Engineering Optimization
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/geno20

Heterogeneous Selection Genetic Algorithms For Traveling Salesman Problems

Huai-Kuang Tsai [a], Jinn-Moon Yang [b], Yuan-Fang Tsai [c] & Cheng-Yan Kao [c d]

[a] Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan

[b] Department of Biological Science and Technology and Institute of Bioinformatics, National Chiao Tung University, Hsinchu, 30050, Taiwan

[c] Department of Information Management, Chung Yu Junior College of Business Administration, Keelung 201, Taiwan

[d] Bioinformatics Center, National Taiwan University, Taipei 106, Taiwan

Published online: 22 Jan 2013.

To cite this article: Huai-Kuang Tsai, Jinn-Moon Yang, Yuan-Fang Tsai & Cheng-Yan Kao (2003) Heterogeneous Selection Genetic Algorithms For Traveling Salesman Problems, Engineering Optimization, 35:3, 297-311, DOI: 10.1080/0305215031000109622

To link to this article: http://dx.doi.org/10.1080/0305215031000109622

Eng. Opt., Vol. 35, No. 3, June, 2003, pp. 297–311

HETEROGENEOUS SELECTION GENETIC ALGORITHMS FOR TRAVELING SALESMAN PROBLEMS

HUAI-KUANG TSAI[a,*], JINN-MOON YANG[b], YUAN-FANG TSAI[c] and CHENG-YAN KAO[a,d]

[a]*Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan;* [b]*Department of Biological Science and Technology and Institute of Bioinformatics, National Chiao Tung University, Hsinchu 30050, Taiwan;* [c]*Department of Information Management, Chung Yu Junior College of Business Administration, Keelung 201, Taiwan;* [d]*Bioinformatics Center, National Taiwan University, Taipei 106, Taiwan*

(Received 23 August 2002; In final form 18 February 2003)

This paper proposes a genetic algorithm, called the heterogeneous selection genetic algorithm (HSGA), integrating local and global strategies via family competition and edge similarity, for the traveling salesman problem (TSP). Local strategies include neighbor-join mutation and family competition, and global strategies consist of heterogeneous pairing selection and edge assembly crossover. Based on the mechanisms of preserving and adding edges, the search behaviors of neighbor-join mutation and edge assembly crossover are studied. The proposed method has been implemented and applied to 17 well-known TSPs whose numbers of cities range from 101 to 13,509. Experimental results indicate that this approach, although somewhat slower, performs very robustly and is very competitive with other approaches in the best surveys. This approach is able to find the optimum, and the average solution quality is within 0.00048 above the optima of each test problem.

Keywords: Edge assembly crossover; Heterogeneous pairing selection; Family competition; Genetic algorithm; Neighbor-join mutation; Traveling salesman problem

1 INTRODUCTION

The traveling salesman problem (TSP) is a well-known NP-hard optimization problem which requires the determination of the shortest route passing through a set of M cities under the condition that each city is visited exactly once. TSPs raise important issues because many problems in science, engineering, and bioinformatics fields, such as routing, scheduling problems, flexible manufacturing systems, physical mapping problems [2], and phylogenetic tree construction [15] can be formulated as TSPs.

A large number of approaches have been developed for solving TSPs. They can be roughly divided into local and global search methods. The local search algorithms, such

* Corresponding author. E-mail: d7526010@csie.ntu.edu.tw

ISSN 0305-215X print; ISSN 1029-0273 online
DOI: 10.1080/0305215031000109622

as 2-opt and 3-opt [17], and the Lin–Kernigan heuristic [18], are efficient but may get stuck at local minima. Some stochastic approaches, including simulated annealing [14], Hopfield neural networks [11], tabu search [34], and evolutionary algorithms [6, 8, 19, 20, 35], have been proposed to reduce the disadvantage of local search methods. However, these approaches often converge more slowly than local search methods.

A very promising direction among these stochastic search approaches is the evolutionary algorithm, considered as a global search mechanism. It is based on the ideas borrowed from genetics and natural selection. An evolutionary algorithm is a generally adaptable concept for problem solving that is especially well suited for solving difficult optimization problems, where traditional optimization methods are less efficient. Genetic algorithms [9], evolution strategies [4], and evolutionary programming [7] are three main independently developed but strongly related evolutionary algorithms. Because the original evolutionary algorithms are not very efficient for some specific application domains, one trend is to incorporate local search techniques into evolutionary algorithms to improve solution quality [10, 33]. Such a hybrid approach may possess both the global optimality of the evolutionary algorithm as well as the convergence of the local search.

To further improve the above evolutionary approaches, the present authors applied two key mechanisms to evolutionary algorithms: (1) incorporating multi-genetic operators, including local and global search mechanisms, each compensating for the others' disadvantages; (2) maintaining the population diversity naturally. Previous results have demonstrated that these mechanisms are useful for some continuous optimization problems [31], training neural networks [29], thin-film optical coatings [30], and flexible ligand docking [32]. Two special ideas for TSPs were also considered: preserving good edges [20] and adding new edges into the offspring [28].

This paper proposes an evolutionary algorithm, called the heterogeneous selection genetic algorithm (HSGA), by applying the above four ideas for solving TSPs. The HSGA consists of a new neighbor-join mutation (NJ), an edge assembly crossover (EAX) [20], a new crossover pairing selection, named heterogeneous pairing selection (HpS), and a family competition. The NJ mutation, viewed as a local search mechanism, has properties of both mutation and recombination. The EAX, viewed as a global search mechanism, has been considered as a useful crossover operator for TSPs [5, 16, 25, 28]. The EAX and the NJ mutation generate the offspring by preserving good edges from parents and adding new edges based on heuristics. The HpS selects two parents for crossover operators to reduce the premature convergence effect based on the edge similarity of a population. Finally, the family competition, derived from $(1+\lambda)-$ES [4] and the Lin–Kernigan heuristic, acts as a local search procedure. The main difference in methodology between the present work and previous studies [27] is the addition of the HpS selection and the NJ mutation.

