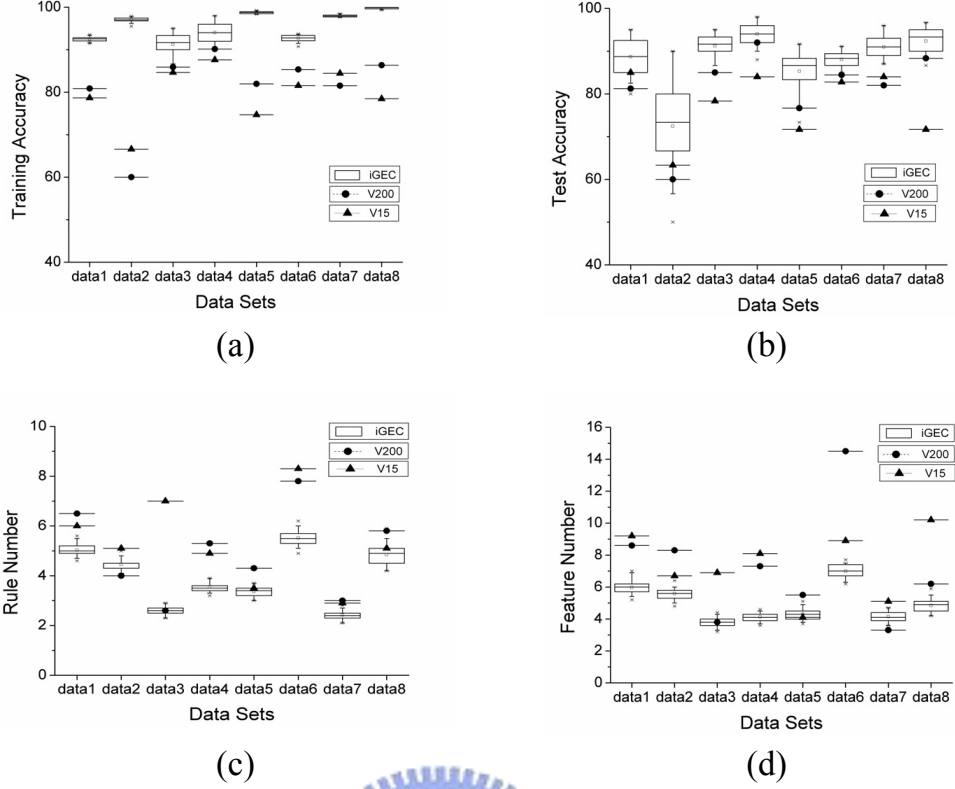


Figure 4.4. The box plots of the statistical results. (a) training accuracy, (b) test accuracy, (c) number of rules, and (d) number of used genes.

et al. [13]. The same data which have the same partition are used for iGEC, V200, and V15. Figure 4.4 presents the experimental results using box plots. Figure 4.5(a) and 4.5(b) show the three-dimensional scatter plots in terms of test accuracy, rule number, and gene number for data sets lung cancer and SRBCT, respectively.

From Table 4.2, we can observe that iGEC performs better than the Vinterbo's classifier using 200 candidate genes (V200) in the five measures:  $TrCR$  (97.1% vs. 81.5%),  $TeCR$  (87.9% vs. 81.2%),  $N_r$  (3.9 vs. 4.9),  $N_f$  (5.0 vs. 7.2), and  $N_r/C$  (1.1 vs. 1.4). Note that V200 is better than V15 but using more candidate genes and computation time. Moreover, the classifiers V200 compare favorably to those of a logistic regression model which is one of the frequently used classification method applied in the biomedical domain [13].

Figure 4.6 shows an example of iGEC using the data set leukemia1 where 90% samples are for training and the rest for test. The classifier has four fuzzy rules using three genes L05148, U46499, and U05259, where  $TrCR = 100\%$  and  $TeCR = 100\%$ . The fuzzy rules

Table 4.2. The statistical results of iGEC and the Vinterbo's classifier on training accuracy ( $TrCR$ ), test accuracy ( $TeCR$ ), number of rules ( $N_r$ ), number of genes ( $N_f$ ), and rule number per class ( $N_r/C$ ).

Data Set	Method	$TrCR(\%)$	$TeCR(\%)$	$N_r$	$N_f$	$N_r/C$
brain tumor1	iGEC	$92.4 \pm 0.5$	$88.7 \pm 4.0$	$5.0 \pm 0.2$	$5.9 \pm 0.4$	1.00
	V200	80.85	81.25	6.50	8.60	1.30
	V15	78.66	85.00	6.00	9.20	1.20
brain tumor2	iGEC	$97.0 \pm 0.5$	$72.4 \pm 9.9$	$4.4 \pm 0.2$	$5.5 \pm 0.3$	1.11
	V200	60.00	60.00	4.00	8.30	1.00
	V15	66.60	63.33	5.10	6.70	1.27
DLBCL	iGEC	$98.5 \pm 0.3$	$91.2 \pm 2.6$	$2.5 \pm 0.1$	$3.7 \pm 0.3$	1.28
	V200	85.91	85.00	2.60	3.80	1.30
	V15	84.65	78.33	7.00	6.90	3.50
leukemia1	iGEC	$99.7 \pm 0.1$	$94.0 \pm 2.5$	$3.5 \pm 0.1$	$4.1 \pm 0.2$	1.18
	V200	90.15	92.00	5.30	7.30	1.76
	V15	87.61	84.00	4.90	8.10	1.63
leukemia2	iGEC	$98.7 \pm 0.2$	$85.3 \pm 4.4$	$3.3 \pm 0.1$	$4.3 \pm 0.3$	1.12
	V200	81.97	76.67	4.30	5.50	1.43
	V15	74.70	71.67	3.50	4.10	1.16
lung cancer	iGEC	$92.7 \pm 0.7$	$88.0 \pm 1.8$	$5.5 \pm 0.3$	$6.9 \pm 0.3$	1.10
	V200	85.35	84.44	7.80	14.50	1.56
	V15	81.57	82.78	8.30	8.90	1.66
prostate tumor	iGEC	$97.9 \pm 0.2$	$90.9 \pm 2.5$	$2.4 \pm 0.1$	$4.1 \pm 0.3$	1.21
	V200	81.50	82.00	3.00	3.30	1.50
	V15	84.46	84.00	2.90	5.10	1.45
SRBCT	iGEC	$99.8 \pm 0.1$	$92.3 \pm 2.7$	$4.3 \pm 0.2$	$4.8 \pm 0.4$	1.08
	V200	86.36	88.33	5.80	6.20	1.45
	V15	78.44	71.67	5.10	10.20	1.27
Mean	iGEC	97.1	87.9	3.9	5.0	1.1
	V200	81.5	81.2	4.9	7.2	1.4
	V15	79.6	77.6	5.4	7.4	1.6

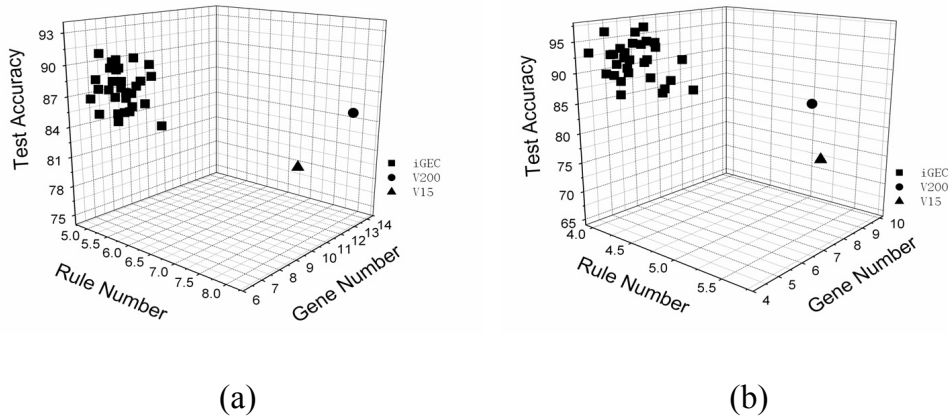


Figure 4.5. The 3D scatter plots. (a) lung cancer (b) SRBCT.

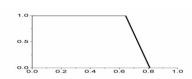
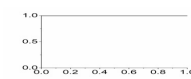
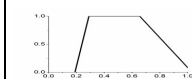
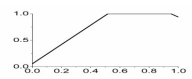
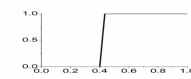
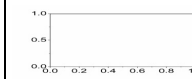
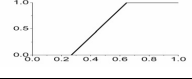
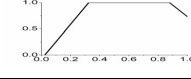

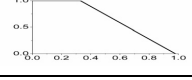
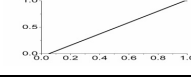
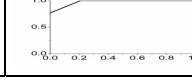
	Gene L05148	Gene U46499	Gene U05259	CL	CF
$R_1$				ALL B-Cell	0.243
$R_2$				ALL B-Cell	0.682
$R_3$				ALL T-Cell	0.710
$R_4$				AML	0.722

Figure 4.6. Fuzzy rules of the data set leukemia1 using 90% samples for training and the rest for test. The training and test accuracies are both 100%.

are linguistically interpretable as follows:

$R_1$ : If L05148 is not up-regulated and U05259 is not down-regulated, then Class “ALL B-Cell” with  $CF = 0.243$ ;

$R_2$ : If L05148 is ALL and U46499 is neutral or up-regulated, then Class “ALL B-Cell” with  $CF = 0.682$ ;

$R_3$ : If L05148 is not down-regulated, U46499 is ALL and U05259 is ALL, then Class “ALL T-Cell” with  $CF = 0.710$ ;

$R_4$ : If L05148 is ALL, U46499 is ALL and U05259 is ALL, then Class “AML” with  $CF = 0.722$ .

Where the membership functions of genes U46499 and U05259 in  $R_1$  and  $R_2$ , respectively, are “don’t care” which can reduce the rule length. From the compact rule base, it is easy to interpret the classification model from gene expression data. The fuzzy rules can be examined by biomedical researchers. Due to the natural clustering property of gene expression data, each of the classes “ALL T-Cell” and “AML” has one fuzzy rule corresponding to one fuzzy region while the class “ALL B-Cell” has two fuzzy regions overlapped. Furthermore, we can know the distribution of samples of each class from the corresponding membership function in the feature space. The fuzzy rule base can determine the class of unknown samples using Eq. 4.3.

To further realize whether these three genes L05148, U46499, and U05259 make sense

Table 4.3. Selected genes for the leukemia1 data set example. For each gene we counted the number of articles that were retrieved by a PubMed query consisting of the gene name and the string “leukemia”.

Gene	Description	# of References
M11722	Human terminal transferase mRNA	154
L05148	Human protein tyrosine kinase related mRNA sequence	26
M63138	Human cathepsin D	24
M31523	Human transcription factor (E2A) mRNA	17
U05259	Human MB-1 gene, complete cds	12
U46499	Homo sapiens microsomal glutathione transferase (MGST1) gene, 3' sequence	10
M27891	Human cystatin C gene	5
U16954	Human (AF1q) mRNA	3

as a group and their biological relationship, we process the average linkage (average distance, UPGMA) clustering based on Euclidean distances squared by EPCLUST [73]. Figure 4.7 shows the clustering result. From Figure 4.7, we can observe that most of the samples belonging to same class are grouped together. From thousands of genes, the proposed method can identify few but relevant genes to make accurate classification. Furthermore, the biological finding is interpretable from the obtained compact fuzzy rule base. Therefore, iGEC is beneficial to microarray data analysis and development of inexpensive diagnostic tests.

Besides the leukemia1 classifier using the gene set {L05148, U46499, U05259} shown in Figure 4.6, there are other sets of three genes which can establish the classifiers with both 100% training and test accuracies as follows: {L05148, M63138, U05259}, {M11722, L05148, U46499}, {M31523, U16954, U46499}, and {U16954, M27891, U05259}. This scenario results from that the microarray data have a large number of genes but a very small number of samples. iGEC can provide important knowledge to biological scientists. Table 4.3 gives descriptions of the selected genes from the data set leukemia1 of 72 samples. For each gene, we counted the number of articles that were retrieved by a PubMed query containing the gene name and the key string “leukemia”. By combining more gene sets of solutions, most of genes highly related to the leukemia disease can be obtained.

Due to different merits of fuzzy partitions such as grid partition, tree partition, and scatter partition, they cannot be directly compared using some specific measurements [37].

