An Evolutionary Approach for Gene Expression Patterns

Huai-Kuang Tsai, Jinn-Moon Yang, Yuan-Fang Tsai, and Cheng-Yan Kao

***Abstract*—This study presents an evolutionary algorithm, called a heterogeneous selection genetic algorithm (HeSGA), for analyzing the patterns of gene expression on microarray data. Microarray technologies have provided the means to monitor the expression levels of a large number of genes simultaneously. Gene clustering and gene ordering are important in analyzing a large body of microarray expression data. The proposed method simultaneously solves gene clustering and gene-ordering problems by integrating global and local search mechanisms. Clustering and ordering information is used to identify functionally related genes and to infer genetic networks from immense microarray expression data. HeSGA was tested on eight test microarray datasets, ranging in size from 147 to 6221 genes. The experimental clustering and visual results indicate that HeSGA not only ordered genes smoothly but also grouped genes with similar gene expressions. Visualized results and a new scoring function that references predefined functional categories were employed to confirm the biological interpretations of results yielded using HeSGA and other methods. These results indicate that HeSGA has potential in analyzing gene expression patterns.**

***Index Terms*—Clustering, genetic algorithm (GA), gene clustering, gene expression, gene ordering, heterogeneous pairing selection (HpS), microarray.**

NOMENCLATURE

GA	Genetic algorithm.
TSP	Traveling salesman problem.
HeSGA	Heterogeneous pairing selection genetic algorithm.
EAX	Edge assembly crossover.
NJ	Neighbor-joining mutation.
HpS	Heterogeneous pairing selection.
MIPS	Munich information center for protein sequence.
π	Gene order.
$D(g_{\pi_i}, g_{\pi_{i+1}})$	Distance between two genes g_{π_i} and $g_{\pi_{i+1}}$.
$\mathrm{fit}(\pi)$	Sum of distances between pairs of adjacent genes in a linear ordering π.
$\mathrm{cat}(\pi)$	Scoring function to estimate the biological significance by referencing MIPS functional categories.

Manuscript received September 25, 2003; revised December 12, 2003.

H.-K. Tsai is with the Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan. R.O.C. (e-mail: d7526010@csie.ntu.edu.tw).

J.-M. Yang is with the Department of Biological Science and Technology, Institute of Bioinformatics, National Chiao Tung University, Hsinchu 30050, Taiwan, R.O.C. (e-mail: moon@cc.nctu.edu.tw).

Y.-F. Tsai is with the Department of Social Studies Education, National Taipei Teachers College, Taipei 106, Taiwan, R.O.C. (e-mail: tyf.ts@msa.hinet.net).

C.-Y. Kao is with the Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan. R.O.C., and also with the Bioinformatics Center, National Taiwan University, Taipei 106, Taiwan, R.O.C. (e-mail: cykao@csie.ntu.edu.tw).

Digital Object Identifier 10.1109/TITB.2004.826713

I. INTRODUCTION

THE DEVELOPMENT of microarray technologies has enabled the monitoring of the expression levels of many genes simultaneously. Advances in microarray technologies have resulted in a significant increase in the amount of biological data available. Given thousands of genes and hundreds of experiments, a very large number of gene expression profiles must be analyzed.

Cluster analysis and displays of gene expression patterns are considered to be useful tools in analyzing a large amount of microarray data [1], [2]. Clustering methods group genes with similar patterns of expression [3]. The clustering results are then ordered linearly for display. Such clustering and ordering of gene expression information is the basis of identifying functionally related genes [4], [5] and inferring genetic networks [6], [7].

Clustering methods can be broadly divided into hierarchical and nonhierarchical clustering approaches. Hierarchical clustering approaches, which are extensively used [1], [8]–[12], group gene expressions into trees of clusters. They start with singleton sets and merge all genes until all nodes belong to only one set. The agglomerative nature of such hierarchical clustering methods may cause genes to be grouped according to local decisions [4], [13]. Nonhierarchical clustering approaches, such as k means [14], self-organizing map (SOM) [4], Bayesian clustering [15], CAST [16], and CLICK [17], separate genes into groups according to the degree of similarity (as quantified by Euclidian distances, Pearson correlation) among genes. The relationships among the genes in a particular cluster generated by nonhierarchical clustering methods are lost.

A gene-ordering method determines an order in which to display smoothly the clustered genes. Finding an optimal order of genes is an nondeterministic polynomial time complete (NP-complete) problem [18], so some heuristic methods have been developed to generate gene orders [1], [9]. Bar-Joseph *et al.* [19] further applied dynamic programming to flip internal nodes and Herrero *et al.* [20] used neural networks to reorder the leaves in a hierarchical solution. These ordering methods sometimes stick at local minima.

Currently, most tools of analyzing microarray data separately executed gene clustering and gene ordering [1], [9], [13]. These tools often applied local search or greedy methods to analyze

gene expression data. In this study, we proposed a GA, called HeSGA, integrating local and global search mechanisms, to simultaneously solve gene clustering and gene ordering in analyzing microarray data.

The GA, introduced by Holland [21], was first used to design and implement robust adaptive systems. DeJong [22] considered GAs in relation to function optimization and Goldberg [23] considered them in relation to gas pipeline operations. GAs are inspired by Darwin's theory of evolution. GAs solve problems in an evolutionary fashion. A GA begins with a set of solutions represented as chromosomes called a population. Genetic operators, such as crossover and mutation operators, are applied to solutions in one population to generate new solutions. Solutions are selected to form new generations according to their fitness.

GAs have been successfully applied to optimization problems and some biological problems [24], [25]. However, general problem-independent GAs are normally not efficient in solving some specific problems. Numerous approaches, such as developing problem-specified operators [26], incorporating local searches [27], and maintaining the diversity of the population [28], have been proposed to improve GAs used to solve particular problems.

In this study, two key mechanisms are incorporated into HeSGA: 1) incorporating two problem-specific genetic operators, including local and global search mechanisms and 2) maintaining the diversity of population. The results of the authors' earlier work have demonstrated that these mechanisms are effective in solving continuous optimization problems in some fields [29]–[31]. The main difference in methodology between the present work and our previous studies is the addition of two problem-specific operators for efficiently optimizing the clustering and ordering of genes.

The proposed method was tested on eight test microarray datasets, ranging in size from 147 to 6221 genes. The experiments revealed that the results were comparable with those obtained by other methods. A new scoring function, referencing predefined functional categories, was also investigated to evaluate the performance of the analytical tools. The score of the gene order obtained by the proposed HeSGA is strongly related to the real biological classes and is consistent with the biological meanings. The experimental clustering and visual results indicate that HeSGA can order genes smoothly and group genes with similar expressions.

II. Problem Definition

The spotted DNA microarray, developed at Pat Brown's Laboratory in Stanford University [33], was developed especially for assaying gene expression. DNA microarrays are broadly applied in many biological applications, such as detecting mutation [34], [35], molecular evolution studies, molecular diagnosis [36], [37], functional genomics [38], [39], genetic mapping [40], and others. Gene clustering and gene ordering are important in analyzing very large bodies of microarray expression data. In clustering and ordering genes in an analysis of gene expression data, a linear ordering of genes is sought, such that genes with similar expression profiles are close to each other. An optimal gene order, a minimum sum of distances between pairs of adjacent genes in a linear ordering $\pi = (g_{\pi_1}, g_{\pi_2}, \ldots, g_{\pi_M})$ can be formulated as [18]

$$\text{fit}(\pi) = \sum_{i=1}^{M-1} D(g_{\pi_i}, g_{\pi_{i+1}}) \tag{1}$$

where M is the number of genes and $D(g_{\pi_i}, g_{\pi_{i+1}})$ is the distance between two genes g_{π_i} and $g_{\pi_{i+1}}$. In this study, the *centered Pearson correlation* is used to specify the distance $D(g_{\pi_i}, g_{\pi_{i+1}})$. For convenience of description, let $X = x_1, x_2, \ldots, x_k$ and $Y = y_1, y_2, \ldots, y_k$ be the expression levels of the two genes in terms of log-transformed data obtained over a series of k experiments. The distance between X and Y is

$$D(X, Y) = 1 - d_{X,Y} \tag{2}$$

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

$$d_{X,Y} = \frac{1}{k} \sum_{i=1}^{k} \left(\frac{x_i - \overline{X}}{\sigma_X} \right) \left(\frac{y_i - \overline{Y}}{\sigma_Y} \right) \tag{3}$$