The method was applied to 17 well-known traveling salesman problems [24] whose numbers of cities range from 101 to 13,509 cities. The experimental results indicate that the solution quality of the HSGA stays within 0.00048 of the optima of the test problems, and is more robust than comparative approaches. The analysis results also show that the performance of the NJ mutation and the HpS selection is very promising.

The rest of this paper is organized as follows. Section 2 introduces the evolutionary nature of the proposed approach. Section 3 analyzes the characteristics and search behaviors of the HSGA. Section 4 shows the comparative results of the HSGA with five hybrid evolutionary approaches and with five heuristic methods on eight large TSPs. Concluding comments are drawn in Section 5.

2 APPROACH

2.1 System Architecture Overview

In this section, the details of the proposed genetic algorithm for TSPs are presented. The HSGA has four major mechanisms, including a new crossover pairing selection, named heterogeneous pairing selection (HpS), the edge assembly crossover (EAX), a family competition, and a new mutation operator, called neighbor-join mutation (NJ). The HpS based on the edge similarity selects two parents for the EAX crossover. The EAX and the NJ mutation are genetic operators considered to be able to preserve and add good edges to generate a child. The family competition is a local search mechanism incorporated into the EAX and the NJ mutation. These four mechanisms have been studied to balance exploration and exploitation in the search space.

Figure 1 shows the main steps of the HSGA. N solutions are randomly generated as the initial population. Each solution is represented as a random permutation from 1 to M where M is the number of cities. After evaluating the fitness, each solution in the population sequentially becomes the "family father (s_i)" which applies the following steps to generate a child: The family father uses the HpS to select itself (s_i) and another individual from the population based on the edge similarity. These two individuals become the parents of the EAX which generates only one intermediate offspring (I_i). The NJ mutation is then executed L times (L is the family competition length) to generate a child (c_i) by refining the intermediate solution I_i.

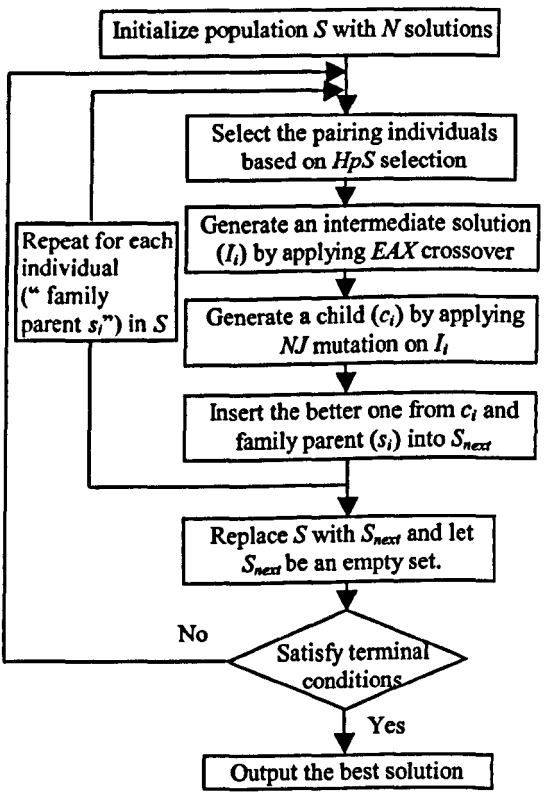


FIGURE 1 Overview of proposed genetic algorithm (HSGA).

In each pair of family father (s_i) and its child (c_i), the one with the better solution survives. Each individual (s_i) in the population sequentially executes the above steps to generate its child where $1 \leq i \leq N$. These N solutions become the new population of the next generation.

The algorithm is terminated when one of the following criteria is satisfied: (1) the maximum preset search time is exhausted, (2) all individuals of a population represent the same solution, or (3) all of the children generated in five consecutive generations are worse than their respective family parents. In the following subsections, the HpS selection, the EAX, and the NJ mutation are described.

2.2 Heterogeneous Pairing Selection (HpS)

For each "family father (s_i)", the HpS selects s_i and another individual from the current population based on the edge similarity for crossover operators, such as EAX in this paper. The HpS is used to the disadvantages of trapping into local optimal by avoiding incest. In evolutionary processes, incest may cause two ill effects: loss of population diversity and ineffective execution of crossover operations. The formulation and implementation of the HpS is described as follows: Let $\{s_1, s_2, \ldots, s_N\}$ be the current population, $E(s_i)$ be the set of the edges of s_i, and $\|E(s_i)\|$ be the number of the edges of $E(s_i)$. The number of identical edges $\|T_{i,j}\|$ of two individuals (s_i and s_j) is defined as

$$\|T_{i,j}\| = \|E(s_i) \cap E(s_j)\|$$

For each individual s_i, let t_i be the average number of identical edges between s_i and the other individuals in the population

$$t_i = \frac{1}{N-1} \sum_{j=1, j \neq i}^{N} \|T_{i,j}\| \tag{1}$$

where N is the population size. There are two extreme cases for the value of t_i: $t_i = 0$ when no edge of s_i appears in the other individuals and $t_i = M$ when all individuals are the same in the population where M is the number of the cities of a TSP.

For given the individual s_i, the HpS selects s_i and another individual s_j with $\|T_{i,j}\| \leq t_i$ for the EAX operator. This similarity-based mechanism is useful for keeping the population diversity. Our experimental results were consistent with this claim.