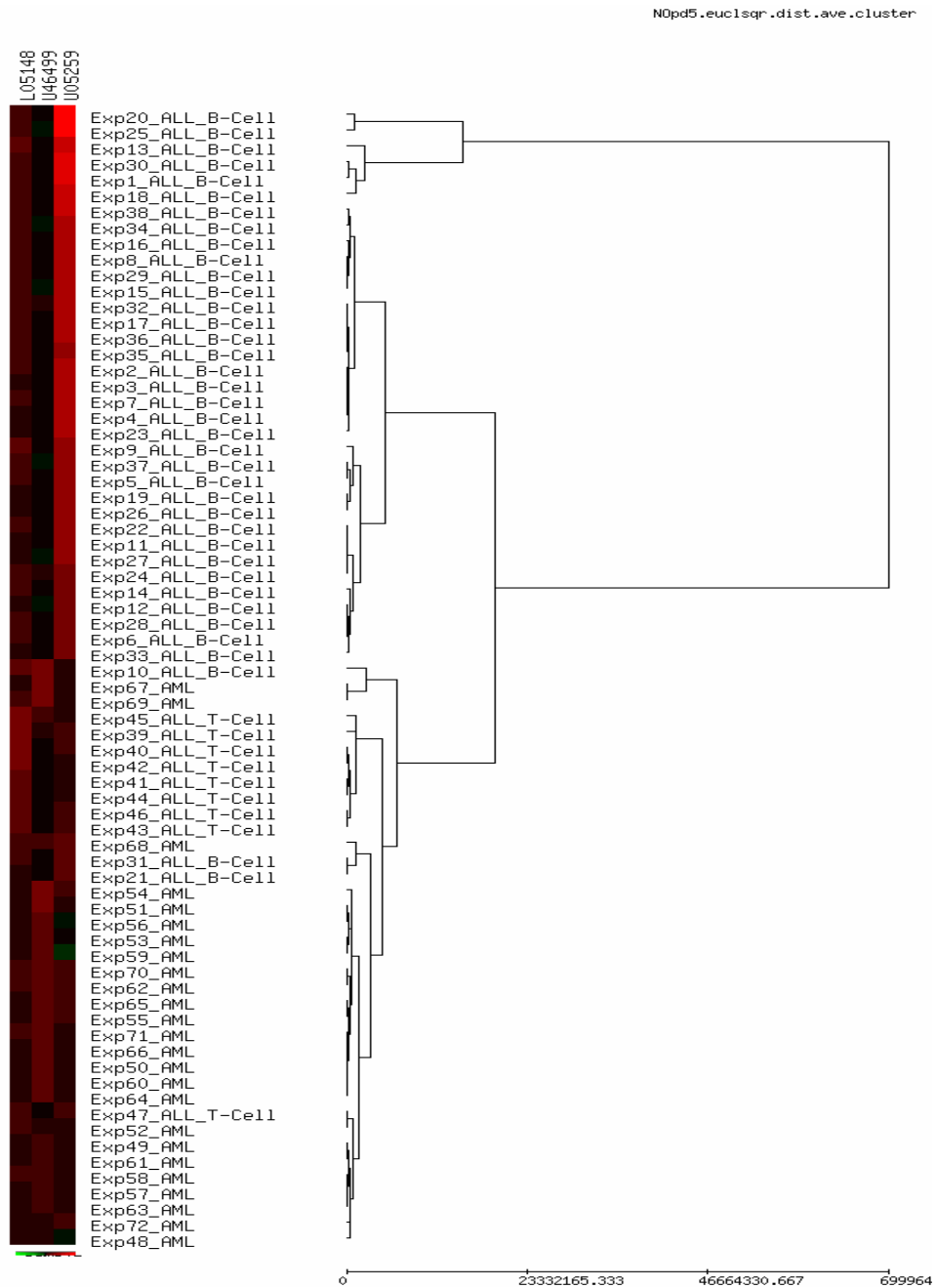


Figure 4.7. The clustering result of 72 samples in data set leukemia1 using the three selected genes by the clustering algorithm EPCLUST [73].



However, iGEC has 1.1 fuzzy regions for describing the sample distribution of each class averagely. Besides the above-mentioned advantages of easy interpretation and economical experiments, the proposed fuzzy rule-base method using a scatter partition of feature space can enclose all possible occurrences of samples in the same class with one or few hyperbox-type fuzzy regions. In other words, the fuzzy regions of scatter partition can represent one class more independently than those of grid partition. Therefore, iGEC can reject the unknown sample if it belongs to no fuzzy region that no fuzzy rule is fired.

### 4.2.3 Experiment 2-Comparison between iGEC and Non-rule-based Classifiers

To further evaluate accuracy of the proposed method, we compared iGEC with some non-rule-based classifiers without using gene selection methods in [12]. Table 4.4 shows the test accuracy comparisons using 10-CV on the eight data sets between iGEC and the following methods: multi-category support vector machine (SVM),  $k$ -nearest neighbors ( $k$ -NN), backpropagation neural networks (BNN), and probabilistic neural networks (PNN) which are the most common methods for gene expression data analysis. The results are obtained from [12].

Table 4.4 indicates that the multi-category SVM with 93.63% average test accuracy on the eight data sets is the most accurate classifier for diseases classification. However, it is not practical to use as many as 7965.6 genes on average to classify diseases samples for economical biomedical test in real applications. The proposed fuzzy classifier iGEC with 87.9% using 5.0 genes on average is superior to  $k$ -NN (84.49%), NN (82.54%), and PNN (79.49%) in terms of accuracy and number of genes. Because the sample sizes of microarray data are extremely small, it results in the high training accuracy (97.1%) and relatively low test accuracy (87.9%). When the number of samples is increased, the test accuracy can be further advanced [37]. From the viewpoint of analysis and practical applications, iGEC can serve as one of efficient tools for analysis of gene expression profiles.

Table 4.4. The test accuracies and numbers of used genes for iGEC and non-rule-based classifiers using 10-CV. The results of the non-rule-based classifiers without using gene selection methods are obtained from [12].

Data Set	# of genes in non- rule-based classifiers	Accuracy(%)					# of genes in iGEC
		SVM	k-NN	BNN	PNN	iGEC	
brain tumor1	5920	91.67	87.94	84.72	79.61	88.71	6
brain tumor2	10367	77.00	68.67	60.33	62.83	72.45	6
DLBCL	5469	97.50	86.96	89.64	80.89	91.22	4
leukemia1	5327	97.50	83.57	76.61	85.00	94.00	4
leukemia2	11225	97.32	87.14	91.03	83.21	85.33	4
lung cancer	12600	96.05	89.64	87.80	85.66	88.09	7
prostate tumor	10509	92.00	85.09	79.18	79.18	90.97	4
SRBCT	2308	100.00	86.90	91.03	79.50	92.33	5
Mean	7965.6	93.63	84.49	82.54	79.49	87.89	5.0

### 4.3 Discussions of iGEC

In pattern recognition problems, the scoring ability is important not only to quantify the certainty grades of samples belonging to each class, but also to help researchers to finding out the true active samples and filtering out the background noise [74]. Liu *et al.* [75] proposed a scoring algorithm based on negative entropy to position specific frequency matrix (PSFM) and Markov model to predict protein-DNA binding site. Murvai *et al.* [76] used a probabilistic scoring method for protein domain identification. Jensen and Liu [77] proposed a bayesian scoring function approach to motif discovery.

It is necessary to cope with the following difficulties in designing the scoring system, described below. 1) It is desirable to select a minimal number of relevant genes while maintaining the highest accuracy for designing tumor classifiers, which is essential for developing inexpensive diagnostic tests. 2) The derived scores can faithfully respond to accurate tumor classification with an interpretable manner. To achieve the above-mentioned goals, our proposed interpretable gene expression classifier (iGEC) can be extended to be a scoring method named iSFC, interpretable scoring fuzzy classifier.

The design of iSFC has the same three classification objectives as iGEC and one scoring function objectives to be simultaneously optimized: maximal classification accuracy,

minimal number of rules, minimal number of used features, and maximal area under a ROC curve. The detail designing implementation and the experimental results of iSFC can be referred from our previous research [78]. From the experimental results, we have shown that iSFC has concisely interpretable rules and better performance than the existing Vinterbo's classifier [13]. iSFC is also comparable to some non-rule-based methods [12] using a large number of genes in terms of accuracy performance. Furthermore, the efficient scoring ability of iSFC is evaluated using the mean areas under ROC curves having 0.984 and 0.930 for training and test data, respectively.

## 4.4 Conclusions for iGEC

Microarray data analysis and gene expression classification are important research topics in bioinformatics such that how to design an accurate, compact, and linguistically interpretable classifier is the major concern in this study. We proposed an interpretable gene expression classifier, named iGEC, for microarray data analysis. The design of iGEC includes almost all aspects related to the design of compact fuzzy rule-based classification systems: gene selection, rule selection, membership function tuning, consequent class determination, and certainty grade tuning. Consequently, an efficient optimization algorithm IGA is used to solve the resultant optimization problem with a large number of parameters.

The superiority of the proposed iGEC was evaluated by computer simulation on eight data sets of gene expression. The experimental results reveal that the proposed method can obtain interpretable classifiers with an accurate and compact fuzzy rule base, compared with the existing fuzzy classifier. iGEC is an efficient tool for analysis of gene expression profiles. Furthermore, the proposed iGEC can be extended to an interpretable scoring fuzzy classifier, named iSFC, which has the ability to effectively quantify the certainty grades of samples belonging to each class.

# Chapter 5

## Inference of Genetic Network

In this thesis, we propose an intelligent two-stage evolutionary algorithm (iTEA) to efficiently infer the S-system models of large-scale genetic networks from small-noise gene expression profiles using a single-processor PC. To cope with curse of dimensionality, the proposed algorithm consists of two stages where each uses a divide-and-conquer strategy. The optimization problem is first decomposed into  $N$  subproblems having  $2(N + 1)$  parameters. At the first stage, each subproblem is solved using the novel intelligent genetic algorithm (IGA) which is a specific variant of the intelligent evolutionary algorithm [38]. The intelligent crossover of IGA applies orthogonal experimental design (OED) [60, 61, 62] to speed up the search by using a systematic reasoning method instead of the conventional generate-and-go method of GA. At the second stage, the obtained  $N$  solutions to the  $N$  subproblems are combined and refined using an OED-based simulated annealing algorithm (OSA) [63] for handling noisy gene expression profiles. The effectiveness of iTEA is evaluated using simulated expression patterns with and without noise. It will be shown that: 1) IGA is efficient enough to solve subproblems; 2) IGA is significantly superior to the existing method SPXGA [29] in solving subproblems; and 3) iTEA performs well in inferring S-system models of genetic networks from small-noise gene expression profiles.

### 5.1 The Investigated Problem

#### 5.1.1 Problem Statement

Generally, the genetic network inference problem using an S-system model is formulated as a parameter optimization problem with  $2N(N + 1)$  S-system parameters  $(\alpha_i, \beta_i, g_{ij}, h_{ij})$

and the following objective function [29, 30, 31]:

$$\text{minimize } f = \sum_{i=1}^N \sum_{t=1}^T \left( \frac{X_{cal,i,t} - X_{exp,i,t}}{X_{exp,i,t}} \right)^2 \quad (5.1)$$

where  $X_{exp,i,t}$  is an experimentally observed expression level of gene  $i$  at time  $t$ , and  $X_{cal,i,t}$  is a numerically calculated expression level,  $N$  is the number of genes in the network, and  $T$  is the number of sampling points of observed data. When all S-system parameters are estimated,  $X_{cal,i,t}$  can be derived by using Eq. 2.2 and the given initial level  $X_{exp,i,0}$ .

Since the degree of freedom of an S-system model is high, multiple sets of time-series data are generally conducted to enhance the probability of finding correct solutions. Because of high cost of experiments, it is not convenient to get sufficient time-series data generally. Due to the high degree of freedom, inference of the S-system model often has multiple optimal solutions to best fit the observed time-series data [26, 29, 30, 31, 32, 33, 35]. The investigated problem is difficult due to the characteristics of high degree of freedom, high dimensionality, multimodality, strong interaction among parameters of the S-system model, and measurement noise. Therefore, it is hard to obtain a correct network structure with accurate parameter values. Generally, additional data or biological knowledge is needed to improve solution quality [26].

### 5.1.2 Useful Techniques

Two useful techniques in optimizing the objective function 5.1 are introduced. One is the problem decomposition strategy for large-scale genetic networks [32] and the other is to incorporate a priori knowledge to reduce computation cost [30, 31, 35], described below.

#### Problem decomposition

The large-scale problems of S-system models are difficult to solve directly. Maki et al. [32] proposed an efficient strategy of dividing the inference problem into  $N$  separated small subproblems. Each subproblem corresponds to one gene. The objective function of the  $i$ -th subproblem for gene  $i$  is as follows:

$$\text{minimize } f_i = \sum_{t=1}^T \left( \frac{X_{cal,i,t} - X_{exp,i,t}}{X_{exp,i,t}} \right)^2. \quad (5.2)$$

For noise-free or small-noise gene expression profiles, the expression level  $X_{cal,i,t}$  of gene  $i$  at time  $t$  can be numerically calculated by using Eq. 2.2. Otherwise, the following modified differential equations are used for large-noise gene expression profiles [30, 31]:

$$\frac{dX_i(t)}{dt} = \alpha_i \prod_{j=1}^N Y_j^{g_{ij}}(t-1) - \beta_i \prod_{j=1}^N Y_j^{h_{ij}}(t-1) \text{ where } \begin{cases} Y_j(t-1) = X_j(t-1) & \text{if } j = i \\ Y_j(t-1) = \hat{X}_j(t-1) & \text{otherwise} \end{cases}. \quad (5.3)$$

When  $2(N + 1)$  S-system parameters  $\{\alpha_i, g_{i1}, \dots, g_{iN}, \beta_i, h_{i1}, \dots, h_{iN}\}$  are estimated, we can obtain the estimated gene expression level  $X_{cal,i,t}$  for the  $i$ -th subproblem using Eq. 2.2 or Eq. 5.3 depending on the size of measurement noise. However, how to effectively obtain accurate  $\hat{X}_j$  is essentially important. To overcome the disadvantage of the problem decomposition when dealing the given data with large measurement noise [30], Kimura et al. [31] used a cooperative coevolutionary algorithm to simultaneously solve the subproblems by deriving  $\hat{X}_j$  from estimating the best individuals of the subproblems, each of which is given as a solution of Eq. 5.3. It is shown empirically that the method slightly enhanced the probability of finding the correct interactions of a network using a PC cluster [31].

### Adding a penalty term

In the S-system model, if there are no interaction between two genes  $i$  and  $j$ , the S-system parameters correspond to the interaction term,  $g_{ij}$  and  $h_{ij}$ , are zero. Because of the connectivity of the genetic network has been known to be sparse [79], the following fitness function incorporating a penalty term is conveniently added to reduce the search space and improve the accuracy of the inferred genetic network model [30, 31]:

$$\text{minimize } f_i = \sum_{t=1}^T \left( \frac{X_{cal,i,t} - X_{exp,i,t}}{X_{exp,i,t}} \right)^2 + c \sum_{j=1}^{N-I} (|G_{ij}| + |H_{i,j}|), \quad (5.4)$$

where  $c$  is a penalty weight,  $I$  is a maximum indegree that the maximal number of genes which directly affect gene  $i$ .  $G_{ij}$  and  $H_{ij}$  are given by rearranging  $g_{ij}$  and  $h_{ij}$  in ascending order of their absolute values. The penalty term forces most of the kinetic orders ( $g_{ij}$ ,  $h_{ij}$ ) down to zero. In the meantime, if the number of genes that directly affect the gene  $i$  is smaller than  $I$ , this term will not penalize. In such case, the optimal solutions to the

fitness functions Eq. 5.2 and Eq. 5.4 are identical. To reduce the computation cost, the structure skeletalizing technique [35] was applied. This technique assigns a value of zero to the kinetic orders when their absolute values are less than a given threshold  $\delta_s$ . In this study,  $\delta_s = 3 \times 10^{-2}$ .