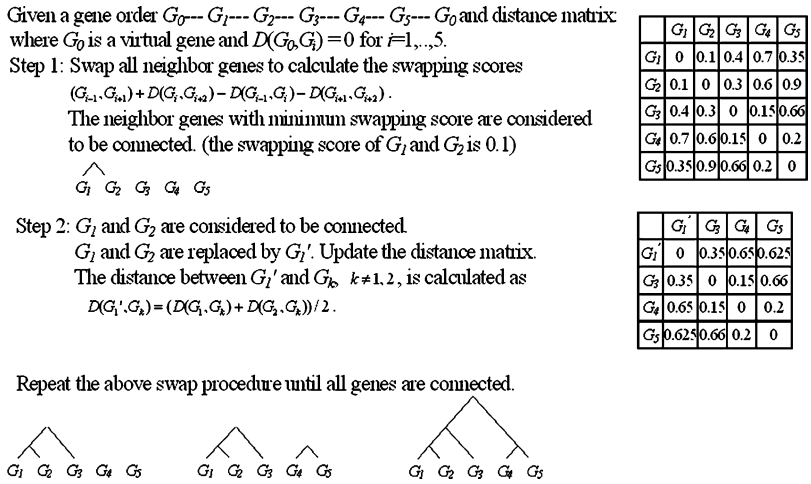
where $\overline{X}$ and σ_X are the mean and standard derivation of the gene X, respectively. Now

$$\sigma_X = \sqrt{\frac{1}{k} \sum_{i=1}^{k} (x_i - \overline{X})^2}. \tag{4}$$

The value of $d_{X,Y}$ is between -1 and 1 and the value of $D(X, Y)$ in (1) ranges from zero to two. Intuitively, if two genes are similar, then the distance between them is short.

The formula (1) for optimal gene ordering is the same as that used to determine the shortest Hamiltonian path through a set of M cities on the condition that each city is visited only once. This problem is equivalent to an $(M+1)$-city TSP with an additional city whose distance from all the other cities is zero. TSP is well known to be NP-complete [41]. This paper presents a robust GA to solve this problem.

Ordered genes are flexible and can be easily transformed into various clusters and hierarchical trees according to the conditions or requirements [42]. Users and biological experts can use the information provided by the ordering to identify clusters and interpret those microarray data. Fig. 1 presents an example of the construction of a hierarchical tree by the proposed procedure [42]. Given the order of genes $\{G_1, \ldots, G_n\}$, the score associated with swapping two adjacent genes is defined as $D(G_{i-1}, G_{i+1}) + D(G_i, G_{i+2}) - D(G_{i-1}, G_i) - D(G_{i+1}, G_{i+2})$. The two neighboring genes with the minimum score of *all pairs of genes* are connected. Assume that G_j and G_{j+1} are connected; these two genes are replaced by G'_j, and the gene order is updated as $\{G_1, \ldots, G_{j-1}, G'_j, G_{j+2}, \ldots, G_n\}$. The distance between G'_j and G_k, $k \neq j, j+1$, is given by $D(G'_j, G_k) = (D(G_j, G_k) + D(G_{j+1}, G_k))/2$. Repeat the substutions until all genes are connected in a single tree.



	G_1	G_2	G_3	G_4	G_5
G_1	0	0.1	0.4	0.7	0.35
G_2	0.1	0	0.3	0.6	0.9
G_3	0.4	0.3	0	0.15	0.66
G_4	0.7	0.6	0.15	0	0.2
G_5	0.35	0.9	0.66	0.2	0

	G_1'	G_3	G_4	G_5
G_1'	0	0.35	0.65	0.625
G_3	0.35	0	0.15	0.66
G_4	0.65	0.15	0	0.2
G_5	0.625	0.66	0.2	0

Fig. 1. Steps and an example of constructing a hierarchical tree from a given gene order. This approach [42] is able to transform an order gene into different clusters.

III. Heterogeneous Selection Genetic Algorithm

The HeSGA consists of a new NJ, an EAX [26], a new HpS, and a family competition. The EAX and NJ are used to preserve adjacent genes with similar expressions globally and locally, respectively. The family competition and the HpS maintain the diversity of the population. Notably, a good edge is one that connects adjacent genes with similar expression profiles.

The NJ mutation, a local search mechanism, can align two genes with similar expression levels. The EAX, a global search mechanism, based on graphical theories, has been considered to be an effective crossover operator for TSPs [43]–[46]. The EAX and the NJ mutation generate offspring by preserving good edges from parents and adding new edges heuristically. The HpS selects two parents according to similarities among edges in a population for applying crossover operators to reduce the prematurity effects. Finally, the family competition, derived from $(1+\lambda)$-ES [47] and the Lin–Kernigan heuristic, is the local search procedure for the optimal solution.

Fig. 2 depicts the main steps of the HeSGA. The initial population includes N chromosomes. Each chromosome s_i represents a gene order π_i, where $1 \leq i \leq N$ and N is the size of the population. Consider M genes; the chromosome s_i is represented as

$$s_i = (g_{\pi_{i1}}, g_{\pi_{i2}}, \ldots, g_{\pi_{iM}}). \tag{5}$$

After the fitness is evaluated from (1), each individual in the population sequentially becomes the "family father (s_i)" such that by HpS can select another individual s_j and produce some offspring by the EAX and family competition. The individual with the lowest fitness value among s_i and its offspring becomes the intermediate solution. The NJ then refines the intermediate solution to generate a child (c_i). Each individual in the population sequentially undergoes the above steps to generate its child. These N solutions $(c_1, \ldots, c_N)$ become the population of the next generation.

The algorithm terminates when one of following criteria is satisfied: 1) the maximum preset search time is exhausted; 2) all individuals of a population are the same; or 3) all of the children generated over five generations are poorer than their parents.

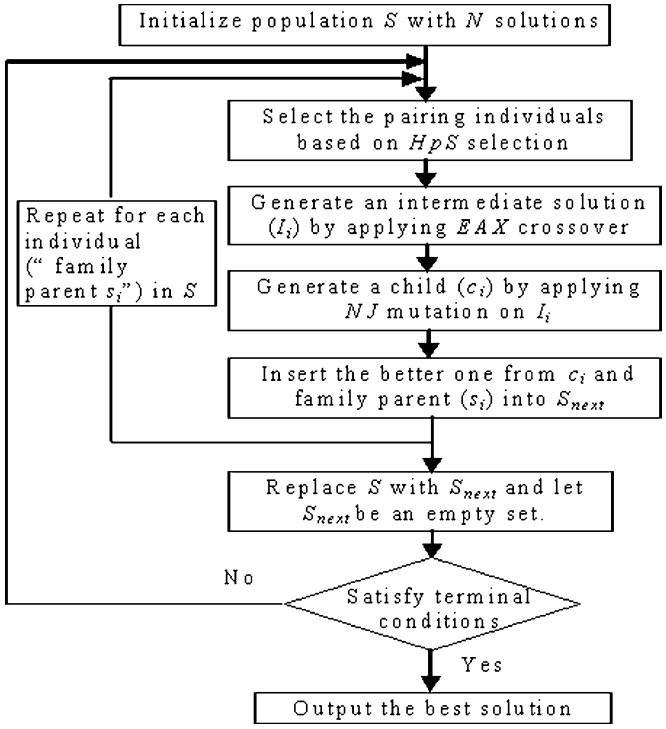


Fig. 2. Overview of the proposed GA (HeSGA).

Notably, both crossover and mutation rates are one. HpS, EAX, and NJ are briefly described below.