In practical implementation, another method was used to calculate t_i because the time complexity of getting all t_i through calculating $\|T_{i,j}\|$ is $O(N^2M^2)$. At the beginning of each generation, $F(e)$, the count of edge e appearances in the current population, is calculated in advance where $e \in \{E(s_1) \cup E(s_2) \cup \cdots \cup E(s_N)\}$. The sum of $\|T_{i,j}\|$ of s_i in the population, can be reformulated as

$$\begin{aligned} \sum_{j=1, j \neq i}^{N} \|T_{i,j}\| &= \sum_{j=1}^{N} \|T_{i,j}\| - \|T_{i,i}\| \\ &= \sum_{e \in E(s_i)} F(e) - M \\ &= \sum_{e \in E(s_i)} (F(e) - 1) \end{aligned} \tag{2}$$

Substituting Eq. (2) into Eq. (1), t_i becomes

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

Therefore, all t_i can be calculated in $O(NM)$ via looking up the pre-calculated table. Since the EAX crossover also uses the information $F(e)$, the extra effort of calculating t_i is limited.

2.3 Edge Assembly Crossover

The EAX [20] is considered a powerful crossover operator [5, 16, 25, 28]. It has two important features: preserving parents' edges with a novel approach and adding new edges with a greedy method, analogous to a minimal spanning tree. Several issues, such as the selection mechanism and heuristic methods, influencing EAX performance have been discussed [21, 22, 27, 28]. This paper retains the main spirit of the EAX and used the HpS to replace the random pair selection (RpS), which was the original selection mechanism of the EAX genetic algorithm. The EAX is considered as the global search strategy in our proposed algorithm.

The EAX is briefly described here. Two individuals, denoted as A and B, are selected as the parents. The EAX first merges A and B into a single graph denoted G. The EAX travels G to generate many *AB-cycles* by alternately picking edges from parents A and B. According to the heuristic and random selection rules, some of the *AB-cycles* are selected to generate a quasi solution which contains some disjointed subtours. Then, the EAX uses a greedy method to merge these disjointed subtours into a valid solution. This solution is returned if the fitness of this solution is better than its parents. Otherwise this procedure is repeated until a solution is found that is better than both A and B, or L children are produced where L is the family competition length.

2.4 Neighbor-Join Mutation

A new mutation operator, called *neighbor-join* (NJ) mutation, is proposed to improve an intermediate solution generated by the EAX. The NJ mutation constructs a new solution by stealing edges from other individuals in the population or by considering the geometric information. The NJ mutation is inspired by the inver-over mutation [26] and by analyzing the TSP search space [23]. The main difference between the inver-over mutation and other mutations is that it inherits edges both from its parent and from other individuals in the current population. On the other hand, according to the analysis of the optimal tours of some TSPs [23], most of the links in the optimal tours were found to be neighbor cities of each city.

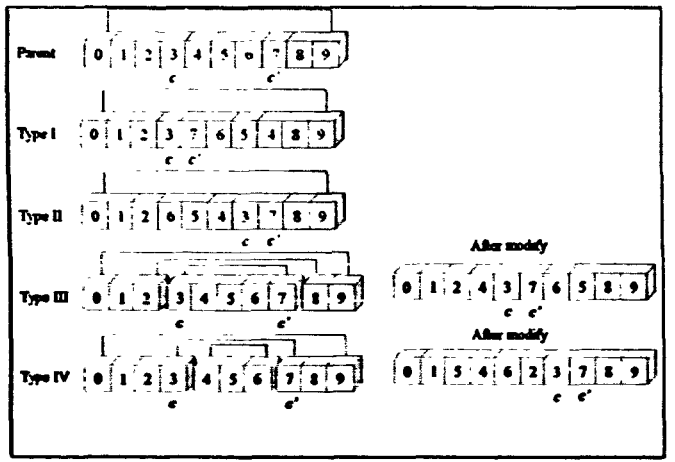
Figure 2(a) outlines the steps of the NJ mutation. By giving the input of an individual s_i and the family competition length L, the NJ mutation generates L children from the start solution s_i. The NJ mutation applies the following steps to generate a child: First a city c is randomly selected from s_i. With equal probability, another city c' is randomly selected either from the geometrically nearest three neighbors of the city c or the neighbor cities of c of another individual, which is randomly selected from the population. If the edge between cities c and c' is not in s_i, connect the cities c and c' and generates four possible types shown in Figure 2(b). The NJ mutation generates four candidates by sequentially executing each type. Among these four candidates and s_i, only the one with the best objective value is selected as the parent of the next loop of the NJ mutation.

For Types I and II, the invert operator is used to connect the cities c and c' as shown in Figure 2(b). For Types III and IV, a greedy method was applied to merge two disjoint subtours into a valid solution. The greedy method works as follows: Let v_i represent a city, (v_i, v_j), $i \neq j$, represent an edge, and $w(v_i, v_j)$ be the edge length of (v_i, v_j). At the same time, let (v_r, v_{r+1}) and (v_s, v_{s+1}) be the edges of the subtours G_r and G_s, respectively. We find a pair of new edges, (v_r, v_{s+1}) and (v_s, v_{r+1}) to connect these two subtours, G_r and G_s, into a legal tour by maximizing the value of the following equation

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

1. Input: family competition length L and an individual s_i
2. **repeat** L times
   - 2.1 randomly select a city c from s_i
   - 2.2 if ($rand() > 0.5$)
     - Randomly select another individual from current population. Randomly select the city c' connecting to c in the selected individual.
     - else randomly select the city c' from the geometric nearest three cities of c.
   - 2.3 if the city c' doesn't connect to c in s_i
     - a. Generate four candidates by connecting c' and c (see Fig. 2.(b)).
     - b. Select the one with lowest fitness from s_i and these four candidates to replace s_i as the parent of next loop.
3. **return** s_i

(a)



(b)

FIGURE 2(a) The neighbor-join mutation algorithm. (b) Four types of connecting cities c and c' in the NJ mutation The results of Type I and Type II are obtained via the invert operation. The results of Type III and Type IV are constructed by applying a greedy method (see text) from two subtours.