## 5.2 The Proposed Intelligent Two-stage Evolutionary Algorithm (iTEA)

It is well recognized that divide-and-conquer is an efficient approach to solving large-scale problems. The divide-and-conquer mechanism breaks a large-scale problem into several subproblems that are similar to the original one but smaller in size, solves the sub-problems concurrently, and then combines these solutions to create a solution to the original problem. Figure 5.1 shows a flowchart of the proposed two-stage evolutionary algorithm iTEA. At the first stage,  $N$  solutions to the  $N$  subproblems are obtained by IGA which is an efficient population-based optimization algorithm. At the second stage, the  $N$  solutions are combined into an initial solution to be refined by searching for a globally optimal solution using OSA which is an efficient point-based optimization algorithm. Both IGA and OSA use the divide-and-conquer mechanism based on orthogonal experimental design.

### 5.2.1 Orthogonal Experimental Design and Factor Analysis

The two-level and three-level OAs are used for IGA and OSA, respectively. The two-level OAs used in IGA are described below. Let there be  $\alpha$  factors, with two levels each. The total number of level combinations is  $2^\alpha$  for a complete factorial experiment. To use an OA of  $\alpha$  factors, we obtain an integer  $M = 2^{\lceil \log_2(\alpha+1) \rceil}$  where the bracket represents an upper ceiling operation, build an OA  $L_M(2^{M-1})$  with  $M$  rows and  $M - 1$  columns, use the first  $\alpha$  columns, and ignore the other  $M - \alpha - 1$  columns. OA can reduce the number of level combinations for factor analysis. The number of OA combinations required to analyze all individual factors is only  $M = O(\alpha)$ , where  $\alpha + 1 \leq M \leq 2\alpha$ .

OSA uses three-level OAs where each factor has three levels. The total number of level combinations for  $\alpha$  factors is  $3^\alpha$  for a complete factorial experiment. To use a three-level

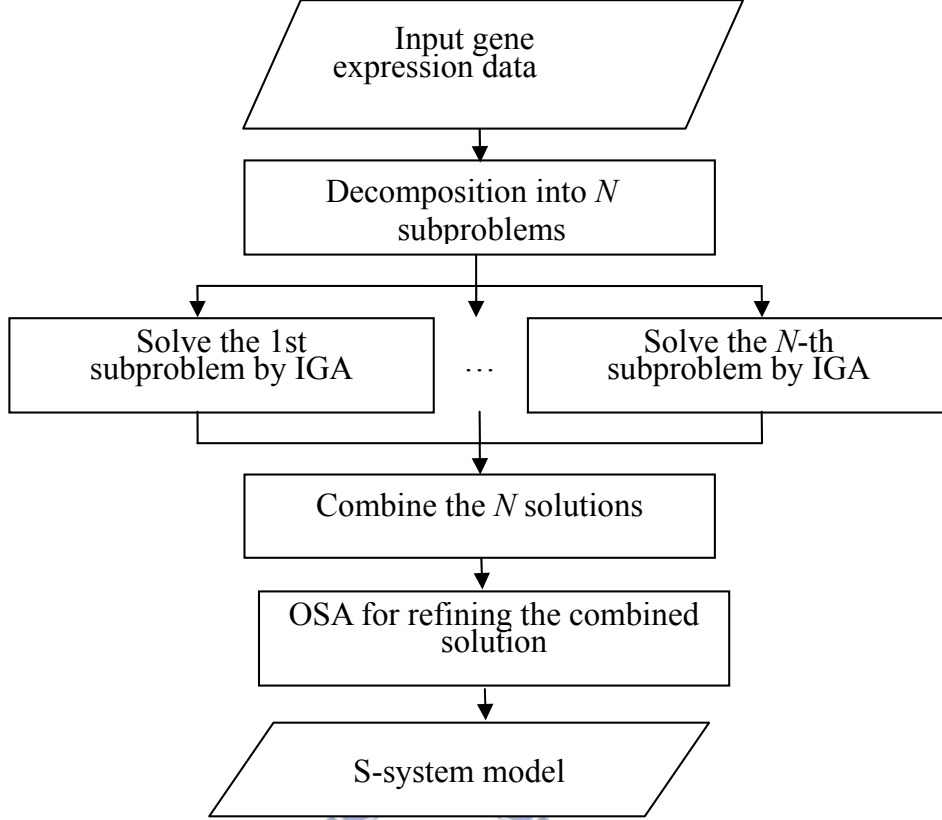


Figure 5.1. Flowchart of the proposed two-stage evolutionary algorithm iTEA.

OA of  $\alpha$  factors, we obtain an integer  $M = 3^{\lceil \log_3(2\alpha+1) \rceil}$ , build an OA  $L_M(3^{(M-1)/2})$  with  $M$  rows and  $(M-1)/2$  columns, use the first  $\alpha$  columns, and ignore the other  $(M-1)/2 - \alpha$  columns. The number of OA combinations required to analyze all individual factors is only  $M = O(\alpha)$ , where  $2\alpha + 1 \leq M \leq 6\alpha - 3$ .

Algorithm of constructing the two- and three-level OAs can be found in [63]. After proper tabulation of experimental results, the summarized data are analyzed using factor analysis to determine the relative effects of levels of various factors as follows. Let  $y_t$  denote a objective function value of the combination  $t$ , where  $t = 1, \dots, M$ . Define the main effect of factor  $d$  with level  $k$  as  $S_{dk}$  where  $d = 1, \dots, \alpha$ :

$$S_{dk} = \sum_{t=1}^M y_t W_t, \quad (5.5)$$

where  $W_t = 1$  if the level of factor  $d$  of combination  $t$  is  $k$ ; otherwise,  $W_t = 0$ . Consider that the objective function is to be minimized. For the two-level OA, level 1 of factor  $d$  makes a better contribution to the objective function than level 2 of factor  $d$  does when



$S_{d1} < S_{d2}$ . If  $S_{d1} > S_{d2}$ , level 2 is better. If  $S_{d1} = S_{d2}$ , levels 1 and 2 have the same contribution. The main effect reveals the individual effect of a factor. The most effective factor  $d$  has the largest main effect difference  $MED_d = |S_{d1} - S_{d2}|$ .

For the three-level OA, the level  $k$  of factor  $d$  makes the best contribution to the objective function than the other two levels of factor  $d$  do when  $S_{dk} = \min\{S_{d1}, S_{d2}, S_{d3}\}$ . On the contrary, if the objective function is to be maximized, the level  $k$  is the best one when  $S_{dk} = \max\{S_{d1}, S_{d2}, S_{d3}\}$ . The most effective factor has the largest one of main effect differences  $MED_d = \max\{S_{d1}, S_{d2}, S_{d3}\} - \min\{S_{d1}, S_{d2}, S_{d3}\}$ . After the better one of two/three levels of each factor is determined, a reasoned combination consisting of  $\alpha$  factors with the better/best levels can be easily derived.

## 5.2.2 IGA for Solving Subproblems

### Intelligent Crossover

The intelligent crossover plays an important role in IGA. IGA solves an individual subproblem with  $N$  genes having  $2(N+1)$  parameters to be optimized. The intelligent crossover uses a divide-and-conquer approach, which consists of adaptively dividing two parents into  $\alpha$  pairs of parameter groups, economically identifying the potentially better one of two groups of each pair, and systematically obtaining a potentially good approximation to the best one of all  $2^\alpha$  combinations using at most  $2\alpha$  fitness evaluations. Like traditional GAs, two parents  $P_1$  and  $P_2$  produce two children  $C_1$  and  $C_2$  using one crossover operation. The intelligent crossover determines the recombination of  $P_1$  and  $P_2$  for efficiently generating good children. Let the set of parameters in the  $i$ -th subproblem be  $\{\alpha_i, g_{i1}, \dots, g_{iN}, \beta_i, h_{i1}, \dots, h_{iN}\}$ . We divided the two sets  $INC = \{\alpha_i, g_{i1}, \dots, g_{iN}\}$  and  $DEC = \{\beta_i, h_{i1}, \dots, h_{iN}\}$  which control the gene expression level increasing or decreasing into  $\lceil \alpha/2 \rceil$  and  $\lfloor \alpha/2 \rfloor$  groups, respectively. To make a sufficient use of all columns in OAs,  $\alpha$  is usually set to  $2^\omega - 1$  where  $\omega$  is an integer. In this study, we used  $\alpha = 7$  for problems with  $N \leq 30$ . The value of  $\alpha$  would properly increase when  $N$  increases. The discussion between  $\alpha$  and the number of parameters to be optimized can be referred to [38].

Because the parameters belonging to the same one of two sets  $INC$  and  $DEC$  have strong interactions, we don't use the conventional encoding scheme of GA that all pa-

rameters are encoded into a chromosome in a fixed order. Instead, all parameters are represented using real values with no order. For each time using an intelligent crossover operation, *INC* and *DEC* are randomly divided into  $\lceil \alpha/2 \rceil$  and  $\lfloor \alpha/2 \rfloor$  groups with a variable size for each group. The parameters of two parents are grouped using the same division operation. Each group is treated as a factor. The  $\alpha$  factors are randomly numbered in using OED. The numbering order does not affect the effectiveness of intelligent crossover because of the property of OA [38]. Note that there is no fixed genotype of S-system parameters used. The following steps describe how to use OED with  $\alpha$  factors to achieve the intelligent crossover of IGA for a fitness function  $y$ .

- Step 1: The two sets  $INC = \{\alpha_i, g_{i1}, \dots, g_{iN}\}$  and  $DEC = \{\beta_i, h_{i1}, \dots, h_{iN}\}$  of S-system parameters are randomly divided into  $\lceil \alpha/2 \rceil$  and  $\lfloor \alpha/2 \rfloor$  groups (factors), respectively.
- Step 2: Use a two-level OA  $L_{\alpha+1}(2^\alpha)$  with  $\alpha + 1$  rows and  $\alpha$  columns.
- Step 3: Let levels 1 and 2 of factor  $d$  represent the  $d$ -th groups coming from parents  $P_1$  and  $P_2$ , respectively.
- Step 4: Evaluate the fitness values  $y_t$  for experiment  $t$  where  $t = 2, \dots, \alpha + 1$ . The value  $y_1$  is the fitness value of  $P_1$ .
- Step 5: Compute the main effect  $S_{dk}$  where  $d = 1, \dots, \alpha$  and  $k = 1, 2$ .
- Step 6: Determine the better one of two levels of each factor according to the main effect.
- Step 7: The chromosome of  $C_1$  is formed using the combination of the better groups from the derived corresponding parents.
- Step 8: The chromosome of  $C_2$  is formed similarly as  $C_1$ , except that the factor with the smallest main effect difference adopts the other level.
- Step 9: The best two individuals among  $P_1, P_2, C_1, C_2$ , and  $\alpha$  combinations of OA are used as the final children  $C_1$  and  $C_2$  for elitist strategy.

One intelligent crossover operation takes  $\alpha + 2$  fitness evaluations to explore the search space of  $2^\alpha$  combinations. Generally,  $C_1$  is a potentially good approximation to the best one of  $2^\alpha$  combinations.

## Illustrative Example of Intelligent Crossover

Tables 5.1 and 5.2 show an illustrative example of using intelligent crossover with OED in solving the first subproblem of inferring an S-system model with  $N = 5$ . The details of the test problem are given in Section 5.3.1. We used an OA  $L_8(2^7)$  for  $\alpha = 7$ . The two sets of S-system parameters  $INC = \{\alpha_1, g_{11}, \dots, g_{15}\}$  and  $DEC = \{\beta_1, h_{11}, \dots, h_{15}\}$  are randomly divided and assigned to four and three groups (factors) respectively as follows:  $V_1 = \{h_{13}, h_{15}\}$ ,  $V_2 = \{g_{14}\}$ ,  $V_3 = \{g_{12}, g_{13}\}$ ,  $V_4 = \{\alpha_1, g_{15}\}$ ,  $V_5 = \{h_{11}, h_{12}\}$ ,  $V_6 = \{\beta_1, h_{14}\}$ , and  $V_7 = \{g_{11}\}$ . The parameter values of parents are given in Table 5.2. Table 5.1 shows all results of intelligent crossover using OED. First, we evaluate the response variable  $y_t$  of the combination  $t$ , where  $t = 1, 2, \dots, 8$ . Second, we compute the main effect  $S_{dk}$  where  $d = 1, 2, \dots, 7$  and  $k = 1, 2$ . For example,  $S_{22} = y_3 + y_4 + y_7 + y_8 = 147.65$ . Third, the better level of each factor based on the main effect is determined. For example, the better level of factor 1 is level 2 since  $S_{12}(153.97) < S_{11}(157.50)$ . Finally, the better levels of factors ( $V_1, V_2, V_3, V_4, V_5, V_6, V_7$ ) are (2, 2, 1, 1, 1, 2, 2) and then  $y = 30.22$  can be obtained from the reasoned combination. This reasoned combination is used to form the child  $C_1$  of the crossover operation. The least effective factor is  $d = 5$  with  $MED_5 = 2.06$  which is the smallest one, so the second child  $C_2$  is formed similarly as  $C_1$  except  $V_5$ , which adopts level 2. Note that the ranks of  $C_1$  and  $C_2$  are 2 and 4 respectively among 128 combinations of a complete factorial experiment. It reveals that the reasoning operation of intelligent crossover for generating children is efficient.