A. HpS

The HpS selects each "family father (s_i)" and another individual from the current population according to the edge similarity to which the crossover operators, such as EAX, are applied. The HpS reduces the probability of becoming trapped at a local optimum by avoiding incest. The formulation and implementation of the HpS is as follows. Let $\{s_1, s_2, \ldots, s_N\}$ be the current population, $E(s_i)$ be the set of the edges of the solution s_i, and $||E(s_i)||$ be the number of edges of $E(s_i)$. The number of identical edges $||T_{i,j}||$ of two individuals (s_i and s_j) is $||T_{i,j}|| = ||E(s_i) \cap E(s_j)||$.

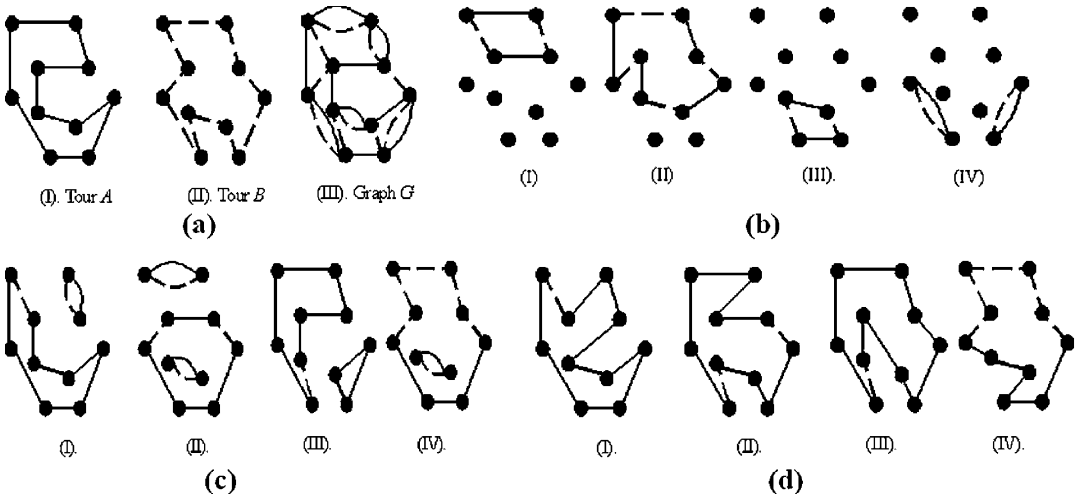


Fig. 3. Steps and an example of the EAX crossover operator. (a) Given tours A, B, the graph G is obtained by overlapping tours A and B. (b) The instances of the division of undirected edges into AB cycles on G. (I), (II), and (III) are effective AB cycles, while the AB cycle in (IV) is ineffective. (c) Four examples of EAX individually apply some AB cycles on A. (I), (II), and (III) are the results of applying the AB cycles shown in (b-I), (b-II), and (b-III), respectively. (c-IV) is the result of applying both (b-I) AB cycle and (b-II) AB cycle on A. (d) Four valid individuals are derived from (c) by applying a greedy method.

For each individual s_i, let t_i be the average number of identical edges shared by s_i and the other individual in the population, where $t_i = 1/(N-1)\sum_{j=1,j\neq i}^{N} ||T_{i,j}||$, and N is the size of the population. The EAX operator is applied to an individual s_i, and another individual s_j with $||T_{i,j}|| \leq t_i$, selected by the HpS. HpS randomly chooses individuals s_j until the criterion $||T_{i,j}|| \leq t_i$ is met.

In the practical implementation, another method was used to calculate t_i because the temporal complexity of obtaining all t_i by calculating $||T_{i,j}||$ is $O(N^2M^2)$. At the beginning of each generation $F(e)$, the number of times edge e appears in the current population, is calculated according to $e \in \{E(s_1) \cup E(s_2) \cup \cdots \cup E(s_N)\}$. The sum of the values of $||T_{i,j}||$ over all s_i in the population, is

$$\sum_{j=1,j\neq i}^{N} ||T_{i,j}|| = \sum_{j=1}^{N} ||T_{i,j}|| - ||T_{i,i}|| = \sum_{e\in E(s_i)} F(e) - M = \sum_{e\in E(s_i)} (F(e)-1).$$

Therefore, t_i becomes

$$t_i = \frac{1}{N-1} \sum_{e\in E(s_i)} (F(e)-1).$$

Hence, all t_i values can be calculated in $O(NM)$, by looking up the precalculated table. The EAX crossover also uses $F(e)$, so little extra effort is required to calculate t_i.

B. EAX

The EAX has two important characteristics: it preserves parents' edges in a novel way and adds new edges using a greedy method, analogous to the method of constructing a minimal spanning tree. The EAX is briefly described here. Two individuals, A and B, are selected as parents. The EAX first merges A and B into a single graph, G [Fig. 3(a)]. The EAX traverses G to generate various AB cycles by alternately selecting edges from parents A and B. Fig. 3(b) depicts some examples of AB cycles. According to the heuristic and random selection rules, some AB cycles are selected to generate a quasi-solution that includes some disjointed subtours [Fig. 3(c)].

The EAX then follows a greedy method to merge these subtours into a valid solution [Fig. 3(d)]. This solution is returned if the fitness of this solution exceeds that of its parents. Otherwise, the procedure is repeated until a solution that is fitter than both A and B is obtained or L children are generated where L is the family competition length. EAX is used to preserve adjacent genes with similar global expressions.

C. NJ Mutation

Fig. 4(a) presents the steps of the NJ mutation. The NJ mutation generates L children from a start solution s_i given a family competition length L. The NJ mutation involves the following steps to generate a child. First, a city c is randomly selected from s_i. With equal probability, another city c' is either randomly selected from the geometrically nearest three neighbors to the city c or the neighboring cities of c in another individual, which is randomly selected from the population. If the edge between cities c and c' is absent from s_i, then cities c and c' are reconnected and four types, presented in Fig. 4(b), are generated. Of these four candidates and s_i, the one with the best objective value is selected as the parent for the following loop of the NJ mutation.

The invert operator was used to connect the cities c and c' for Types I and II in Fig. 4(b). For Types III and IV, a greedy method is used to merge two disjoint subtours into a valid solution. The greedy method is as follows. Let v_i represent a city; let (v_i, v_j), $i \neq j$, represent an edge of length $w(v_i, v_j)$. Moreover, let (v_r, v_{r+1}) and (v_s, v_{s+1}) be the edges of the subtours G_r and G_s, respectively. A pair of new edges (v_r, v_{s+1}) and (v_s, v_{r+1}) is determined to connect these two subtours, G_r and G_s, into a valid tour by maximizing

$$w(v_r, v_{r+1}) + w(v_s, v_{s+1}) - w(v_r, v_{s+1}) - w(v_s, v_{r+1}) \quad \forall r, s;\ r \in G_r \text{ and } s \in G_s.$$

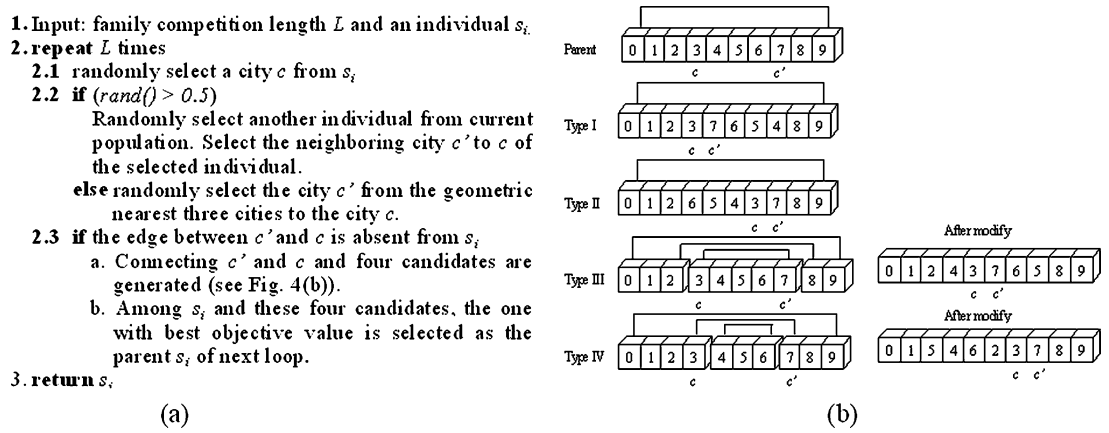


Fig. 4. Pseudocode and examples of the NJ mutation operator. (a) The NJ mutation algorithm. (b) Cities c and c' are reconnected and four candidates are generated. The invert operator is applied to connect the cities c and c'. For type III and type IV, a greedy method is used to merge two subtours into a valid solution.