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 form the new solution. In fact, only the nearest 20 cities of each city are considered.

3 SOME CHARACTERISTICS OF HSGA

This section discusses some characteristics of the HSGA; first the parameter settings, then the mechanisms of keeping the population diversity of HSGA with HpS and random pair selection (RpS) are analyzed. RpS, the original selection mechanism of the EAX genetic algorithm [20], is considered as an efficient mechanism for maintaining the population diversity [28]. Finally the search behaviors of the HSGA with NJ mutation and 2-opt are discussed.

For ease of analysis, let $HSGA(p_1, p_2, p_3)$ denote that the HSGA used the p_1 crossover operator, the p_2 selection, and the p_3 mutation operator. For example, *HSGA(EAX,HpS, NJ)*, or HSGA, is the proposed approach applying the EAX crossover, the *HpS* section, and the NJ mutation. *HSGA(EAX,RpS,None)* represents the original EAX genetic algorithm, using the EAX crossover and random pair selection. "*None*" denotes that no operator is used in the respective operator. Each method with different operators has been implemented in C++ and executed on a Pentium III 500 MHz personal computer with single processor. Table I gives the test problem names selected from TSPLIB [24] along with number of cities and the optimal tour lengths.

3.1 Parameter Settings

As introduced in Section 2, the population size (N) and the family competition length (L) are the main parameters in the algorithm. To decide the parameter values, various values of these two parameters were tested on 15 TSP problems selected from TSPLIB [24]. Figure 3 shows the relationship between the population size and the average error rate on problems *att532* and *fnl4461*. The required CPU time is proportional to the population size and the average error rate is reduced when the population size increases. Generally, the HSGA has similar curves for all testing problems. As a result of the experiments, the population size is set to the number of cities of a TSP when the number of cities is smaller than 1000, and is set to half the number of cities of a TSP when the number of cities is larger than 1000, as a tradeoff between solution quality and convergence time.

TABLE I The Tested Problem Names with Number of Cities and Optimal Tour Lengths.

Problem name	*No. of cities*	*Optimal tour length*
eil101	101	629
kroa200	200	29,368
lin318	318	42,029
pcb442	442	50,778
att532	532	27,686
u574	574	36,905
rat575	575	6773
u724	724	41,910
rat783	783	8806
vm1084	1084	239,297
pcb1173	1173	56,892
u1432	1432	152,970
vm1748	1748	336,556
pr2392	2392	378,032
pcb3038	3038	137,694

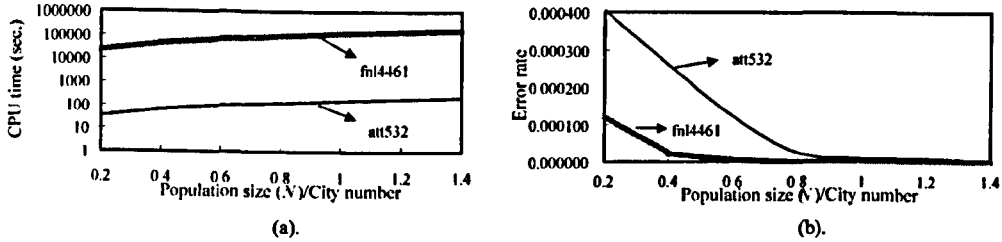


FIGURE 3 The relationships among the average time, average error rate, and the population size of HSGA tested on problems *att532* and *fnl4461*. (a) is the relation between the population size and the average time and (b) is the relation between the population size and the average error rate above the optima.

To understand the influence of the family competition length (L) in the EAX and the NJ operators, the performance of the HSGA was observed with various lengths. Figure 4 shows the relationship between the performance and the family competition length (L) on 15 TSP problems where the values of L are set to 5, 20, and 50. Each problem was tested in 30 distinct trials. For each problem the HSGA has the worst performance when L is 5. The improvement of the solution quality of the HSGA is not significant when L exceeds 20. In practice, the longer L is, more time is required for the HSGA. Therefore, L is set to 20 in this paper.

3.2 Analysis of the HpS

To investigate the ability of the HpS for keeping the population diversity, the edge entropy and the average edge similarity of a population were used to analyze the search behavior of the HSGA with different selection mechanisms, *HSGA(EAX,RpS,none)* and *HSGA(EAX,HpS,none)*. The edge entropy of a population is given as

$$-\sum_{e \in Y} \frac{F(e)}{N} \log_2 \frac{F(e)}{N} \tag{3}$$

where $Y = \{E(s_1) \cup E(s_2) \cup \cdots \cup E(s_N)\}$, $F(e)$ is the count of the number of appearances of edge e in the current population, and N is the population size. A larger value of the edge entropy