## The Used Intelligent Genetic Algorithm

IGA is used to solve the  $N$  individual subproblems with the fitness function Eq. 5.4. The gene expression level of  $X_{cal,i,t}$  is numerically calculated using Eq. 2.2 rather than Eq. 5.3 due to the following reasons:

- 1) According to the simulation using IGA, the method using Eq. 2.2 is simple and fast, and its solution is accurate enough in terms of fitness value from noise-free gene expression profiles.
- 2) We would further refine the combined solutions of the  $N$  subproblems from the aspect of global optimization using OSA for handling noisy gene expression profiles.

Table 5.1. An Illustrative Example of Intelligent Crossover Using OA  $L_8(2^7)$ .

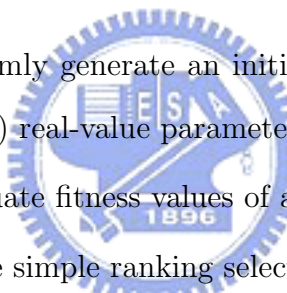
Combination $t$	Factor $d$							Rank	
	$V_1$	$V_2$	$V_3$	$V_4$	$V_5$	$V_6$	$V_7$		
1	1	1	1	1	1	1	1	40.12	81/128
2	1	1	1	2	2	2	2	39.70	72/128
3	1	2	2	1	1	2	2	36.34	23/128
4	1	2	2	2	2	1	1	41.35	97/128
5	2	1	2	1	2	1	2	41.40	102/128
6	2	1	2	2	1	2	1	42.60	125/128
7	2	2	1	1	2	2	1	34.32	13/128
8	2	2	1	2	1	1	2	35.65	18/128
$S_{d1}$	157.60	163.81	149.78	152.17	154.70	158.51	158.39		
$S_{d2}$	153.97	147.65	161.68	159.30	156.76	152.95	153.08		
$MED_d$	3.54	16.16	11.90	7.13	2.06	5.56	5.31		
Child 1( $C_1$ )	2	2	1	1	1	2	2	30.22	2/128
Child 2( $C_2$ )	2	2	1	1	2	2	2	30.80	4/128

Table 5.2. The Contents of Parents and Children.

	$\alpha_1$	$g_{11}$	$g_{12}$	$g_{13}$	$g_{14}$	$g_{15}$	$\beta_{11}$	$h_{11}$	$h_{12}$	$h_{13}$	$h_{14}$	$h_{15}$	$y$
$P_1$	2.98	2.08	0.98	-1.17	2.13	0.00	2.71	2.04	0.10	1.89	2.33	-0.06	40.12
$P_2$	12.68	0.58	-0.24	0.79	0.33	0.82	10.69	0.74	2.31	3.00	1.33	-1.33	32.68
$C_1$	12.68	0.58	-0.24	0.79	0.33	0.82	2.71	2.04	0.10	1.89	2.33	-0.06	30.22
$C_2$	12.68	0.58	-0.24	0.79	0.33	0.82	2.71	0.74	2.31	1.89	2.33	-0.06	30.80

- 3) The estimation method for  $\hat{X}_j$  in Eq. 5.3 using a cooperative coevolutionary algorithm on a PC cluster [31] is not suitable for the IGA-based method because that IGA solves each subproblem independently on a single-processor PC. Furthermore, the method using estimation of  $\hat{X}_j$  only slightly enhanced the probability of finding the correct interactions of a network [31].

The main differences of the used IGA from the conventional GAs are chromosome encoding and crossover operation mentioned above. Besides, the used mutation is also different from the conventional one, described as follows. Assume a real-value parameter  $x$  is to be mutated. A perturbation  $\bar{x}$  is generated by the Cauchy-Lorentz probability distribution [59]. The mutated value of  $x$  is  $x' = x + \bar{x}$  or  $x - \bar{x}$ , determined randomly. If  $x'$  is out of the domain range of  $x$ , we randomly assign a feasible value to  $x'$ . The used simple IGA is described below.

- 
- Step 1: (Initiation) Randomly generate an initial population with  $N_{pop}$  feasible individuals of  $2(N + 1)$  real-value parameters.
- Step 2: (Evaluation) Evaluate fitness values of all individuals.
- Step 3: (Selection) Use the simple ranking selection that replaces the worst  $P_s \times N_{pop}$  individuals with the best  $P_s \times N_{pop}$  individuals to form a new population, where  $P_s$  is a selection probability. Let  $I_{best}$  be the best individual in the population.
- Step 4: (Crossover) Randomly select  $P_c \times N_{pop}$  individuals including  $I_{best}$ , where  $P_c$  is a crossover probability. Perform intelligent crossover operations for all selected pairs of parents.
- Step 5: (Mutation) Apply the above-mentioned mutation operator to the population using a mutation probability  $P_m$ . To prevent the best fitness value from deteriorating, mutation is not applied to the best individual.
- Step 6: (Termination test) If a prespecified number  $N_{eval}$  of fitness evaluations is achieved or some stopping condition is met, then stop the algorithm. Otherwise, go to Step 2.

### 5.2.3 OSA for Refining the Combined Solution

To compensate the disregard of estimating accurate gene expression levels of other genes from noisy data of gene expression, all the solutions to  $N$  subproblems are combined and then refined using OSA from the aspect of global optimization. The main difference of OSA from the conventional simulated annealing is the move generation mechanism, as shown in Figure 5.2 [63]. OSA uses an intelligent generation mechanism (IGM) based on OED to systematically reason a good candidate solution as the next move. The high performance of OSA arises from IGM which is the main phase of OSA. IGM is similar to the intelligent crossover of IGA using the divide-and-conquer mechanism for large-scale optimization problems, which is also efficient in determining a good approximation to the best solution in the neighborhood of the current solution. OSA uses the following objective function for global optimization:

$$\text{minimize } F = \sum_{i=1}^N \sum_{t=1}^T \left( \frac{X_{cal,i,t} - X_{exp,i,t}}{X_{exp,i,t}} \right)^2 + c \sum_{i=1}^n \sum_{j=1}^{N-I} (|G_{ij}| + |H_{i,j}|). \quad (5.6)$$

The following two section describe the used IGM and give the procedure of global optimization using OSA.

#### Intelligent Generation Mechanism (IGM)

Let all the  $N$  solutions be combined into an initial solution  $S$  of OSA to be refined. Let  $S = (s_1, \dots, s_p)$  where  $s_i$  is one of S-system parameters and  $p = 2N(N + 1)$ . IGM generates two temporary solutions  $S_A = (a_1, \dots, a_p)$  and  $S_B = (b_1, \dots, b_p)$  by perturbing  $S$ , where  $a_i$  and  $b_i$  are defined as follows:

$$a_i = s_i + s'_i; \quad b_i = s_i - s'_i, \quad i = 1, \dots, p. \quad (5.7)$$

The values of  $s'_i$  are generated by the Cauchy-Lorentz probability distribution. IGM aims at efficiently combining good values of parameters from solutions  $S$ ,  $S_A$  and  $S_B$  to generate a good candidate solution  $Q$  for the next move of  $S$ .

Divide all the  $p$  parameters into  $m$  nonoverlapping groups with variable sizes using the same division operation for  $S$ ,  $S_A$  and  $S_B$ . In this study, the used OA is  $L_{2m+1}(3^m)$  and  $m = 13$  for  $N \leq 30$ . How to decide the proper value of  $m$  and OA can be referred to

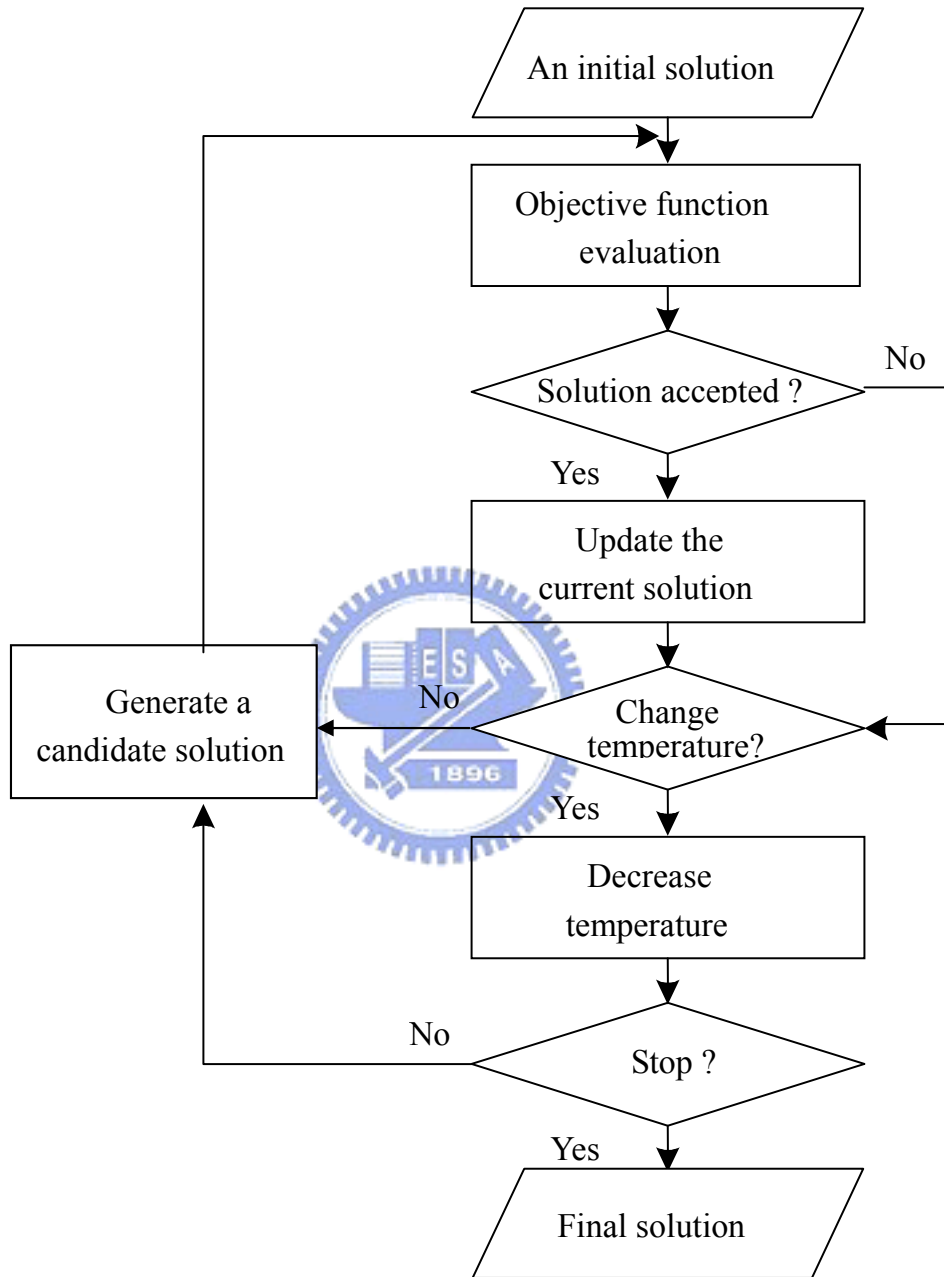


Figure 5.2. Flowchart of OSA with an intelligent generation mechanism applying a systematic reasoning method based on orthogonal experimental design instead of the conventional generate-and-test method [63].



[63]. Due to the same reason with that of intelligent crossover, the two sets  $\{\alpha_i, g_{ij}\}$  and  $\{\beta_i, h_{ij}\}$  are randomly divided into seven and six parameter groups, respectively. How to perform an IGM operation with  $m$  groups using a current solution  $S$  and the objective function  $F$  in Eq. 5.6 is described as follows:

Step 1: Generate two temporary solutions  $S_A$  and  $S_B$  using  $S$ .

Step 2: Using the same division operation, the two sets  $INC = \{\alpha_i, g_{i1}, \dots, g_{iN}\}$  and  $DEC = \{\beta_i, h_{i1}, \dots, h_{iN}\}$  in each of  $S$ ,  $S_A$ , and  $S_B$  are randomly divided into  $(m+1)/2$  and  $(m-1)/2$  groups (factors), respectively.

Step 3: Let levels 1, 2, and 3 of factor represent the groups coming from  $S$ ,  $S_A$ , and  $S_B$ , respectively.

Step 4: Compute  $y_t$  of the generated combination, where  $t = 2, 3, \dots, 2m+1$ . Note that  $y_1 = F(S)$ .

Step 5: Compute the main effect  $S_{dk}$  where  $d = 1, \dots, m$  and  $k = 1, 2, 3$ .

Step 6: Determine the best one of three levels of each factor according to the main effect.

Step 7: The candidate solution  $Q$  is formed using the combination of the best groups.

Step 8: Verify that  $Q$  is superior to the  $2m$  sampling solutions derived from the OA combinations and  $Q \neq S$ . If it is not true, select the best one from the  $2m$  sampling solutions as  $Q$ .

The number of objective function evaluations is  $2m+1$  per IGM operation, which includes  $2m$  evaluations in Step 4 and one in Step 8.

### Global Optimization Using OSA

The main power of OSA mainly arises from using IGM to efficiently search for a good candidate solution. OSA uses a simple geometric cooling rule by updating the temperature at the  $(i+1)$ -th temperature step using the formula:  $Temp_{i+1} = CR \cdot Temp_i$ ,  $i = 0, 1, \dots$ , where  $CR$  is the cooling rate which is a constant smaller than 1 but close to 1 (e.g.,  $CR = 0.99$ ). The higher the temperature, the larger it is the possibility of accepting the

candidate solution worse than the current solution. In this study, a simple version of OSA proposed in [63] is used. The used OSA is described below.

Step 1: (Initialization) Initialize  $Temp = Temp_0$  and  $CR$ . Let the combined solution  $S$  be an initial solution and compute  $F(S)$ .