TABLE I
RESULTS OF HESGA TESTED ON 15 TSP PROBLEMS TAKEN FROM TSPLIB [51] BASED ON THE AVERAGE CPU TIME (*TIME*), THE NUMBER OF TRIALS FINDING THE OPTIMAL SOLUTION (*OPT TIMES*), AND MEAN QUALITY OF THE SOLUTION (*ERROR*). THE ERROR (%) IS DEFINED AS *(AVERAGE-OPTIMUM)/OPTIMUM* WHERE *AVERAGE* IS THE AVERAGE VALUE OF 30 INDEPENDENT TRIALS AND *OPTIMUM* IS THE LENGTH OF THE OPTIMAL TOUR FOR EACH TEST PROBLEM

problem	Time (Sec.)	Opt times	Error (%)
eil101	1.07	30	0.0000
kroa200	9.03	30	0.0000
lin318	31.33	30	0.0000
pcb442	56.94	30	0.0000
att532	161.93	30	0.0000
u574	173.87	30	0.0000
rat575	198.95	30	0.0000
u724	355.34	29	0.0005
rat783	523.21	30	0.0000
vm1084	949.98	29	0.0016
pcb1173	1143.75	30	0.0000
u1432	1734.85	27	0.0034
vm1748	4418.92	30	0.0000
pr2392	8773.49	30	0.0000
pcb3038	19865.7	30	0.0000

The new edges (v_r, v_{s+1}) and (v_s, v_{r+1}) replace the original edges (v_r, v_{r+1}) and (v_s, v_{s+1}) to generate the new solution. In fact, only the 20 cities nearest each city are considered.

IV. RESULTS AND DISCUSSION

In this section, HeSGA is first applied to several TSP benchmarks to evaluate its robustness (Table I). The HeSGA is compared with some other methods [48]–[50], which were efficient for solving TSPs according to our surveys (Table II). Microarray data analysis is a competitive field, and no decisive measure of the performance of methods is available. In this study, HeSGA was applied to eight practical microarray datasets (Table III), taken from earlier studies, to elucidate the performance of HeSGA and compare HeSGA with several well-known clustering (ordering) methods [1], [4], [19]. Two scoring functions, the gene-ordering formula (1), and the biological categories formula (7), were applied to evaluate the performance and derive the biological meaning of the results. The clustering and ordering of genes by the compared methods were also visualized (Fig. 5). Finally, the two scoring functions and visualized results of microarray analysis were discussed.

A. Performance of HeSGA on TSP

HeSGA was implemented in C++ and executed on a Pentium III 500 MHz personal computer with a single processor. HeSGA was tested on 17 TSP benchmarks, with from 101 to 13 509 cities. Each problem was tested over 30 trials. The value of λ was set as 20. The size of the population (N) was set to the number of cities in problems that involved fewer than 1000 cities, and half of the number of cities in larger problems.

Table I lists the results obtained using HeSGA to these problems, including execution time (*time*), the number of trials required by HeSGA to determine the optimal solution (*opt times*), and the mean quality of the solution (*error*) over 30 trials. As presented in Table I, HeSGA can find an optimal solution to each problem in at least 27 out of 30 independent trials. The mean error for each problem is only 0.004% from the optimal solution. The result of testing HeSGA on these TSPs is promising.

HeSGA was compared with some Lin–Kernighan (LK)-based heuristic methods, including Concorde LK [48], chained LK (CLK) [48], Johnson LK [49], iterative LK (ILK) [49], and Tabu search with LK [50], yielding results presented in Table II, to show the robustness of HeSGA on large TSPs. These five methods performed well on these test problems, according to the results of the "8th DIMACS Implementation Challenge: The Traveling Salesman Problem" (http://www.research.att.com/~dsj/chtsp/). The size of the problem *usa13509* is set to 4000, due to memory limitations.

Table II reveals that the HeSGA outperforms other LK-based approaches in testing problems. The HeSGA can determine the optimum and the mean solution quality is no more than 0.000 48 above the optimum for each testing problem, although the HeSGA is somewhat slower than these other methods. For a large problem, *usa13509*, ILK is around 50 times faster than HeSGA with a population size of 4000. Fortunately however, when the population size is 100, the running time for HeSGA is approximately that of ILK and the average tour length is 20 014 159 (0.001 566), which is slightly better than that of ILK.

TABLE II
COMPARISON HeSGA WITH SOME OTHER LK-BASED METHODS ON EIGHT LARGE TSP PROBLEMS BASED ON THE AVERAGE TOUR LENGTH AND THE ERROR, DEFINED AS *AVERAGE-OPTIMUM)/OPTIMUM*. THESE LK-BASED RESULTS ARE DIRECTLY SUMMARIZED FROM PREVIOUS STUDIES, INCLUDING CONCORDE LK [48], CLK [48], JOHNSON LK [49], ILK [49], AND TABU SEARCH WITH LK [50]. THE LENGTH OF THE OPTIMAL TOUR OF EACH PROBLEM IS IN BRACKETS IN THE FIRST COLUMN

Problem		HeSGA	Concorde LK	Johnson LK	ILK	CLK	Tabu with LK
vm1084	Tour length	**239300**	245931	241449	239349	239301	240238
(239297)	(error)	**(0.000016)**	(0.027723)	(0.008993)	(0.000217)	(0.000017)	(0.003932)
pcb1173	Tour length	**56892**	58110	57388	56897	56984	57290
(56892)	(error)	**(optimal)**	(0.021409)	(0.008718)	(0.000088)	(0.001617)	(0.006996)
u1432	Tour length	**152975**	156827	155386	153122	153328	153727
(152970)	(error)	**(0.000034)**	(0.025214)	(0.015794)	(0.000994)	(0.002340)	(0.004949)
vm1748	Tour length	**336556**	340206	338917	**336556**	336721	337683
(336556)	(error)	**(optimal)**	(0.010845)	(0.007015)	**(optimal)**	(0.000490)	(0.003349)
pr2392	Tour length	**378032**	390510	385029	378597	379629	380486
(378032)	(error)	**(optimal)**	(0.033008)	(0.018509)	(0.001495)	(0.004225)	(0.006492)
pcb3038	Tour length	**137694**	140668	139115	137861	138055	138893
(137694)	(error)	**(optimal)**	(0.021599)	(0.010320)	(0.001213)	(0.002622)	(0.008708)
fnl4461	Tour length	**182567**	185771	184624	182814	182840	184373
(182566)	(error)	**(0.000005)**	(0.017555)	(0.011273)	(0.001358)	(0.001501)	(0.009898)
usa13509	Tour length	**19992528**	20486590	20236820	20015598	20022550	20160648
(19982859)	(error)	**(0.000484)**	(0.025208)	(0.012709)	(0.001638)	(0.001986)	(0.008897)

TABLE III
SUMMARY OF EIGHT TEST MICROARRAY DATASETS CONSISTS OF THE DATASET NAME, THE NUMBER OF EXPERIMENTS, THE NUMBER OF GENES, AND THE DATA SOURCE

Dataset name	#. of experiments.	#. of genes	Source
Energy	80	147	The energy production related genes [1]
Metabolism	80	577	The metabolism related genes [1]
Complex	79	979	A selected subset [19] of the complete yeast dataset [1]
Subset 1	80	3222	Yeast Genes with no missing values [1]
All yeast	80	6221	The complete yeast dataset in Eisen *et al.* [1]
Subset 2	7	468	A randomly selected subset from the dataset [52]
Cdc15	24	782	A selected subset [19] of the *Saccharomyces cerevisiae* dataset [2]
Cell cycle	59	803	A selected subset [19] of the *Saccharomyces cerevisiae* dataset [2]

B. Microarray Datasets

Table III presents the eight tested biological datasets, selected from three independent microarray experiments on *Saccharomyces cerevisiae*, [1], [2], [52]. Each dataset is selected either randomly or according to biological functionality, such as energy production and associated metabolism. The numbers of genes in the datasets range from 147 to 6221, and the numbers of experiments range from 7 to 80. Functional catalogues, defined in the MIPS yeast complexes database, are used herein (http://www.mips.biochem.mpg.de/proj/yeast/catalogues).