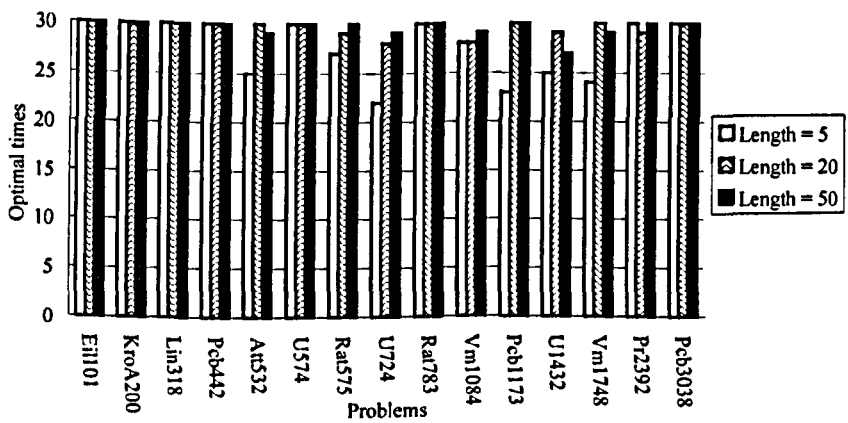


FIGURE 4 Performance of the *HSGA* on 15 TSP problems with different search lengths 5, 20, and 50 based on the number finding the optimal solution in 30 trials. The HSGA has the worst performance when L is 5 and has similar performance when L is 20 and 50.

implies that the population diversity is higher. Figure 5(a) shows the relationships between the values of the edge entropy and the number of generations of *HSGA(EAX,RpS,none)* and *HSGA(EAX,HpS,none)* on the problems *att532* and *fnl4461*. Although these two methods have similar trends, the value of the edge entropy of *HSGA(EAX,HpS,none)* is decreasing more slowly than that of *HSGA(EAX,RpS,none)*. In Figure 5(a), the value of the edge entropy of *HSGA(EAX,RpS,none)* approaches zero when the number of generation exceeds 75 in problem *att532* and 180 in problem *fnl4461*. On the other hand, the edge entropy of *HSGA(EAX,HpS,none)* is greater than that of *HSGA(EAX,RpS,none)* in both problems. This result implies that the HpS is more powerful than the RpS for keeping population diversity.

To further analyze the HpS effect on population diversity, the average edge similarity of a population is given as

$$\frac{2}{N(N-1)}\sum_{i}^{N}\sum_{j=1,j\neq i}^{N}\|T_{i,j}\| \tag{4}$$

where N is the population size and $\|T_{i,j}\|$ is the number of identical edges of two individuals (s_i and s_j). Figure 5(b) shows that the value of the edge similarity obtained by *HSGA(EAX,HpS,none)* increases more slowly than that obtained by *HSGA(EAX,RpS,none)* on the problems *att532* and *fnl4461*. Almost all individuals in a population generated by *HSGA(EAX,RpS,none)* are the same when the number of the generations exceeds 70 in problem *att532* and 180 in problem *fnl4461*. Again *HSGA(EAX,HpS,none)* is able to retain the diversity of a population.

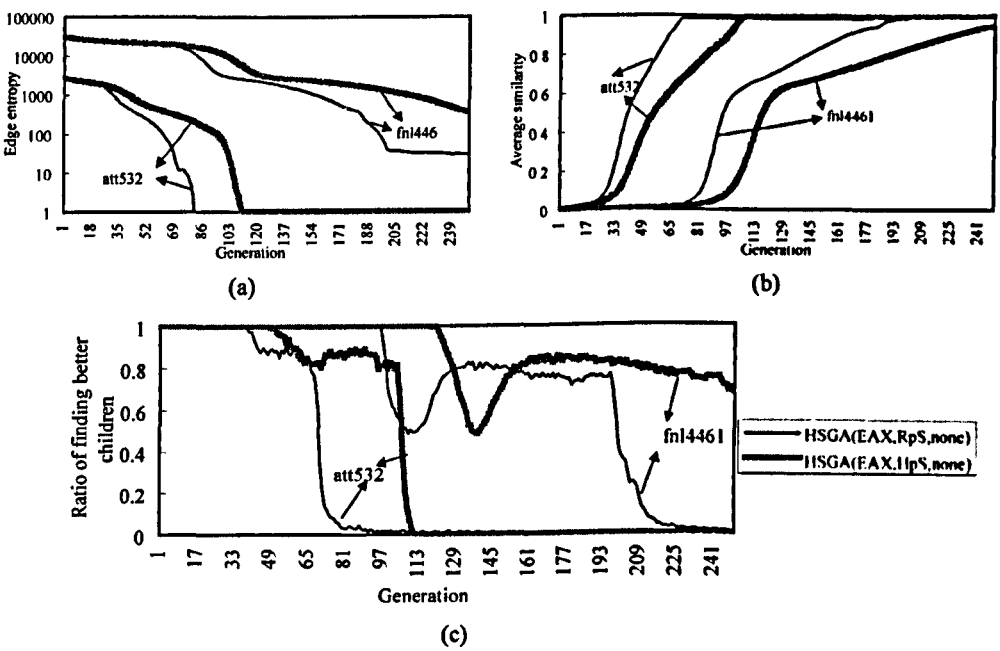


FIGURE 5 Comparisons of the edge entropy (see Eq. (3)), the edge similarities of a population (see Eq. (4)), and the ability of generating better children of *HSGA(EAX,RpS,none)* and *HSGA(EAX,HpS,none)* on the problems *att532* and *fnl4461*. (a) The edge entropy of *HSGA(EAX,RpS,none)* declines faster than that of *HSGA(EAX,HpS,none)*. The larger value of the edge entropy indicates that the population diversity is higher. (b) The average value of the edge similarity of *HSGA(EAX,RpS,none)* grows faster than the latter. (c) The curve of the *HSGA(EAX,RpS,none)* decreases steeply when the number of generations is over 60 in problem *att532* and 190 in *fnl4461*, but the *HSGA(EAX,HpS,none)* is able to keep the probability over 80%.