Step 2: (Perturbation) Perform an IGM operation using  $S$  to generate a candidate solution  $Q$ .

Step 3: (Acceptance criterion) Accept  $Q$  to be the new  $S$  with probability  $P(Q)$ :

$$P(Q) = \begin{cases} 1 & , \text{ if } F(Q) \leq F(S), \\ \exp\left(\frac{F(S)-F(Q)}{Temp}\right) & , \text{ if } F(Q) > F(S). \end{cases} \quad (5.8)$$

Step 4: (Decreasing temperature) Let the new value of  $Temp$  be  $CR \cdot Temp$ .

Step 5: (Termination test) If a pre-specified stopping criterion is met, stop the algorithm. Otherwise, go to Step 2.

OA specifies a small number of representative combinations that are uniformly distributed over the neighborhood of the current solution. Furthermore, the factor analysis makes IGM more efficient in obtaining a good candidate solution which is a potentially good approximation to the best solution in the neighborhood of the current solution. When OSA is compared with SA using the same number of function evaluations, the actual computation time of OSA is generally much smaller than that of SA because OSA uses a smaller number of iterations [63].

In this study, if the solutions to subproblems are accurate enough whose fitness values are sufficiently small, OSA plays a role in finely tuning the values of parameters but not the structure from the aspect of global optimization. In such case,  $Temp_0$  can be set to a very small value. If the fitness values are not satisfactory,  $Temp_0$  can be enlarged to search for a better solution which the structure of the S-system model may be modified.

#### 5.2.4 iTEA Using IGA and OSA

The proposed algorithm iTEA uses both IGA and OSA in stages 1 and 2, respectively. IGA aims to obtain solutions to subproblems with significant accuracy in terms of the

objective function value which can best fit the given gene expression profiles. If noise is very small, IGA is effective enough and the improvement of OSA in stage 2 is not significant. When noise becomes larger, the best fit of the observed gene expression profiles is leaved to OSA from the aspect of global optimization. The proposed evolutionary divide-and-conquer approach using two stages is given as follows:

Stage 1: Apply IGA to solve  $N$  individual subproblems independently using the fitness function Eq. 5.4. The gene expression level of  $X_{cal,i,t}$  at time  $t$  is numerically calculated using Eq. 2.2.  $R > 1$  independent runs are conducted for each subproblem and the best solution to each subproblem is selected.

Stage 2: Combine these  $N$  best solutions  $(\alpha_i, g_{i1}, \dots, g_{iN}, \beta_i, h_{i1}, \dots, h_{iN})$ ,  $i = 1, \dots, N$  into an initial solution  $S$ . Apply OSA to refine  $S$  using an objective function  $F$  in Eq. 5.6.

## 5.3 Experimental Results of iTEA

We conducted some evaluations for the proposed algorithm iTEA. The used parameters in iTEA are described below. For IGA,  $N_{pop} = 20$ ,  $P_s = 0.2$ ,  $P_c = 0.8$ , and  $P_m = 0.2$ . The penalty coefficient  $c = 1.0$  in Eq. 5.4 and Eq. 5.6. For noisy gene expression profiles,  $R = 10$  for IGA in stage 1. From our computer simulation results, OSA performs well generally by giving a very small value to  $Temp_0$ , e.g., 0.001.

### 5.3.1 Experiment 1-Performance of IGA

In this experiment, two target genetic networks with  $N = 5$  and 10 are used to evaluate the performance of IGA. For a small-scale target network, we used the same S-system model of a genetic network consisting of  $N = 5$  genes from [29]. This model has been developed to analyze the interaction of regulator and effector genes, which has feedback loops. The S-system parameters of the target network are given in Table 5.3. The ranges of the S-system parameters are  $[0, 15.0]$  for  $\alpha_i$  and  $\beta_i$ , and  $[-3.0, 3.0]$  for  $g_{ij}$  and  $h_{ij}$ . To enhance the probability of finding the correct solution, 15 sets of noise-free time-series data were used where each covering all 5 genes as a sufficient amount of observed gene

expression profiles. The sets of time-series data were obtained by using Eq. 2.2. The initial values of these sets are listed in Table 5.4.  $T = 11$  sampling points for the time-series data were assigned on each gene in each set, and hence the observed time-series data on each gene consist of  $15 \times 11 = 165$  sampling points. For this network model, we have to estimate 60 parameters of the S-system. Let  $I = 2$  and  $N_{eval} = 2 \times 10^5$  for IGA in this experiment. Because of no measurement noise, no refinement of using OSA is required.

Table 5.3. The S-system parameters of a small-scale target network with  $N = 5$  from [29].

$i$	$\alpha_i$	$g_{i1}$	$g_{i2}$	$g_{i3}$	$g_{i4}$	$g_{i5}$	$\beta_i$	$h_{i1}$	$h_{i2}$	$h_{i3}$	$h_{i4}$	$h_{i5}$
1	5	0	0	1	0	-1	10	2	0	0	0	0
2	10	2	0	0	0	0	10	0	2	0	0	0
3	10	0	-1	0	0	0	10	0	-1	2	0	0
4	8	0	0	2	0	-1	10	0	0	0	2	0
5	10	0	0	0	2	0	10	0	0	0	0	2

Table 5.4. 15 Sets of initial gene expression levels of the target network with  $N = 5$  from [29].

Set	$X_1$	$X_2$	$X_3$	$X_4$	$X_5$
1	0.70	0.12	0.14	0.16	0.18
2	0.10	0.70	0.14	0.16	0.18
3	0.10	0.12	0.70	0.16	0.18
4	0.10	0.12	0.14	0.70	0.18
5	0.10	0.12	0.14	0.16	0.70
6	0.70	0.70	0.14	0.16	0.70
7	0.10	0.70	0.70	0.16	0.18
8	0.10	0.12	0.70	0.70	0.18
9	0.10	0.12	0.14	0.70	0.70
10	0.70	0.12	0.14	0.16	0.70
11	0.20	0.70	0.14	0.10	0.40
12	0.10	0.15	0.10	0.16	0.18
13	0.30	0.12	0.70	0.10	0.10
14	0.10	0.25	0.70	0.10	0.30
15	0.20	0.12	0.70	0.16	0.20

The S-system parameters of the best solution in terms of fitness value from 30 independent runs are listed in Table 5.5. The fitness value is smaller than  $10^{-6}$  for each subproblem using Eq. 5.4. The genetic structure is correct and the parameter values are precise enough to biologically interpret the network. Our method running on a single-processor

PC (Pentium III 933 MHz) takes 5.8 minutes averagely to solve the five subproblems without using any auxiliary technique to save computation time, such as the estimation of slopes in the preprocessing using an artificial neural network [26]. This is far less computation time from a comparison with the method PEACE1 (Predictor by Evolutionary Algorithms and Canonical Equations 1) proposed in Kikuchi et al. [29] using SPXGA. PEACE1 running on a PC cluster (Pentium III 933 MHz  $\times$  1040 processors) took more than 10 hours to estimate the same S-system parameters. The method [31] running on a PC cluster (Pentium III 933 MHz  $\times$  8 processors) required about 89.0 minutes to solve the same problem.

Table 5.5. The estimated S-system parameters sets with  $N = 5$ . The fitness value of each subproblem is smaller than  $10^{-6}$ .

$i$	$\alpha_i$	$g_{i1}$	$g_{i2}$	$g_{i3}$	$g_{i4}$	$g_{i5}$
1	5.002163	0	0	0.999995	0	-0.999868
2	10.000714	1.999865	0	0	0	0
3	9.997750	0	-1.000208	0	0	0
4	7.991448	0	0	2.000909	0	-1.000599
5	10.004309	0	0	0	1.998261	0
$i$	$\beta_i$	$h_{i1}$	$h_{i2}$	$h_{i3}$	$h_{i4}$	$h_{i5}$
1	10.001932	1.996719	0	0	0	0
2	9.996852	0	1.999167	0	0	0
3	10.000212	0	-1.000058	1.999683	0	0
4	9.992125	0	0	0	2.006558	0
5	9.976999	0	0	0	0	1.994196

Another genetic network of  $N = 10$  genes is given in Tables 5.6 and 5.7. There are 169 and 51 zero and non-zero values of S-system parameters, respectively. We used 15 sets of gene expression profiles with  $I = 3$  and  $T = 11$ . The stopping condition is when the fitness value is still not improved after 100 generations. The best solution in terms of fitness value from 30 independent runs is listed in Table 5.8. The corresponding fitness values in Eq. 5.4 for 10 subproblems are given in Table 5.9. There are no false-negative interaction and only 9 false-positive interactions whose magnitudes are relatively small. This experiment running on a single-processor PC (P4 2.8 GHz) takes 4.5 minutes averagely for one subproblem. From these encouraging results, IGA has the ability to efficiently solve the inference problems with  $N = 5$  and 10, better than the SPXGA-based method [29] which can infer the dynamics of a small network with  $N = 5$ .

Table 5.6. S-system parameters of target network with  $N = 10$ .

$i$	$\alpha_i$	$g_{i1}$	$g_{i2}$	$g_{i3}$	$g_{i4}$	$g_{i5}$	$g_{i6}$	$g_{i7}$	$g_{i8}$	$g_{i9}$	$g_{i10}$
1	5	0	0	0	1	0	-2	0	0	0	0
2	10	0	0	1	0	0	0	0	-1	0	0
3	8	-1	0	0	-1	0	0	0	0	0	0
4	10	0	0	0	0	2	0	0	0	1	0
5	10	0	2	0	0	0	-1	0	0	0	0
6	5	0	0	0	0	0	0	0	0	2	-2
7	10	0	0	0	0	0	1	0	0	0	-1
8	5	1	-2	0	0	0	0	1	0	0	0
9	10	0	0	1	0	0	0	0	-2	0	0
10	8	2	0	0	0	0	0	-1	0	0	0

$i$	$\beta_i$	$h_{i1}$	$h_{i2}$	$h_{i3}$	$h_{i4}$	$h_{i5}$	$h_{i6}$	$h_{i7}$	$h_{i8}$	$h_{i9}$	$h_{i10}$
1	10	2	0	0	0	0	0	0	0	0	0
2	10	0	2	0	0	0	0	0	0	0	0
3	10	0	0	2	0	0	0	0	0	0	0
4	10	0	0	0	2	0	0	0	0	0	0
5	10	0	0	0	0	2	0	0	0	0	0
6	10	0	0	0	0	0	2	0	0	0	0
7	10	0	0	0	0	0	0	2	0	0	0
8	10	0	0	0	0	0	0	0	2	0	0
9	10	0	0	0	0	0	0	0	0	2	0
10	10	0	0	0	0	0	0	0	0	0	2

### 5.3.2 Experiment 2-Comparison between SPXGA and IGA

In this experiment, we conducted some experiments using S-system models containing  $N = 5, 10, \dots, 30$  genes as target networks to show that IGA is significantly better than SPXGA [29] for solving subproblems in terms of fitness value using the same number  $N_{eval}$  of fitness evaluations.

We generated feasible expression patterns for comparing the optimization abilities of IGA and SPXGA. Six sets of time-series data of gene expression are generated where each covering all  $N$  genes. The values of gene expression levels are generated in the range  $[0, 1.0]$ . The parameters  $\alpha_i$  and  $\beta_i \in [0, 15.0]$  and  $g_{ij}$  and  $h_{ij} \in [-3.0, 3.0]$ . Let  $I = 3$  and  $T = 11$ . We conducted 30 independent runs for the first subproblem of each experiment using IGA and SPXGA to compare the effectiveness by a statistical t-test method. For each experiment, the stopping condition is  $N_{eval} = 5000 \times N$ . The parameters used in SPXGA are identical to those in [29].

Figure 5.3 gives the convergences of 30 runs for comparisons between IGA and SPXGA

Table 5.7. 15 Sets of initial gene expression levels of the target network with  $N = 10$ .

Set	$X_1$	$X_2$	$X_3$	$X_4$	$X_5$	$X_6$	$X_7$	$X_8$	$X_9$	$X_{10}$
1	0.10	0.12	0.14	0.16	0.18	0.20	0.22	0.24	0.26	0.28
2	0.70	0.12	0.14	0.16	0.18	0.20	0.22	0.24	0.26	0.70
3	0.10	0.70	0.14	0.16	0.18	0.20	0.22	0.24	0.70	0.28
4	0.10	0.12	0.70	0.16	0.18	0.20	0.22	0.70	0.26	0.28
5	0.10	0.12	0.14	0.70	0.18	0.20	0.70	0.24	0.26	0.28
6	0.10	0.12	0.14	0.16	0.70	0.20	0.22	0.24	0.26	0.28
7	0.10	0.12	0.14	0.16	0.18	0.70	0.22	0.24	0.26	0.28
8	0.10	0.12	0.14	0.16	0.18	0.20	0.70	0.24	0.26	0.28
9	0.10	0.12	0.14	0.16	0.18	0.20	0.22	0.70	0.26	0.28
10	0.10	0.12	0.14	0.16	0.18	0.20	0.22	0.24	0.70	0.28
11	0.14	0.12	0.14	0.16	0.70	0.20	0.22	0.24	0.26	0.10
12	0.45	0.10	0.14	0.16	0.10	0.10	0.22	0.70	0.26	0.10
13	0.20	0.12	0.14	0.10	0.18	0.10	0.70	0.24	0.70	0.10
14	0.50	0.10	0.14	0.16	0.18	0.20	0.70	0.70	0.26	0.10
15	0.30	0.12	0.10	0.16	0.18	0.20	0.22	0.24	0.70	0.10

with various values of  $N$ . It can be seen obviously that IGA is superior to SPXGA, especially when  $N$  is large. Table 5.10 shows the t-test results of the six target networks. It can be found that the mean fitness values and variances of IGA are much smaller than those of SPXGA where their p value is near to zero. From these results, it reveals that IGA is significantly better than SPXGA in solving the individual subproblems in a limited amount of computation time.