MIPS categorizations, in a tree-like structure, were made by classifying around 6450 ORFs (open reading frames) into categories and subcategories. On first level of the tree, 3936 ORFs fall into at least one of the 17 defined categories, while 2514 ORFs classified as "unknown" or "not yet clear-cut."

C. Comparing Fitness of Orderings

Table IV summarizes the results obtained by the proposed method and five other approaches, based on the score used to optimize gene ordering (1). Single-link [53], complete-link [54], and average-link [55] are three extensively used hierarchical clustering approaches. The SOM [4] maps grids into k dimensional space and moves data points according to their distance. Software provided by Eisen *et al.* [1] was used to obtain the results of the above four methods. Joseph *et al.* [19] improved the average-link hierarchical clustering: by flipping the internal nodes to maximize the sum of the similarities between adjacent leaves. In this work, Joseph's method is implemented as in his original paper.

Table IV indicates that the HeSGA seems more robust than the comparative methods on eight testing problems, as measured by the lowness of the values of $\mathrm{fit}(\pi)$ that they found. The proposed algorithm consistently yielded the lowest fitness score using the test dataset. The quality of the solution obtained by of Joseph's method was the second best. The following subsections elucidate the relationship between $\mathrm{fit}(\pi)$ and the real biological meaning using a perfect scoring function and visualization.

D. Biological Interpretation

A new scoring function is investigated by referencing MIPS predefined functional categories to determine that $\mathrm{fit}(\pi)$ is biologically relevant. The biological hypothesis tested by microarray data analysis is that genes with a common functional role exhibit similar expression patterns across different experiments. This hypothesis is an approximation to a common biological reality, and is useful for determining the approximate distribution of groups. Based on this hypothesis, a new scoring function is defined, based on the functional categories defined in the MIPS yeast complexes database.

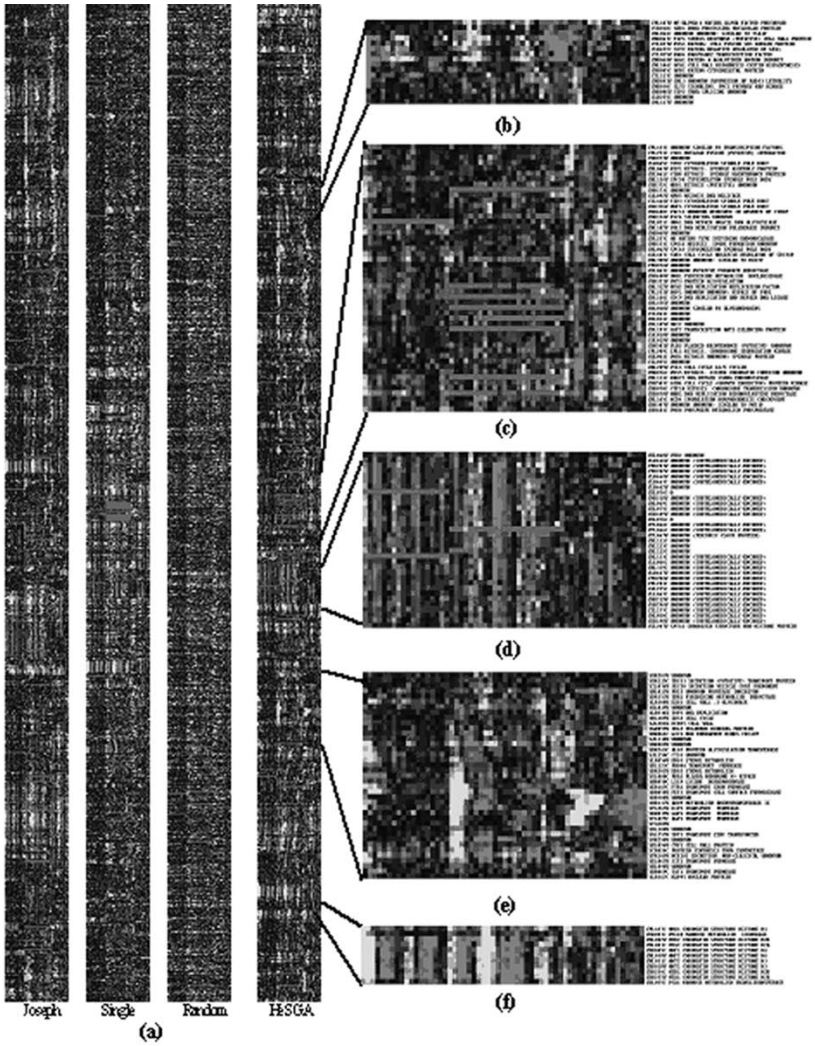


Fig. 5. Comparing HeSGA with three methods (Joseph's method, single-linkage method, and random permutation) on visualized gene expressions associated with gene orders obtained on the cell cycle data [2]. The expression profiles are represented as lines of gray boxes, each of which corresponds to a single experiment. (a) These visualized results of HeSGA, Joseph's method, and the single-linkage method are consistently more organized than those generated by random permutation. (b)–(f) Some grouped genes obtained by HeSGA have similar expression patterns and are coexpressed in each group: (b) mating related; (c) mitosis and cytoskeletal related; (d) subtelomerically encoded proteins; (e) transport permeases; and (f) chromatin structure histones.

TABLE IV
COMPARISONS WITH SINGLE-LINKAGE, AVERAGE-LINKAGE, COMPLETE-LINKAGE, SELF-ORGANIZING MAP, AND JOSEPH [19] ON EIGHT BIOLOGICAL DATASETS BASED ON THE GENE ORDERING FUNCTION, $\mathrm{fit}(\pi)$ (1)

Dataset name	HeSGA	Single- linkage	Average- linkage	Complete- linkage	SOM	Joseph
Energy	**61.57**	95.17	80.73	77.12	105.68	69.91
Metabolism	**247.68**	422.72	326.43	307.92	464.37	284.61
Complex	**308.80**	598.87	420.63	392.05	589.92	366.53
Subset 1	**417.55**	568.14	463.33	445.96	599.23	436.97
All yeast	**1715.6**	5470.7	2552.9	2341.9	4183.8	2192.8
Subset 2	**69.24**	84.46	76.28	74.93	90.66	72.65
Cdc15	**140.41**	655.53	247.03	230.09	335.92	191.11
Cell cycle	**221.77**	596.59	315.99	291.21	409.38	267.83

Each gene (or ORFs) that has undergone MIPS categorization belongs to at lest one category, so, a vector $V(g) = (v_1, v_2, \ldots, v_j)$ was used to represent the category status of each gene g, where j is the number of categories. The value of v_i is one if gene g is in the ith category; otherwise v_i is zero.