TABLE II Comparisons Between *HSGA(EAX,RpS,none)* and *HSGA(EAX,HpS,none)* on 15 TSP Problems Taken from TSPLIB [24] Based on the Average CPU Time (Time), the Number of Trials Finding the Optimal Solution (Opt Times), and Average Solution Qualities (Error) in 30 Trials.

	HSGA(EAX,HpS,none)			*HSGA(EAX,RpS,none)*		
Problem	*Time (sec.)*	*Opt. times*	*Error (%)*	*Time (sec.)*	*Opt. times*	*Error (%)*
eil101	1.06	30	0.0000	0.64	30	0.0000
kroa200	8.96	30	0.0000	5.77	30	0.0000
lin318	30.99	29	0.0079	19.13	26	0.0291
pcb442	56.6	30	0.0000	35.89	30	0.0000
att532	160.01	29	0.0008	102.97	7	0.0373
u574	172.32	30	0.0000	106.37	30	0.0000
rat575	198.69	29	0.0005	130.63	21	0.0043
u724	353.22	27	0.0025	234.08	18	0.0122
rat783	515.48	30	0.0000	350.67	30	0.0000
vm1084	936.86	28	0.0022	655.15	18	0.0277
pcb1173	1123.53	30	0.0000	769.54	24	0.0050
u1432	1714.28	25	0.0019	1214.08	16	0.0145
vm1748	4336.53	30	0.0000	3142.41	19	0.0189
pr2392	8635.32	30	0.0000	6081.21	21	0.0049
pcb3038	19453.3	30	0.0000	13965.04	24	0.0071

Here the error (%) is defined as (*average-optimum/optimum* where *average* is the average value of the best solutions obtained by both methods.

Figure 5(c) shows the probabilities of generating better children in each generation by *HSGA(EAX,RpS,none)* and *HSGA(EAX,HpS,none)* on the problems *att532* and *fnl4461*. The probabilities of both methods are near 100% in the early stages. However, the probability of *HSGA(EAX,RpS,none)* declines steeply, while the probability of *HSGA(EAX,HpS,none)* is kept at about 80% even when it has exhausted its search steps.

Finally the solution quality is a critical factor for evaluating the power of applying the HpS with the EAX operator. Table II summarizes the results of both *HSGA(EAX,RpS,none)* and *HSGA(EAX,HpS,none)* on 15 TSPs. On average the solution quality of the latter is better and more stable than that of the former. *HSGA(EAX,RpS,none)* is not stable, especially, for hard problems, such as problems *att532* and *rat575*. In contrast, the solutions of *HSGA(EAX,HpS,none)* are very near the optimum for all testing problems.

In summary, the tests demonstrated that the HpS is able to improve the solution quality for the EAX via maintenance of population diversity and provision of a good pairing scheme. It is believed that the HpS may improve the performance for most crossover operators and the HpS scheme will be tested on more crossover operators in the near future.

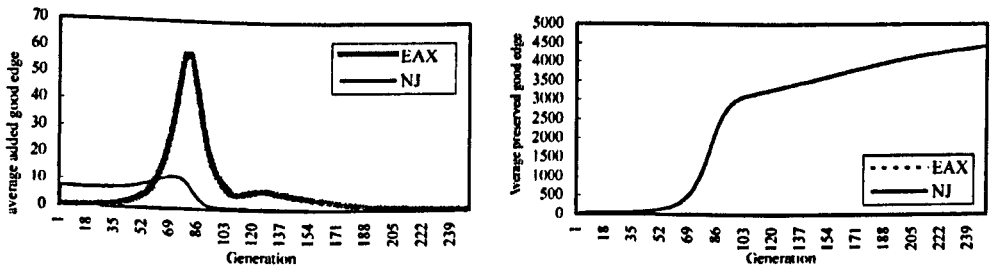


FIGURE 6 Comparisons of the ability of adding and preserving good edges of *EAX* and *NJ* in HSGA on the problem *fnl4461*. A good edge is defined as an edge appearing in the optimal tour. (a) and (b) are the comparison of average added good edges and average preserved good edges by *EAX* and *NJ*, respectively.

TABLE III Comparisons of Various Approaches of our Method Applying Different Operators: *HSGA(EAX,HpS,NJ)*, *HSGA(EAX,HpS,2-opt)*, *HSGA(EAX,RpS,NJ)*, and *HSGA(EAX,RpS,2-opt)*.

	HSGA(EAX,HpS,NJ)			*HSGA(EAX,HpS,2-opt)*			*HSGA(EAX,RpS,NJ)*			*HSGA(EAX,RpS,2-opt)*		
Problem	*Time (sec.)*	*Opt times*	*Error (%)*	*Time (sec.)*	*Opt times*	*Error (%)*	*Time (sec.)*	*Opt times*	*Error (%)*	*Time (sec.)*	*Opt times*	*Error (%)*
eil101	1.07	30	0.0000	1.08	30	0.0000	0.64	30	0.0000	0.65	30	0.0000
kroa200	9.03	30	0.0000	9.36	30	0.0000	5.79	30	0.0000	5.51	30	0.0000
lin318	31.33	30	0.0000	31.64	29	0.0079	19.18	30	0.0000	19.34	27	0.0191
pcb442	56.94	30	0.0000	57.45	30	0.0000	35.96	30	0.0000	37.15	30	0.0000
att532	161.93	30	0.0000	167.21	28	0.0017	103.13	24	0.0090	115.32	11	0.0312
u574	173.87	30	0.0000	176.35	30	0.0000	106.41	30	0.0000	108.86	30	0.0000
rat575	198.95	30	0.0000	202.47	29	0.0005	130.61	28	0.0009	129.57	23	0.0031
u724	355.34	29	0.0005	360.76	23	0.0051	233.91	27	0.0019	227.55	16	0.0059
rat783	523.21	30	0.0000	524.76	30	0.0000	350.19	30	0.0000	358.03	30	0.0000
vm1084	949.98	29	0.0016	957.47	23	0.0558	655.52	21	0.0227	669.56	20	0.0099
pcb1173	1143.75	30	0.0000	1144.88	23	0.0021	771.85	29	0.0005	763.31	27	0.0005
u1432	1734.85	27	0.0034	1767.42	18	0.0068	1175.05	26	0.0080	1216.63	24	0.0038
vm1748	4418.92	30	0.0000	4423.26	24	0.0456	3053.64	27	0.0074	3205.26	21	0.0125
pr2392	8773.49	30	0.0000	8818.56	29	0.0003	6070.45	30	0.0000	6017.36	28	0.0009
pcb3038	19865.7	30	0.0000	19842.3	30	0.0000	13563.5	28	0.0004	13962.2	25	0.0062