### 5.3.3 Experiment 3-iTEA for noisy gene expression profiles

In this experiment, we will evaluate the proposed iTEA using noisy gene expression profiles. We adopted the Gaussian noise which is commonly used for the simulated experiment [80]. Three target genetic networks with  $N = 5, 10$ , and  $15$  are used where the networks with  $N = 5$  and  $10$  are the same with those in Experiment 5.3.1. The target network of  $N = 15$  has also 15 sets of gene expression profiles with  $I = 3$  and  $T = 11$ . First, we added  $k\%$  Gaussian noise to all gene expression level points of  $N$  genes. The mean of the Gaussian noise is zero and the standard deviation equals to  $X_{exp,i,t} \times k\%$ .

We applied iTEA to estimate the parameters of S-system model. The function value of Eq. 5.6 reflecting the fitting quality of time-series data is used to evaluate the ability of the used optimization algorithm. However, the major concern is to obtain a correct

Table 5.8. The obtained S-system parameters with  $N = 10$ . The nine highlighted numbers indicated false-positive interqaction. No false-negative interaction is found.

$i$	$\alpha_i$	$g_{i1}$	$g_{i2}$	$g_{i3}$	$g_{i4}$	$g_{i5}$	$g_{i6}$	$g_{i7}$	$g_{i8}$	$g_{i9}$	$g_{i10}$
1	5.00	0	0	0	1.00	0	-2.00	0	0	0	0
2	9.94	0	0	1.00	0	0	0	0	-1.00	0	0
3	8.02	-1.00	0	0	-1.00	0	0	0	0	0	0
4	9.99	0	0	0	0	2.00	0	0	0	1.00	0
5	10.00	0	2.00	0	0	0	-1.00	0	0	0	0
6	5.01	0	0	0	0	0	0	0	0	2.00	-1.99
7	10.06	0	0	0	0	0	1.00	0	0	0	-1.00
8	4.98	1.00	-1.99	0	0	0	0	1.00	0	0	0
9	9.41	0	0	1.02	0	0	0	0	-2.02	-0.03	0
10	8.00	2.00	0	0	0	0	0	-1.00	0	0	0

$i$	$\beta_i$	$h_{i1}$	$h_{i2}$	$h_{i3}$	$h_{i4}$	$h_{i5}$	$h_{i6}$	$h_{i7}$	$h_{i8}$	$h_{i9}$	$h_{i10}$
1	10.00	2.00	0	0	0	0	0	0	0	0	0
2	9.22	0	2.05	0	0	0	0	-0.03	0	0	-0.06
3	10.07	0	0	2.00	0	0	0	0	0	0	0
4	10.00	0	0	0	2.00	0	0	0	0	0	0
5	9.98	0	0	0	0	2.00	0	0	0	0	0
6	9.39	0	0	0	0	0	2.02	0	0	0	0
7	10.09	0	0	0	0	0	-0.05	2.04	0	0.03	0
8	11.59	0	0.30	0	0	0	0	0	1.96	-0.28	0
9	9.09	0.04	0	0	0	0	0	0	0.13	2.13	0
10	10.01	0	0	0	0	0	0	0	0	0	2.00

Table 5.9. The fitness values in Eq. 5.4 for all subproblems with  $N = 10$ .

$i$	1	2	3	4	5
$f_i$	0.000001	0.000416	0.000019	0.000001	0.000004
$i$	6	7	8	9	10
$f_i$	0.003180	0.000547	0.000020	0.001786	0.000000

network structure with accurate parameter values. We define the true positive rate as sensitivity  $SN = TP/(TP + FN)$  where  $TP$  is true positive and  $FN$  is false negative; and the true negative rate as specificity  $SP = TN/(TN + FP)$  where  $TN$  is true negative and  $FP$  is false positive.

In the above-mentioned experiments, IGA is shown to be efficient enough for solving subproblems. To illustrate the effectiveness of  $R > 1$  runs in stage 1 and refinement of OSA in stage 2, we take the target model of  $N = 10$  with 5% Gaussian noise as an example. At first, IGA is performed one run ( $R = 1$ ) to solve each individual subproblem and then all the  $N = 10$  solutions to the 10 subproblems are combined as a final solution



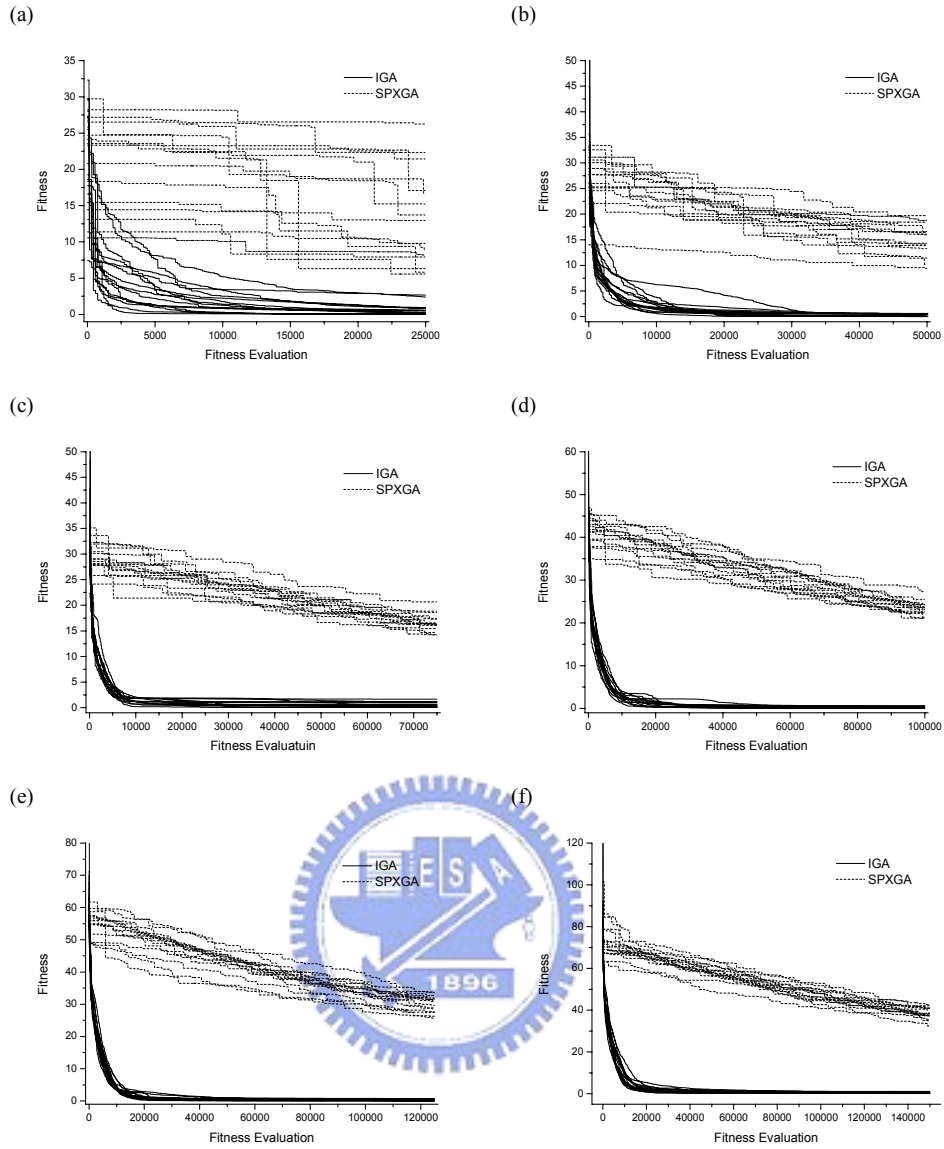


Figure 5.3. The convergence comparison between IGA and SPXGA using 30 independent runs. (a)  $N = 5$  (b)  $N = 10$  (c)  $N = 15$  (d)  $N = 20$  (e)  $N = 25$  (f)  $N = 30$ .

to the inference problem. On the other hand, a combined solution  $S$  is also obtained from stage 1 using IGA with  $R = 10$ . Figure 5.4 shows the distribution of the  $S$  and 30 final solutions. From Figure 5.4, it can be found that the best solution  $S_{best}$  of  $R = 1$  has values  $F = 6.42$ ,  $SN = 98\%$ , and  $SP = 82.7\%$ . The solution  $S$  has  $F = 6.17$ ,  $SN = 100\%$ , and  $SP = 82.25\%$ , which is slightly better than  $S_{best}$ . Therefore, it is effective to combine the best one of  $R > 1$  solutions for all individual subproblems. After performing OSA 30 independent runs to refine the solution  $S$ , the obtained model has values  $F = 4.70$ ,  $SN = 100\%$ , and  $SP = 82.52\%$  averagely. It reveals that OSA can effectively improve the model in terms of fitness value (referred to Table 5.11).

Table 5.10. T-test results for comparisons between IGA and SPXGA with various values of  $N$ .

$N$	5		10		15	
$N_{eval}$	25000		50000		75000	
Algorithm	IGA	SPXGA	IGA	SPXGA	IGA	SPXGA
Best	0.01578	4.65101	$2.84 \times 10^{-4}$	9.42883	0.02343	13.30884
Worst	2.65831	26.24824	0.72764	21.09678	1.64378	20.64385
Mean	0.53829	11.41142	0.24137	15.24435	0.49844	16.3107
Variance	0.4979	29.68118	0.06484	10.13658	0.18244	2.97019
t value	-11.25817		-25.56804		-47.24382	
p value	$4.20046 \times 10^{-12}$		$1.92315 \times 10^{-21}$		$5.45875 \times 10^{-29}$	
$N$	20		25		30	
$N_{eval}$	100000		125000		150000	
Algorithm	IGA	SPXGA	IGA	SPXGA	IGA	SPXGA
Best	0.01155	18.86678	0.01924	25.71603	0.06302	32.38494
Worst	1.00323	27.30871	0.76638	34.00673	1.1908	44.98229
Mean	0.30912	22.65516	0.32504	30.39126	0.58418	38.86353
Variance	0.05177	3.27567	0.05363	6.44214	0.11128	7.25008
t value	-64.91653		-65.25943		-76.31632	
p value	$5.91115 \times 10^{-33}$		$5.07875 \times 10^{-33}$		$5.56666 \times 10^{-35}$	

Table 5.11 shows the results of iTEA using artificial data with  $N = 5, 10,$  and  $15$  where 3% and 5% Gaussian noises are added. The stopping condition of OSA is to use 3000 iterations. OSA performed 30 independent runs using the same initial solution  $S$  obtained from the best solutions of all individual subproblems using IGA with  $R = 10$ . The simulation results show that IGA is good at solving subproblems and OSA can further refine the combined solution  $S$  having a relatively small value  $F(S)$ . Furthermore, iTEA can effectively solve the inference problems with the value of  $N$  as large as 15. For  $N = 5$  and 10, the average sensitivity performances are near or equal to 100%. For  $N = 15$ ,  $SN > 93\%$ . The specificity performances ranged from 54.05% to 87.13% seem not as good as the sensitivity performance from the aspect of  $SP$  value. By carefully examining the results and analyzing the inference performance, iTEA can often obtain a satisfactory S-system model, discussed below.

- 1) Because the Gaussian noise is added into the gene expression profiles, the original (true) values of S-system parameters may be not the best solution to fit the noisy gene expression profiles. Note that iTEA aims to find the S-system model which can best fit the noisy gene expression profiles. As a result, some small false-positive

interactions may be additionally occurred.

- 2) The specificity performance highly depends on the threshold value in using the structure skeletalizing technique ( $\delta_s = 0.03$  in this study and  $\delta_s = 0.1$  in [26]). When interpreting the interaction from the estimated values of S-system parameters for noisy data, it is better to filter the small interactions using a larger threshold value. If an additional threshold  $\delta_t = 0.1$  is used to filter the small interactions that their absolute values are smaller than 0.1 by assigning a value of zero to them, the sensitivity performance is unchanged and the specificity performance would be obviously enhanced, ranged from 79.10% to 92.33%.