Based on the information about categorization, the score of a gene order $\mathrm{cat}(\pi)$ is

$$\mathrm{cat}(\pi) = \sum_{j=1}^{M-1} G(g_{\pi_j}, g_{\pi_{j+1}}) \tag{7}$$

A gene order $\pi = (g_1, g_2, g_3, g_4, g_5)$
$V(g_1) = (1,1,0,0,0,0,0,0,0,1,0,0,0,0,0)$
$V(g_2) = (0,1,1,0,0,0,0,0,0,1,0,0,1,0,0)$
$V(g_3) = (0,0,1,0,0,1,0,0,0,1,0,0,0,0,1)$
$V(g_4) = (0,0,0,0,0,1,0,0,0,0,0,0,0,0,0)$
$V(g_5) = (0,0,0,0,0,0,0,0,0,0,0,0,1,0,0)$
$G(g_1, g_2) = 2$, $G(g_2, g_3) = 2$, $G(g_3, g_4) = 1$, $G(g_4, g_5) = 0$
$cat(\pi) = G(g_1, g_2)+G(g_2, g_3)+G(g_3, g_4)+G(g_4, g_5)$
$= 2 + 2 + 1 + 0 = 5$

Fig. 6. Example of calculating $\mathrm{cat}(\pi)$ (7): $V(g)$ is the category vector of the gene g and $G(g_i, g_j)$ is the distance between genes g_i and g_j.

where M is the number of genes, g_{π_j} and $g_{\pi_{j+1}}$ are the adjacent genes, and $G(g_{\pi_j}, g_{\pi_{j+1}})$ is

$$G(g_{\pi_j}, g_{\pi_{j+1}}) = \sum_{i=1}^{k} V(g_{\pi_j})_i V(g_{\pi_{j+1}})_i. \tag{8}$$

The score of a gene order $\mathrm{cat}(\pi)$ is high if the genes in the same group are aligned. If the $\mathrm{cat}(\pi)$ score is the highest possible, then genes are correctly classified. Fig. 6 presents an example in which $\mathrm{cat}(\pi)$ is calculated. The scoring function (7) is perfect for interpreting biological meaning because it is based on predefined biological categories of the tested microarray data.

The scoring function $\mathrm{cat}(\pi)$ (7) yields the scores of the solutions (Table IV) obtained by the proposed method and the five other methods. HeSGA was also the best method and Joseph's method was the second best. Tables IV and V reveal that the fitness of $\mathrm{fit}(\pi)$ is small when $\mathrm{cat}(\pi)$ is large.

The following procedure was used to generate ten solutions to discuss further the relationship between $\mathrm{fit}(\pi)$ and $\mathrm{cat}(\pi)$. A gene order π_0 was randomly generated for each solution, and a new order π was then generated by switching of the genes in the old order π_0 at random. The fitness values $\mathrm{fit}(\pi)$ and $\mathrm{cat}(\pi)$ were calculated for this new gene order. The new gene order replaces of the old gene order if $\mathrm{fit}(\pi)$ for the new order exceeds that for the old. These steps are repeated 10 000 times for each gene order. Fig. 7 reveals that $\mathrm{fit}(\pi)$ is highly correlate with $\mathrm{cat}(\pi)$. These results demonstrate that the scoring function $\mathrm{fit}(\pi)$ is a useful scoring function and that HeSGA is a robust method for optimizing the scoring function.

The scoring function $\mathrm{cat}(\pi)$ is good for optimizing the order of genes. Unfortunately, using $\mathrm{cat}(\pi)$ as the fitness function is impractical since the information about categories is unknown in the real world. Table V and Fig. 7 indicate that $\mathrm{fit}(\pi)$ can be used to approximate $\mathrm{cat}(\pi)$.

E. Visualization of Results

Users can appreciate the results of the microarray experiments if the expression data are presented as colored boxes. Fig. 5(a) presents the visualized gene expressions associated with gene orders, obtained by applying HeSGA, Joseph's method, single linkage method, and random permutation, to the cell cycle data [2]. The expression profiles are represented as lines of gray boxes, each of which corresponds to a single experiment. All of the results in Fig. 5 were displayed using TreeView [1].

TABLE V
COMPARISONS WITH SINGLE-LINKAGE, AVERAGE-LINKAGE, COMPLETE-LINKAGE, SELF-ORGANIZING MAP, AND JOSEPH [19] ON EIGHT BIOLOGICAL DATASETS BASED ON THE PREDEFINED FUNCTIONAL CATEGORIES, $\mathrm{cat}(\pi)$ (7)

Dataset name	Category	HeSGA	Single	Average	Complete	SOM	Joseph
Energy	10	**50**	40	47	46	39	43
Metabolism	8	**184**	156	176	161	170	178
Complex	17	**1545**	1430	1499	1485	1369	1516
Subset 1	17	**1711**	1421	1546	1618	1222	1588
All yeast	17	**3933**	2186	2916	2978	2302	2947
Subset 2	17	**192**	173	186	183	192	196
Cdc15	17	**336**	316	327	333	316	334
Cell cycle	17	**428**	350	401	396	352	400

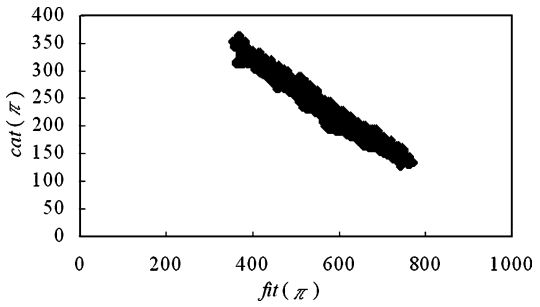


Fig. 7. Relationship between two scoring functions: $\mathrm{fit}(\pi)$ [(1), our objective function] and $\mathrm{cat}(\pi)$ [(7), the scoring function which references the MIPS functional categories]. π represents a gene order. $\mathrm{fit}(\pi)$ and $\mathrm{cat}(\pi)$ are highly correlated based on 100 000 gene orders generated with a simple procedure (see text).

These visualized results of HeSGA, Joseph's method, and the single-linkage method are more organized than those generated by random permutation [Fig. 5(a)]. HeSGA and Joseph's method yield similar results, which were the best of all those obtained by the compared methods. For the other seven testing datasets, HeSGA also generated favorable gene orders. The visualized results were consistent with the previous discussion of $\mathrm{fit}(\pi)$ and $\mathrm{cat}(\pi)$ in Sections IV-C and IV-D.

Neighboring genes in a particular group have the same function. As shown in Fig. 5(a)–(f), most genes in a single group have similar expression patterns and participate in shared cellular processes. For example, most mating-related genes are grouped as depicted in Fig. 5(b); mitosis processes and cytoskeleton-related genes are grouped as shown in Fig. 5(c), and transport permeases genes are grouped as in Fig. 5(e).

The 803 genes in the cell cycle dataset include 27 subtelomerically encoded genes and nine histone genes with chromatin structures, whose expressions are depicted in Fig. 8(a) and (b), respectively. As shown, histone genes (subtelomerically encoded genes) exhibit very similar patterns of expression in the experiments and should, thus, be aligned. HeSGA grouped these nine histones genes in a single group [Fig. 5(f)] of 11 genes. HeSGA clustered 24 subtelomerically encoded genes in one cluster [Fig. 5(d)] of 33 genes. Joseph's method divided 27 subtelomerically encoded genes into three groups, which contained 14, 12, and 1 gene. The single-linkage method generated two groups of 14 and 7 genes, leaving 6 singletons. The visualization of the results demonstrates that HeSGA not only ordered genes smoothly but also grouped genes with similar expressions.