These methods are tested on 15 TSP problems based on the average CPU time (time), the number of trials that found optimal solutions (opt times), and average solution quality (error) in 30 trials. Here the error (%) is defined as *(average-optimum)/optimum* where *average* is the average value of the best solutions obtained by the test approaches.

3.3 Analysis of Combining EAX and NJ

To examine the complementary characteristics of *EAX* and *NJ*, the abilities of adding and preserving "good edges" (*i.e.* the edges in the optimal tour) of the *EAX* and *NJ* operators was measured. Figure 6 shows the results of *EAX* and *NJ* for adding and preserving "good edges". Figure 6(a) shows that *NJ* can add more "good edges" than EAX in the early stages, while *EAX* outperforms *NJ* later. The abilities of preserving "good edges" of these two operators are similar (Fig. 6(b)), indicating that they able to keep "good edges". These two operators can add "good edges" without reducing the ability of preserving "good edges". Although the advantages of combining *EAX* and *NJ* cannot be theoretically proved, they indeed assisted each other in adding and preserving "good edges" in our experiments.

To further investigate the robustness of incorporating EAX and NJ, the performance of our method in combining 2-opt and the NJ mutation with the EAX operator was analyzed. 2-opt is also applied as 20 single 2-opt steps. Table III shows the results of the methods applying different operators for 15 TSP problems. According to Tables II and III, the methods combining 2-opt into the EAX, *i.e. HSGA(EAX,RpS,2-opt)* and *HSGA(EAX,HpS,2-opt)*, are limited in respect of improving the solution quality. On the other hand, the methods combining the NJ mutation and the EAX, *i.e. HSGA(EAX,RpS,NJ)* and *HSGA(EAX,HpS,NJ)*, significantly improve the solution qualities for all test problems. For example, *HSGA(EAX,RpS,NJ)* and *HSGA(EAX,HpS,NJ)* perform better than *HSGA(EAX,RpS,none)* and *HSGA(EAX,HpS, none)*, respectively.

In this section, we have demonstrated the robustness and adaptability of the HSGA for exploring the search space of TSPs. The key novelty of the present work is the seamless ability of the HSGA to integrate global and local search mechanisms through incorporation of a number of genetic operators and selections, each with unique search mechanisms.

4 COMPUTATIONAL RESULTS

Following the detailed discussion of the HSGA in the last section, the HSGA was compared with five stochastic methods which were efficient approaches for TSPs in our surveys. These methods were tested on five TSP benchmark problems, including *lin318*, *pcb442*, *att532*, *rat783*, and *pcb3038*, shown in Table IV because they have been widely used to compare the performance among different algorithms. Our proposed algorithm, *HSGA(EAX, HpS,NJ)*, was executed in 30 independent runs for each problem.

Table IV summarizes the results of our method and these five approaches, including the ant colony system (ACS) [6], the distance-preserving crossover genetic algorithm (DGA) [8], the nature crossover genetic algorithm (NGA) [13], the compact genetic algorithm [3], and the EAX genetic algorithm (EGA) [20]. ACS is an ant colony system combined with the 3-opt operator; DGA combined the distance-preserving crossover and the Lin–Kernighan neighborhood; NGA integrated the nature crossover and LK local search [18]; CGA mimicked the existence of solutions and combined LK local search; and EGA used the EAX crossover. The results of the first four methods were directly summarized from Refs. [3, 6, 13]. EGA was implemented based on the original paper to obtain the results which are slightly better than the results in the original paper [20].

The best tour length, average tour length, and the group standard deviation of trials were used to measure the performance of the comparative methods. The values in parentheses of the best and the average tour length represent the percentage error defined as *(average-optimum)/optimum*, where *average* is the experimental value and *optimum* is the optimum of a TSP problem. The *average CPU time* is only for reference because each approach is executed on different machines. Our HSGA seems slower than other approaches for large TSPs, such as the problem *pcb3038*.

TABLE IV Comparisons of our Method (HSGA) with Other Methods, ACS [6], DGA [8], NGA [13], CGA [3], and EGA[20] on Five TSP Problems Based on the Best Tour Length, Average Tour Length, and the Standard Deviation.