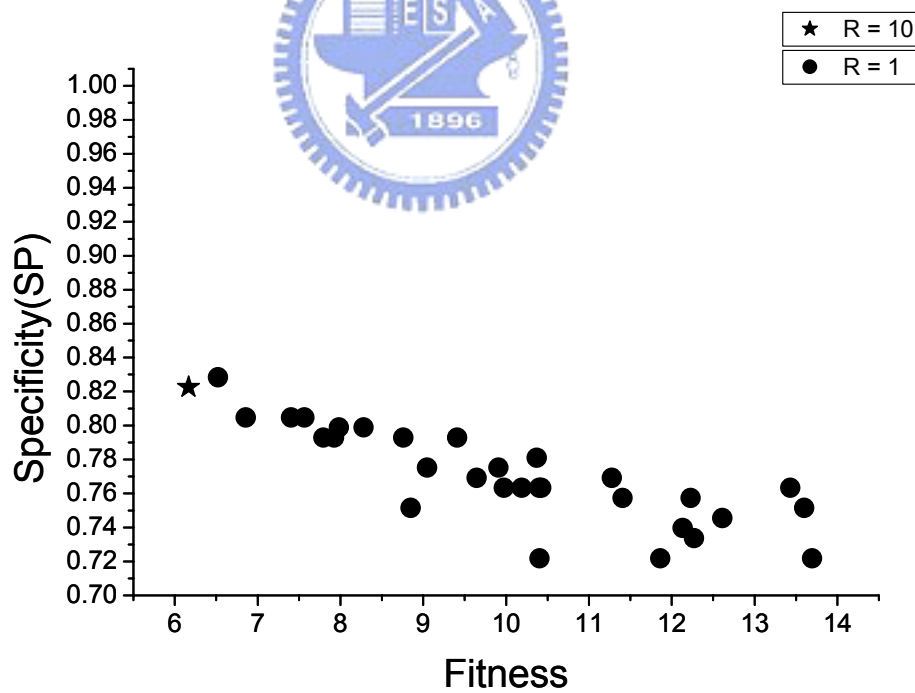
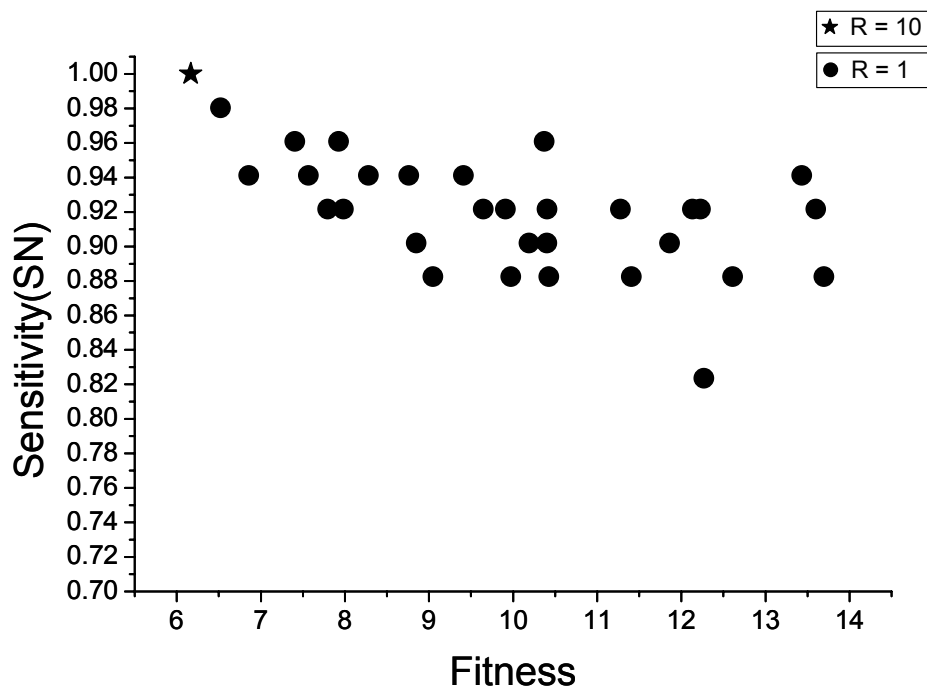
## 5.4 Conclusions for iTEA

S-system model has been considered suitable to characterize biochemical network systems and capable to analyze the regulatory system dynamics. Essentially, the inference of S-system models of genetic networks is a large-scale optimization problem consisting of  $2N(N + 1)$  parameters to be optimized where  $N$  is the number of genes in the genetic network. In this thesis, we propose an intelligent two-stage algorithm iTEA to search for an optimal solution to the reverse engineering problem for inference of genetic network architectures running on a single-processor PC. iTEA solves the optimization problem using a divide-and-conquer approach in each of two stages. At first, the original problem is decomposed into  $N$  individual  $2(N + 1)$  dimensional subproblems. In the first stage, an intelligent genetic algorithm (IGA) is used to solve the individual subproblems independently without further estimating gene expression levels of other genes. Our simulation demonstrated that the proposed IGA-based method is effective in solving the subproblems of inferring S-system models of genetic networks.

To compensate the disregard of estimating accurate gene expression levels of other genes from noisy data of gene expression, all the solutions to  $N$  subproblems are combined and then refined using an orthogonal simulated annealing algorithm (OSA) from the aspect of global optimization. In stage 2, OSA can effectively refine the combined solution quality where 3% and 5% Gaussian noises were added to gene expression profiles. From

simulation results, it has shown that the proposed algorithm iTEA performs well from noise-free and small-noise gene expression profiles. Our future work is to use iTEA to identify the dynamic pathway from actual gene expression profiles with measurement noise where biological knowledge is incorporated to improve solution quality.





(b)

Figure 5.4. The distribution of 30 solutions with  $R = 1$  and one solution with  $R = 10$  using IGA only without refinement of OSA from gene expression profiles of  $N = 10$  and 5% Gaussian noise. (a) Fitness value vs. Sensitivity (b) Fitness value vs. Specificity.

Table 5.11. Performance of the proposed algorithm with OSA is performed 30 runs.  $S$  is the combined initial solution of OSA.

Performance	( $N = 5, 3\%$ )	( $N = 5, 5\%$ )	( $N = 10, 3\%$ )	( $N = 10, 5\%$ )	( $N = 15, 3\%$ )	( $N = 15, 5\%$ )
$F(S)$	1.89	2.06	1.80	6.17	3.10	7.06
before OSA						
Best $F(S)$	1.82	1.94	1.51	4.57	3.01	7.03
after OSA						
Mean $F(S)$	1.82	1.95	1.52	4.70	3.05	7.05
after OSA						
Sensitivity	100%	99.28%	100%	100%	93.42%	94.74%
( $TP, FN$ )	(23.0, 0)	(22.83, 0.17)	(51.0, 0)	(51.0, 0)	(71.0, 5.0)	(72.0, 4.0)
Specificity	54.05%	59.37%	83.41%	82.52%	86.88%	87.13%
( $TN, FP$ )	(20.0, 17.0)	(21.97, 15.0)	(140.97, 28.0)	(139.47, 29.53)	(351.0, 53.0)	(352.0, 52.0)
Specificity	83.60%	79.10%	92.33%	87.69%	91.09%	90.84%
( $TN, FP$ )	(30.93, 6.07)	(29.27, 7.73)	(156.03, 12.97)	(148.20, 20.80)	(368.0, 36.0)	(367.0, 37.0)
Using $\delta_t = 0.1$						

# Chapter 6

## Conclusions

Microarray gene expression profiling technology is one of the most important research topics in cancer research or clinical diagnosis of disease. One goal in analyzing expression data is to determine how genes are expressed as a result of certain cellular conditions (e.g., how genes are expressed in diseased and healthy cells) [1]. Another goal is to determine how the expression of any particular gene might affect the expression of other genes in the same genetic network [2, 3, 4, 5].

To achieve the two objectives of microarray data analysis mentioned above, the most important issues in microarray data analysis are the gene expression data classification problem and the genetic networks inference problem. In this thesis, we proposed two IGA-based optimization algorithms to cope with these two major problems of microarray data analysis. Following are the introductions about the results and future works of our two proposed optimization methods for the two topics mentioned above.

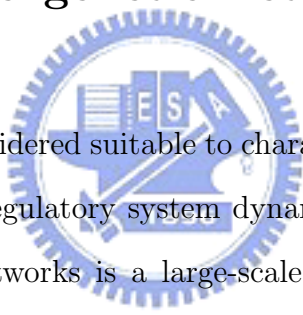
### 6.1 iGEC for the gene expression classification problems

For microarray gene expression classification problem, the important issue is that how to design an accurate, compact, and human-interpretable classifier is the major concern in this study. We proposed an interpretable gene expression classifier, named iGEC, for microarray data analysis. The design of iGEC includes almost all aspects related to the design of compact fuzzy rule-based classification systems: gene selection, rule selection, membership function tuning, consequent class determination, and certainty

grade tuning. Consequently, an efficient optimization algorithm IGA is used to solve the resultant optimization problem with a large number of parameters.

The superiority of the proposed iGEC was evaluated by computer simulation on eight data sets of gene expression. The experimental results reveal that: 1) the proposed method can obtain interpretable classifiers with an accurate and compact fuzzy rule base, compared with the existing fuzzy classifier [13]; 2) iGEC using few genes is worse than SVM but superior to  $k$ -NN, BNN, and PNN using thousands of genes in terms of accuracy; and 3) iGEC has high interpretabilities using 1.13 fuzzy rules of short length for representing one class averagely. Moreover, we extended the proposed iGEC to an interpretable scoring fuzzy classifier, named iSFC, which is able to effectively quantify the certainty grades of samples belonging to each class.

## 6.2 iTEA for the genetic networks inference problems



S-system model has been considered suitable to characterize biochemical network systems and capable to analyze the regulatory system dynamics. Essentially, the inference of S-system models of genetic networks is a large-scale optimization problem consisting of  $2N(N + 1)$  parameters to be optimized where  $N$  is the number of genes in the genetic network. This thesis proposes an intelligent two-stage evolutionary algorithm (iTEA) to solve the optimization problem using a divide-and-conquer strategy in each of two stages. The original problem can be decomposed into  $N$  individual  $2(N + 1)$  dimensional subproblems if the measurement noise is small. In stage 1, an intelligent genetic algorithm is used to solve the individual subproblems independently without further estimating gene expression levels of other genes. Our simulations demonstrated that the proposed IGA-based method is effective in solving the subproblem of inferring S-system models of genetic networks. In stage 2, OSA can refine the combined solution quality in terms of fitness value where 3% and 5% Gaussian noises were added. From simulation results, it has shown that the proposed algorithm iTEA performs well from noise-free and small-noise gene expression profiles. Our future work is to use iTEA to identify the dynamic pathway



from actual gene expression profiles with measurement noise where biological knowledge is incorporated to improve solution quality.



# Bibliography

- [1] C. Creighton and S. Hanash, “Mining gene expression databases for association rules,” *Bioinformatics*, vol. 19, no. 1, pp. 79–86, 2003.
- [2] H. Resson, R. Reynolds, and R. S. Varghese, “Increasing the efficiency of fuzzy logic-based gene expression data analysis,” *Physiological Genomics*, vol. 13, no. 2, pp. 107–117, 2003.
- [3] P. J. Woolf and Y. X. Wang, “A fuzzy logic approach to analyzing gene expression data,” *Physiological Genomics*, vol. 3, no. 1, pp. 9–15, 2000.
- [4] S. Kauffman, C. Peterson, B. Samuelsson, and C. Troein, “Random boolean network models and the yeast transcriptional network,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 100, no. 25, pp. 14 796–14 799, 2003.
- [5] M. Wahde and J. Hertz, “Coarse-grained reverse engineering of genetic regulatory networks,” *Biosystems*, vol. 55, no. 1-3, pp. 129–136, 2000.
- [6] Z. Bar-Joseph, “Analyzing time series gene expression data,” *Bioinformatics*, vol. 20, no. 16, pp. 2493–2503, 2004.
- [7] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein, “Cluster analysis and display of genome-wide expression patterns,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 95, no. 25, pp. 14 863–14 868, 1998.
- [8] 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 cerevisiae* by microarray hybridization,” *Molecular Biology of the Cell*, vol. 9, no. 12, pp. 3273–3297, 1998.

- [9] U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine, “Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 96, no. 12, pp. 6745–6750, 1999.
- [10] A. Ben-Dor, R. Shamir, and Z. Yakhini, “Clustering gene expression patterns,” *Journal of Computational Biology*, vol. 6, no. 3-4, pp. 281–297, 1999.
- [11] 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,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 96, no. 6, pp. 2907–2912, 1999.
- [12] A. Statnikov, C. F. Aliferis, I. Tsamardinos, D. Hardin, and S. Levy, “A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis,” *Bioinformatics*, vol. 21, no. 5, pp. 631–643, 2005.
- [13] S. A. Vinterbo, E. Y. Kim, and L. Ohno-Machado, “Small, fuzzy and interpretable gene expression based classifiers,” *Bioinformatics*, vol. 21, no. 9, pp. 1964–1970, 2005.
- [14] C. H. Q. Ding, “Unsupervised feature selection via two-way ordering in gene expression analysis,” *Bioinformatics*, vol. 19, no. 10, pp. 1259–1266, 2003.
- [15] B. Liu, Q. H. Cui, T. Z. Jiang, and S. D. Ma, “A combinational feature selection and ensemble neural network method for classification of gene expression data,” *BMC Bioinformatics*, vol. 5, 2004, 136.
- [16] X. Zhou and K. Z. Mao, “Ls bound based gene selection for dna microarray data,” *Bioinformatics*, vol. 21, no. 8, pp. 1559–1564, 2005.
- [17] X. X. Liu, A. Krishnan, and A. Mondry, “An entropy-based gene selection method for cancer classification using microarray data,” *BMC Bioinformatics*, vol. 6, 2005, 76.

- [18] L. P. Li, C. R. Weinberg, T. A. Darden, and L. G. Pedersen, “Gene selection for sample classification based on gene expression data: study of sensitivity to choice of parameters of the ga/knn method,” *Bioinformatics*, vol. 17, no. 12, pp. 1131–1142, 2001.
- [19] C. H. Ooi and P. Tan, “Genetic algorithms applied to multi-class prediction for the analysis of gene expression data,” *Bioinformatics*, vol. 19, no. 1, pp. 37–44, 2003.
- [20] J. Y. Li, H. Q. Liu, J. R. Downing, A. E. J. Yeoh, and L. S. Wong, “Simple rules underlying gene expression profiles of more than six subtypes of acute lymphoblastic leukemia (all) patients,” *Bioinformatics*, vol. 19, no. 1, pp. 71–78, 2003.
- [21] T. R. Hvidsten, A. Laegreid, and J. Komorowski, “Learning rule-based models of biological process from gene expression time profiles using gene ontology,” *Bioinformatics*, vol. 19, no. 9, pp. 1116–1123, 2003.
- [22] P. Brazhnik, A. de la Fuente, and P. Mendes, “Gene networks: how to put the function in genomics,” *Trends in Biotechnology*, vol. 20, no. 11, pp. 467–472, 2002.
- [23] P. D’Haeseleer, S. D. Liang, and R. Somogyi, “Genetic network inference: from co-expression clustering to reverse engineering,” *Bioinformatics*, vol. 16, no. 8, pp. 707–726, 2000.
- [24] N. Friedman, M. Linial, I. Nachman, and D. Pe’er, “Using bayesian networks to analyze expression data,” *Journal of Computational Biology*, vol. 7, no. 3-4, pp. 601–620, 2000.
- [25] J. Yu, V. A. Smith, P. P. Wang, A. J. Hartemink, and E. D. Jarvis, “Advances to bayesian network inference for generating causal networks from observational biological data,” *Bioinformatics*, vol. 20, no. 18, pp. 3594–3603, 2004.
- [26] E. O. Voit and J. Almeida, “Decoupling dynamical systems for pathway identification from metabolic profiles,” *Bioinformatics*, vol. 20, no. 11, pp. 1670–1681, 2004.