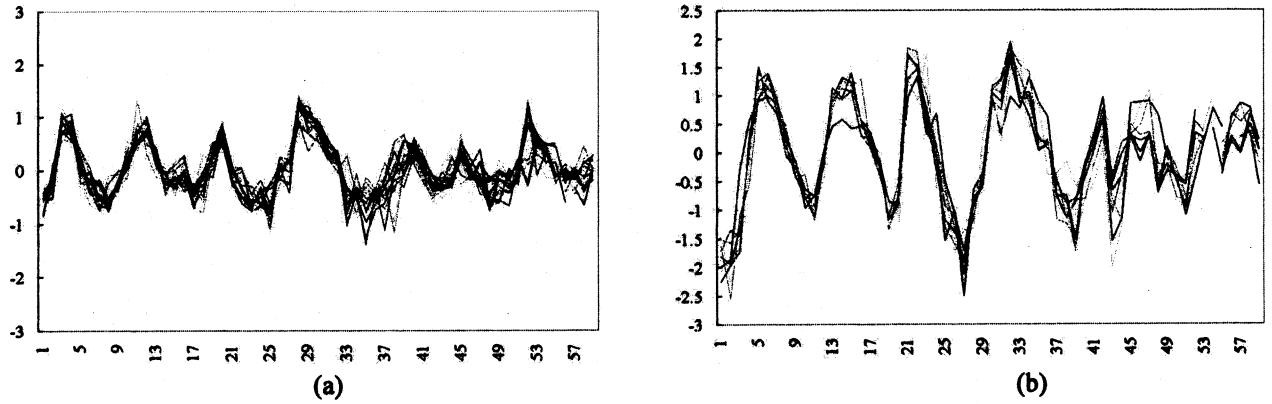


Fig. 8. HeSGA results for typical expression patterns of two clustered genes. These two expression profiles are (a) 27 subtelomerically encoded genes and (b) nine chromatin structure histone genes in the cell cycle dataset. The x axis represents the experiments and the y axis represents the logarithm scale of the expression level. The expression patterns of the genes in the same group seem very similar. HeSGA clustered 24 subtelomerically encoded genes in one cluster [Fig. 5(d)] that contains 33 genes and grouped all histones genes in one group [Fig. 5(f)] that composes 11 genes. The results demonstrate that HeSGA not only ordered genes smoothly but also grouped genes with similar expressions.

V. Conclusion

We have developed an evolutionary-based method (HeSGA) to solve simultaneously gene clustering and gene-ordering problems in the analysis of microaray data. To cluster and order the gene expression patterns is essential to analyze a large amount of microarray data. By integrating a number of genetic operators (EAX operator and NJ operator), each having a unique search mechanism, HeSGA seamlessly blends the local and global searches so that they work cooperatively. Experiments on eight test microarray datasets ranging in size from 147 to 6221 genes verify that the robustness and adaptability of HeSGA in exploring the conformational space in which genes are clustered and ordered. HeSGA is able to determine meaningful cluster boundaries and the relationship between different clusters. The clustering and visual results show that HeSGA ordered the genes in a smooth way and grouped the genes with similar gene expressions together. These results demonstrate that HeSGA is a useful tool for analyzing microarray data.

References

[1] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein, "Cluster analysis and display of genome-wide expression patterns," in *Proc. National Academy Sciences*, vol. 95, 1998, pp. 14 863–14 868.

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

[3] R. Cho *et al.*, "A genome-wide transcriptional analysis of the mitotic cell cycle," *Molecular Cell*, vol. 2, pp. 65–73, 1998.

[4] P. Tamato, D. Slonim, J. Mesirow, 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," in *Proc. National Academy Sciences*, vol. 96, 1999, pp. 2907–2912.

[5] X. Wen, S. Fuhrman, G. S. Michales, D. B. Carr, S. Smith, J. Barker, and R. Somogyi, "Large-scale temporal gene expression mapping of central nervous system development," in *Proc. National Academy Sciences*, vol. 95, 1998, pp. 334–339.

[6] A. Brazma, I. Jonassen, J. Vilo, and E. Ukkonen, "Predicting gene regulatory elements in silico on a genomic scale," *Genome Res.*, vol. 8, pp. 1202–1215, 1998.

[7] P. D'haeseleer, S. Liang, and R. Somogyi, "Genetic network inference: From co-expression clustering to reverse engineering," *Bioinformatics*, vol. 16, pp. 707–726, 2000.

[8] A. A. Alizadeh *et al.*, "Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling," *Nature*, vol. 403, pp. 503–511, 2000.

[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," in *Proc. National Academy Sciences*, vol. 96, 1999, pp. 6745–6750.

[10] S. Kawasaki, C. Borchert, M. Deyholos, H. Wang, S. Brazille, K. Kawai, D. Galbraith, and H. J. Bohnert, "Gene expression profiles during the initial phase of salt stress in rice," *Plant Cell*, vol. 13, pp. 889–906, 2001.

[11] A. B. Khodursky, B. J. Peter, N. R. Cozzarelli, D. Botstein, P. O. Brown, and C. Yanofsky, "DNA microarray analysis of gene expression in response to physiological and genetic changes that affect tryptophan metabolism in Escherichia coli," in *Proc. National Academy Sciences*, vol. 97, 2000, pp. 12 170–12 175.

[12] R. Schaffer, J. Landgraf, M. Accerbi, V. Simon, M. Larson, and E. Wisman, "Microarray analysis of diurnal and circadian-regulated genes in arabidopsis," *Plant Cell*, vol. 13, pp. 113–123, 2001.

[13] B. J. T. Morgan and A. P. G. Ray, "Non-uniqueness and inversions in cluster analysis," *Appl. Stat. (General Interest Section)*, vol. 44, pp. 117–134, 1995.

[14] R. Herwig, A. J. Poustka, C. Muller, C. Bull, H. Lehrach, and J. O'Brien, "Large-scale clustering of cDNA-fingerprinting data," *Genome Res.*, vol. 9, pp. 1093–1105, 1999.

[15] Y. Barash and R. Friedman, "Context-specific Bayesian clustering for gene expression data," *J. Computational Biology*, vol. 9, pp. 169–191, 2002.

[16] A. Ben-Dor, R. Shamir, and Z. Yakhin, "Clustering gene expression patterns," *J. Computational Biology*, vol. 6, pp. 281–297, 1999.

[17] R. Sharan and R. Shamir, "CLICK: A clustering algorithm with applications to gene expression analysis," in *Proc. 8th Int. Conf. Intelligent Systems for Molecular Biology (ISMB 2000)*, 2000, pp. 307–316.

[18] T. Biedl, B. Brejová, E. D. Demaine, A. M. Hamel, and T. Vinar, "Optimal Arrangement of Leaves in the Tree Representing Hierarchical Clustering of Gene Expression Data," Dept. Computer Sci., Univ. Waterloo, Tech. Rep. 2001-14, 2001.

[19] Z. Bar-Joseph, D. K. Gifford, and T. S. Jaakkola, "Fast optimal leaf ordering for hierarchical clustering," *Bioinformatics*, vol. 17, pp. 22–29, 2001.

[20] J. Herrero, A. Valencia, and J. Dopazo, "A hierarchical unsupervised growing neural network for clustering gene expression patterns," *Bioinformatics*, vol. 17, pp. 126–136, 2001.

[21] J. Holland, *Adaptation in Neural and Artificial Systems*. Ann Arbor, MI: Univ. of Michigan Press, 1975.

[22] K. A. De Jong, "An analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, Univ. Michigan, 1975.

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

[24] R. Judson, "Genetic algorithms and their use in chemistry," in *Reviews in Computational Chemistry*, K. B. Lipkowitz and D. B. Boyd, Eds. New York: VCH, 1997, vol. 10, pp. 1–66.

[25] L. 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, pp. 1131–1142, 2001.

[26] Y. Nagata and S. Kobayashi, "Edge assembly crossover: A high-power genetic algorithm for the traveling salesman problem," in *Proc. 7th Int. Conf. Genetic Algorithms (ICGA)*, 1997, pp. 450–457.

[27] P. Merz and B. Freisleben, "Genetic local search for the TSP: New results," in *Proc. IEEE Int. Conf. Evolutionary Computation (CEC)*, 1997, pp. 159–164.
[28] S. Ronald, "Preventing diversity loss in a routing genetic algorithm with hash tagging," *Complexity Int.*, vol. 2, pp. 133–140, 1995.
[29] J.-M. Yang, J.-T. Horng, and C.-Y. Kao, "A genetic algorithm with adaptive mutations and family competition for training neural networks," *Int. J. Neural Syst.*, vol. 10, pp. 333–352, 2000.
[30] J.-M. Yang and C.-Y. Kao, "A family competition evolutionary algorithm for automated docking of flexible ligands to proteins," *IEEE Tran. Inform. Technol. Biomed.*, vol. 4, pp. 225–237, 2000.
[31] J.-M. Yang, J.-T. Horng, C.-J. Lin, and C.-Y. Kao, "Optical coating designs using an evolutionary algorithm," *Evolutionary Computation*, vol. 9, pp. 421–443, 2001.
[32] H.-K. Tsai, J.-M. Yang, and C.-Y. Kao, "A genetic algorithm for traveling salesman problems," in *Proc. Genetic Evolutionary Computation Conf. (GECCO)*, 2001, pp. 687–693.
[33] M. Schena, D. Shalon, R. W. Davis, and P. O. Brown, "Quantitative monitoring of gene expression patterns with a complementary DNA microarray," *Science*, vol. 270, pp. 467–470, 1995.
[34] R. J. Lipshutz, D. Morris, M. Chee, E. Hubbell, M. J. Kozal, N. Shah, N. Shen, R. Yang, and S. P. A. Fodor, "Using oligonucleotide probe arrays to access genetic diversity," *Biotechniques*, vol. 19, pp. 442–447, 1995.
[35] M. Chee, R. Yang, E. Hubbel, A. Berno, X. C. Huang, D. Stern, J. Winkler, D. J. Lockhart, M. S. Morris, and S. P. A. Fodor, "Accessing genetic information with high density DNA arrays," *Science*, vol. 274, pp. 610–614, 1996.
[36] J. G. Hacia, J. B. Fan, O. Ryder, L. Jin, K. Edgemon, G. Ghandour, R. A. Mayer, B. Sun, L. Hsie, C. M. Robbins, L. C. Brody, D. Wang, E. S. Lander, R. Lipshutz, S. P. A. Fodor, and F. S. Collins, "Determination of ancestral alleles for human single-nucleotide polymorphisms using high-density oligonucleotide arrays," *Nature Genetics*, vol. 22, pp. 164–167, 1999.
[37] L. Luo, R. C. Salunga, H. Guo, A. Bittner, J. E. Joe, J. E. Galindo, H. Xiao, K. E. Rogers, J. S. Wan, M. R. Jackson, and M. G. Erlander, "Gene expression profiles of laser-captured adjacent neuronal subtypes," *Nature Medicine*, vol. 5, pp. 117–122, 1999.
[38] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gassenbeck, 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, pp. 531–537, 1999.
[39] P. O. Brown and D. Botstein, "Exploring the new world of the genome with DNA microarrays," *Nature Genetics*, vol. 21, pp. 33–37, 1999.
[40] D. G. Wang *et al.*, "Large-scale identification, mapping, and genotyping of single-nucleotide polymorphisms in the human genome," *Science*, vol. 280, pp. 1074–1082, 1998.
[41] M. R. Garey and D. S. Johnson, *Computers and intractability: A guide to the theory of NP-completeness*. New York: Freeman, 1979.
[42] C. Korostensky and G. H. Gonnet, "Using traveling salesman problem algorithms for evolutionary tree construction," *Bioinformatics*, vol. 16, pp. 619–627, 2000.
[43] S. H. Chen, S. F. Smith, and S. C. Guerra, "The GENIE is out! (Who needs fitness to evolve?)," in *Proc. Congress Evolutionary Computation (CEC)*, 1999, pp. 2102–2106.
[44] N. Krasnogor and J. Smith, "A memetic algorithm with self-adaptive local search: TSP as a case study," in *Proc. Genetic Evolutionary Computation Conf. (GECCO)*, 2000, pp. 987–994.
[45] T. Stützle and M. Dorigo, "ACO algorithms for the traveling salesman problem," *Evolutionary Algorithms Eng. Comput. Sci.*, pp. 163–183, 1999.
[46] J. Watson *et al.*, "The traveling salesrep problem, edge assembly crossover, and 2-opt," *Parallel Problem Solving Nature V (PPSN V)*, pp. 823–832, 1998.
[47] T. Baeck, *Evolutionary Algorithms in Theory and Practice*. New York: Oxford Univ. Press, 1996.
[48] D. Applegate, R. Bixby, V. Chvatal, and W. J. Cook, "Finding Tours in the TSP," Dept. Comput. Appl. Math., Rice Univ., Houston, TX, Tech. Rep. TR99-05, 1999.
[49] D. S. Johnson and L. A. McGeoch, "The traveling salesman problem: A case study in local optimization," in *Local Search in Combinatorial Optimization*, E. H. L. Aarts and J. K. Lenstra, Eds. New York: Wiley, 1997, pp. 215–310.
[50] M. Zachariasen and M. Dam, "Tabu search on the geometric traveling salesman problem," in *Proc. Metaheuristics Int. Conf.*, 1995, pp. 571–587.
[51] G. Reinelt, "TSPLIB- A traveling salesman problem library," *ORSA J. Computing*, vol. 3, pp. 376–384, 1991.
[52] J. L. DeRisi, V. R. Iyer, and P. O. Brown, "Exploring the metabolic and genetic control of gene expression on a genomic scale," *Science*, vol. 278, pp. 680–686, 1997.
[53] R. Sibson, "Slink: An optimally efficient algorithm for a complete link method," *Comput. J.*, vol. 16, pp. 30–34, 1973.
[54] D. Defays, "An efficient algorithm for a complete link method," *Comput. J.*, vol. 20, pp. 364–366, 1977.
[55] E. M. Voorhees, "Implementing agglomerative hierarchic clustering algorithms for use in document retrieval," *Inform. Proc. Management*, vol. 22, pp. 465–476, 1986.

Huai-Kuang Tsai received the B.S., M.S., and Ph.D. degrees, in computer science and information engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 1996, 1998, and 2003, respectively.

Since October 2003, he has been a Postdoctoral Fellow at National Taiwan University. His research interests include evolutionary computation, bioinformatics, combinatorial optimization, and data mining.

Jinn-Moon Yang received the M.S. degree from National Central University, Chung-Li, Taiwan, R.O.C., in 1994, the M.B.A. degree from Tamkang University, Taipei, Taiwan, R.O.C., and the Ph.D. degree in computer science and information engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 2001.

From 1985 to 1998, he worked for PanChiao Telecommunications, Ministry of Transportation and Communications. From 1998 to 2001, he worked for Chunghwa Telecom Training Institute, Taipei, Taiwan, R.O.C. He has been an Assistant Professor with the Department of Biological Science and Technology Computer Science, Institute of Bioinformatics, National Chiao Tung University, Taipei, Taiwan, R.O.C., since 2001. He has published more than 40 technical papers in various journals and conference records. His research interests include evolutionary computation, bioinformatics, structural biology, rational drug-design design, and machine learning.

Yuan-Fan Tsai received the B.S., M.S., and Ph.D. degrees, in hydraulics and ocean engineering, National Cheng Kung University, Tainan, Taiwan, R.O.C., in 1992, 1994, and 1999, respectively.

He has been an Assistant Professor of the Department of Social Studies Education, National Taipei Teachers College, Taipei, Taiwan, R.O.C., since 2004. His research interests include hydraulics engineering, numerical simulation, bioinformatics, and evolutionary computation. He has published more than 30 technical papers in various journals and conference records.

Cheng-Yan Kao was born in Taipei, Taiwan, R.O.C., in 1948. He received the B.S. degree in mathematics from National Taiwan University, Taipei, Taiwan, R.O.C., in 1971, and the M.S. degree in computer science, the M.S. degree in statistics, and the Ph.D. degree in computer science, all from the University of Wisconsin-Madison, in 1976, 1978, and 1981, respectively.

He worked for Ford Aerospace and the Unisys Corporation and for General Electric from 1980 to 1989 at the Johnson Space Center, NASA, Houston, TX. He has been a Professor with the Department of Computer Science and Information Engineering, National Taiwan University, since 1990. He has published more than 40 technical papers in various journals and conference records. His research interests include evolutionary computation, bioinformatics, optimization, and artificial intelligence.