- [27] D. Husmeier, “Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic bayesian networks,” *Bioinformatics*, vol. 19, no. 17, pp. 2271–2282, 2003.
- [28] M. Zou and S. D. Conzen, “A new dynamic bayesian network (dbn) approach for identifying gene regulatory networks from time course microarray data,” *Bioinformatics*, vol. 21, no. 1, pp. 71–79, 2005.
- [29] S. Kikuchi, D. Tominaga, M. Arita, K. Takahashi, and M. Tomita, “Dynamic modeling of genetic networks using genetic algorithm and s-system,” *Bioinformatics*, vol. 19, no. 5, pp. 643–650, 2003.
- [30] S. Kimura, M. Hatakeyama, and A. Konagaya, “Inference of s-system models of genetic networks from noisy time-series data,” *Chem-Bio Informatics Journal*, vol. 4, no. 1, pp. 1–14, 2004.
- [31] S. Kimura, K. Ide, A. Kashihara, M. Kano, M. Hatakeyama, R. Masui, N. Nakagawa, S. Yokoyama, S. Kuramitsu, and A. Konagaya, “Inference of s-system models of genetic networks using a cooperative coevolutionary algorithm,” *Bioinformatics*, vol. 21, no. 7, pp. 1154–1163, 2005.
- [32] Y. Maki, T. Ueda, M. Okamoto, N. Uematsu, K. Inamura, K. Uchida, Y. Takahashi, and Y. Eguchi, “Inference of genetic network using the expression profile time course data of mouse p19 cells,” *Genome Informatics*, vol. 13, pp. 382–383, 2002.
- [33] R. Morishita, H. Imade, I. Ono, N. Ono, and M. Okamoto, “Finding multiple solutions based on an evolutionary algorithm for inference of genetic networks by s-system,” in *The Congress on Evolutionary Computation, CEC*, vol. 1, Canberra, Australia, 2003, pp. 615–622.
- [34] R. Thomas, S. Mehrotra, E. T. Papoutsakis, and V. Hatzimanikatis, “A model-based optimization framework for the inference on gene regulatory networks from dna array data,” *Bioinformatics*, vol. 20, no. 17, pp. 3221–3235, 2004.

- [35] D. Tominaga, N. Koga, and M. Okamoto, “Efficient numerical optimization algorithm based on genetic algorithm for inverse problem,” in *Genetic and Evolutionary Computation Conference, GECCO*, Las Vegas, Nevada, USA, 2000, pp. 251–258.
- [36] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley Publishing Company, 1989.
- [37] S. Y. Ho, H. M. Chen, S. J. Ho, and T. K. Chen, “Design of accurate classifiers with a compact fuzzy-rule base using an evolutionary scatter partition of feature space,” *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, vol. 34, no. 2, pp. 1031–1044, 2004.
- [38] S. Y. Ho, L. S. Shu, and J. H. Chen, “Intelligent evolutionary algorithms for large parameter optimization problems,” *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 6, pp. 522–541, 2004.
- [39] S. B. Cho, “Exploring features and classifiers to classify gene expression profiles of acute leukemia,” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 16, no. 7, pp. 831–844, 2002.
- [40] P. D. Wasserman, *Advanced Methods in Neural Computing*. New York: Van Nostrand Reinhold, 1993.
- [41] G. M. Sun, X. Y. Dong, and G. D. Xu, “Tumor tissue identification based on gene expression data using dwt feature extraction and pnn classifier,” *Neurocomputing*, vol. 69, no. 4-6, pp. 387–402, 2006.
- [42] C. J. Huang and W. C. Liao, “Application of probabilistic neural networks to the class prediction of leukemia and embryonal tumor of central nervous system,” *Neural Processing Letters*, vol. 19, no. 3, pp. 211–226, 2004.
- [43] C. J. Huang, “Class prediction of cancer using probabilistic neural networks and relative correlation metric,” *Applied Artificial Intelligence*, vol. 18, no. 2, pp. 117–128, 2004.

- [44] J. Sim, S. Y. Kim, and J. Lee, “Prediction of protein solvent accessibility using fuzzy k-nearest neighbor method,” *Bioinformatics*, vol. 21, no. 12, pp. 2844–2849, 2005.
- [45] T. Li, C. L. Zhang, and M. Ogihara, “A comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression,” *Bioinformatics*, vol. 20, no. 15, pp. 2429–2437, 2004.
- [46] V. N. Vapnik, “An overview of statistical learning theory,” *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 988–999, 1999.
- [47] T. S. Furey, N. Cristianini, N. Duffy, D. W. Bednarski, M. Schummer, and D. Hausler, “Support vector machine classification and validation of cancer tissue samples using microarray expression data,” *Bioinformatics*, vol. 16, no. 10, pp. 906–914, 2000.
- [48] Y. Lee and C. K. Lee, “Classification of multiple cancer types by tip multicategory support vector machines using gene expression data,” *Bioinformatics*, vol. 19, no. 9, pp. 1132–1139, 2003.
- [49] Y. H. Wang, F. S. Makedon, J. C. Ford, and J. Pearlman, “Hykgene: a hybrid approach for selecting marker genes for phenotype classification using microarray gene expression data,” *Bioinformatics*, vol. 21, no. 8, pp. 1530–1537, 2005.
- [50] J. Yen, “Fuzzy logic - a modern perspective,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 1, pp. 153–165, 1999.
- [51] H. Ishibuchi, T. Nakashima, and T. Murata, “Three-objective genetics-based machine learning for linguistic rule extraction,” *Information Sciences*, vol. 136, no. 1-4, pp. 109–133, 2001, sp. Iss. SI.
- [52] C. C. Wu, H. C. Huang, H. F. Juan, and S. T. Chen, “Genenetwork: an interactive tool for reconstruction of genetic networks using microarray data,” *Bioinformatics*, vol. 20, no. 18, pp. 3691–3693, 2004.
- [53] S. Kauffman, *The Origin of Order, Self-Organization and Selection in Evolution*. New York: Oxford University Press, 1993.

- [54] S. Liang, S. Fuhrman, and R. Somogyi, “Reveal, a general reverse engineering algorithm for inference of genetic network architectures,” in *Proc. of the Pacific Symposium on Biocomputing*, vol. 3, Hawaii, 1998, pp. 18–29.
- [55] T. Akutsu, S. Miyano, and S. Kuhura, “Identification of genetic networks from a small number of gene expression patterns under the boolean network model,” in *Proc. of the Pacific Symposium on Biocomputing*, vol. 4, Hawaii, 1999, pp. 17–28.
- [56] T. Akutsu, S. Miyano, and S. Kuhara, “Algorithms for identifying boolean networks and related biological networks based on matrix multiplication and fingerprint function,” *Journal of Computational Biology*, vol. 7, no. 3-4, pp. 331–343, 2000.
- [57] P. D’Haeseleer, X. Wen, S. Fuhrman, and R. Somogyi, “Linear modeling of mrna expression levels during cns development and injury,” in *Proc. of the Pacific Symposium on Biocomputing*, vol. 4, Hawaii, 1999, pp. 41–52.
- [58] P. D’Haeseleer, “Reconstructing gene networks from large scale gene expression data,” Ph.D. dissertation, University of New Mexico, 2000.
- [59] H. Szu and R. Hartley, “Fast simulated annealing,” *Physics Letters*, vol. 122, pp. 157–162, 1987.
- [60] A. Dey, *Orthogonal Fractional Factorial Designs*. New York: Wiley, 1985.
- [61] A. Hedayat, N. Sloane, and J. Stufken, *Orthogonal Arrays: Theory and Applications*. New York: Springer-Verlag, 1999.
- [62] T. Bagchi, *Taguchi Methods Explained: Practical Steps to Robust Design*. Prentice-Hall India Ltd, 1993.
- [63] S. J. Ho, S. Y. Ho, and L. S. Shu, “Osa: Orthogonal simulated annealing algorithm and its application to designing mixed  $h_2/h_\infty$  optimal controllers,” *IEEE Transactions on Systems Man and Cybernetics Part a-Systems and Humans*, vol. 34, no. 5, pp. 588–600, 2004.



- [64] Z. Michalewicz, D. Dasgupta, R. G. Leriche, and M. Schoenauer, “Evolutionary algorithms for constrained engineering problems,” *Computers and Industrial Engineering*, vol. 30, no. 4, pp. 851–870, 1996.
- [65] S. L. Pomeroy, P. Tamayo, M. Gaasenbeek, L. M. Sturla, M. Angelo, M. E. McLaughlin, J. Y. H. Kim, L. C. Goumnerova, P. M. Black, C. Lau, J. C. Allen, D. Zagzag, J. M. Olson, T. Curran, C. Wetmore, J. A. Biegel, T. Poggio, S. Mukherjee, R. Rifkin, A. Califano, G. Stolovitzky, D. N. Louis, J. P. Mesirov, E. S. Lander, and T. R. Golub, “Prediction of central nervous system embryonal tumour outcome based on gene expression,” *Nature*, vol. 415, no. 6870, pp. 436–442, 2002.
- [66] C. L. Nutt, D. R. Mani, R. A. Betensky, P. Tamayo, J. G. Cairncross, C. Ladd, U. Pohl, C. Hartmann, M. E. McLaughlin, T. T. Batchelor, P. M. Black, A. von Deimling, S. L. Pomeroy, T. R. Golub, and D. N. Louis, “Gene expression-based classification of malignant gliomas correlates better with survival than histological classification,” *Cancer Research*, vol. 63, no. 7, pp. 1602–1607, 2003.
- [67] M. A. Shipp, K. N. Ross, P. Tamayo, A. P. Weng, J. L. Kutok, R. C. T. Aguiar, M. Gaasenbeek, M. Angelo, M. Reich, G. S. Pinkus, T. S. Ray, M. A. Koval, K. W. Last, A. Norton, T. A. Lister, J. Mesirov, D. S. Neuberg, E. S. Lander, J. C. Aster, and T. R. Golub, “Diffuse large b-cell lymphoma outcome prediction by gene expression profiling and supervised machine learning,” *Nature Medicine*, vol. 8, no. 1, pp. 68–74, 2002.
- [68] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander, “Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring,” *Science*, vol. 286, no. 5439, pp. 531–537, 1999.
- [69] S. A. Armstrong, J. E. Staunton, L. B. Silverman, R. Pieters, M. L. de Boer, M. D. Minden, S. E. Sallan, E. S. Lander, T. R. Golub, and S. J. Korsmeyer, “Mll translocations specify a distinct gene expression profile that distinguishes a unique leukemia,” *Nature Genetics*, vol. 30, no. 1, pp. 41–47, 2002.

- [70] A. Bhattacharjee, W. G. Richards, J. Staunton, C. Li, S. Monti, P. Vasa, C. Ladd, J. Beheshti, R. Bueno, M. Gillette, M. Loda, G. Weber, E. J. Mark, E. S. Lander, W. Wong, B. E. Johnson, T. R. Golub, D. J. Sugarbaker, and M. Meyerson, “Classification of human lung carcinomas by mrna expression profiling reveals distinct adenocarcinoma subclasses,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 98, no. 24, pp. 13 790–13 795, 2001.
- [71] D. Singh, P. G. Febbo, K. Ross, D. G. Jackson, J. Manola, C. Ladd, P. Tamayo, A. A. Renshaw, A. V. D’Amico, J. P. Richie, E. S. Lander, M. Loda, P. W. Kantoff, T. R. Golub, and W. R. Sellers, “Gene expression correlates of clinical prostate cancer behavior,” *Cancer Cell*, vol. 1, no. 2, pp. 203–209, 2002.
- [72] J. Khan, J. S. Wei, M. Ringner, 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*, vol. 7, no. 6, pp. 673–679, 2001.
- [73] A. Brazma, H. Parkinson, U. Sarkans, M. Shojatalab, J. Vilo, N. Abeygunawardena, E. Holloway, M. Kapushesky, P. Kemmeren, G. G. Lara, A. Oezcimen, P. Rocca-Serra, and S. A. Sansone, “Arrayexpress - a public repository for microarray gene expression data at the ebi,” *Nucleic Acids Research*, vol. 31, no. 1, pp. 68–71, 2003.
- [74] M. Friberg, P. von Rohr, and G. Gonnet, “Scoring functions for transcription factor binding site prediction,” *BMC Bioinformatics*, vol. 6, 2005, 84.
- [75] X. S. Liu, D. L. Brutlag, and J. S. Liu, “An algorithm for finding protein-dna binding sites with applications to chromatin-immunoprecipitation microarray experiments,” *Nature Biotechnology*, vol. 20, no. 8, pp. 835–839, 2002.
- [76] J. Murvai, K. Vlahovicek, and S. Pongor, “A simple probabilistic scoring method for protein domain identification,” *Bioinformatics*, vol. 16, no. 12, pp. 1155–1156, 2000.
- [77] S. T. Jensen and J. S. Liu, “Biooptimizer: a bayesian scoring function approach to motif discovery,” *Bioinformatics*, vol. 20, no. 10, pp. 1557–1564, 2004.

- [78] S.-Y. Ho, C.-H. Hsieh, K.-W. Chen, H.-L. Huang, H.-M. Chen, and S.-J. Ho, “Scoring method for tumor prediction from microarray data using an evolutionary fuzzy classifier,” in *Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Singapore, 2006.
- [79] D. Thierry, A. Huerta, E. Perez-Rueda, and J. Collado-Vides, “From specific gene regulation to genomic networks: a global analysis of transcriptional regulation in escherichia coli,” *BioEssays*, vol. 20, no. 5, pp. 433–440, 1998.
- [80] S. Wei, T. Sun, and R. Wesel, “Quasi-convexity and optimal binary fusion for distributed detection with identical sensors in generalized gaussian noise,” *IEEE Trans. Information Theory*, vol. 47, no. 1, pp. 446–450, 2001.