Problems	*Methods*	*Best (error %)*	*Average (error %)*	*Standard deviation*	*Average CPU time*
lin318	ACS	42,029(0.000)	42,029(0.000)	0.00	537
	DGA	42,029(0.000)	42033.44(0.011)	13.5	35
	NGA	42,029(0.000)	42029.00(0.000)	0.00	36
	CGA	42,029(0.000)	42029.00(0.000)	0.00	12
	EGA	42,029(0.000)	42041.23(0.029)	17.1	19
	HSGA	42,029(0.000)	42029.00(0.000)	0.00	31
pcb442	ACS	N/A	N/A	N/A	N/A
	DGA	50,778(0.000)	50,778(0.000)	0.00	53
	NGA	50,778(0.000)	50,778(0.000)	0.00	31
	CGA	50,778(0.000)	50,778(0.000)	0.00	22
	EGA	50,778(0.000)	50,778(0.000)	0.00	36
	HSGA	50,778(0.000)	50,778(0.000)	0.00	56
att532	ACS	27,693(0.000)	27718.20(0.112)	N/A	810
	DGA	27,686(0.000)	27697.58(0.042)	4.80	106
	NGA	27,686(0.000)	27695.61(0.035)	7.10	76
	CGA	27,686(0.000)	27686.00(0.000)	0.00	112
	EGA	27,686(0.000)	27696.33(0.037)	7.92	102
	HSGA	27,686(0.000)	27686.00(0.000)	0.00	160
rat783	ACS	8818(0.136)	8837.90(0.362)	N/A	1280
	DGA	8806(0.000)	8806.00(0.000)	0.00	53
	NGA	8806(0.000)	8806.00(0.000)	0.00	35
	CGA	8806(0.000)	8806.00(0.000)	0.00	111
	EGA	8806(0.000)	8806.00(0.000)	0.00	351
	HSGA	8806(0.000)	8806.00(0.000)	0.00	515
pcb3038	ACS	N/A	N/A	N/A	N/A
	DGA	137,705(0.008)	137760.55(0.048)	42.8	1880
	NGA	137,695(0.001)	137765.02(0.052)	45.5	816
	CGA	N/A	N/A	N/A	N/A
	EGA	137,694(0.000)	137703.77(0.007)	9.33	13,965
	HSGA	137,694(0.000)	137694.00(0.000)	0.00	19,453

"N/A" represents not available in the original papers. Here the error (%) is defined as (*average-optimum*)/*optimum* where *average* is the average values of the best solutions obtained by the test approaches.

Table IV shows that the HSGA performs more robustly than the comparative methods on the test problems. The HSGA is able to find the optimum of the five tested benchmarks for each trial. On the other hand, except for CGA, the other comparative approaches are not stable for hard problems, such as problems *att532* and *pcb3038*.

To show the robustness of HSGA on large TSPs, HSGA was compared with some LK-based heuristic methods, including Concorde LK [1], chained LK (CLK) [1], Johnson LK [12], iterative LK (ILK) [12], and Tabu search with LK [34], as shown in Table V. These five approaches 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/). All parameters settings of our method follow the descriptions in Section III (A) except for *usa13509*, whose population size is set to 4000 due to the memory size.

Table V shows that HSGA outperforms other LK-based approaches in the test problems. The HSGA is able to find the optimum, and the average solution quality is within 0.00048 above the optimum value for each test problem although the HSGA is somewhat slower than these approaches. For the larger problem such as *usa13509*, ILK is about 50 times faster than HSGA with population size equal to 4000. Fortunately, the running time for HSGA is about the same as ILK and the average tour length is 20014159 (0.001566) which is slightly better than ILK when the population size is 100.

TABLE V Comparisons of our Method (HSGA) with LK-based Methods, Concorde LK [1], Chained LK (CLK) [1], Johnson LK [12], Iterative LK (ILK) [12], and Tabu Search with LK [34] on Eight Larger TSP Problems Based on the Average Tour Length.

Problem	*HSGA*	*Concorde LK*	*Johnson LK*	*ILK*	*CLK*	*Tabu with LK*
vm1084	239,300	245,931	241,449	239,349	239,301	240,238
(239297)	(0.000016)	(0.027723)	(0.008993)	(0.000217)	(0.000017)	(0.003932)
pcb1173	56,892	58,110	57,388	56,897	56,984	57,290
(56892)	(optimal)	(0.021409)	(0.008718)	(0.000088)	(0.001617)	(0.006996)
u1432	152,975	156,827	155,386	153,122	153,328	153,727
(152970)	(0.000034)	(0.025214)	(0.015794)	(0.000994)	(0.002340)	(0.004949)
vm1748	336,556	340,206	338,917	336,556	336,721	337,683
(336556)	(optimal)	(0.010845)	(0.007015)	(optimal)	(0.000490)	(0.003349)
pr2392	378,032	390,510	385,029	378,597	379,629	380,486
(378032)	(optimal)	(0.033008)	(0.018509)	(0.001495)	(0.004225)	(0.006492)
pcb3038	137,694	140,668	139,115	137,861	138,055	138,893
(137694)	(optimal)	(0.021599)	(0.010320)	(0.001213)	(0.002622)	(0.008708)
fnl4461	182,567	185,771	184,624	182,814	182,840	184,373
(182566)	(0.000005)	(0.017555)	(0.011273)	(0.001358)	(0.001501)	(0.009898)
usa13509	19,992,528	20,486,590	20,236,820	20,015,598	20,022,550	20,160,648
(19982859)	(0.000484)	(0.025208)	(0.012709)	(0.001638)	(0.001986)	(0.008897)

The Length of tour is given without brackets and that with error is given in brackets. Here the Error is Defined as (*average-optimum*)/*optimum* where *average* is the Average Value of the Best Solutions Obtained by the Test Approaches.

5 CONCLUSION

This study has demonstrated that the HSGA is a stable approach for TSPs. From our experience, we suggest that a global optimization method for TSPs should consist of both global and local search strategies as well as implementing the mechanisms of preserving good edges and inserting new edges into offspring. In our approach, the edge assembly crossover and heterogeneous edge selection are global search strategies; the family competition and the neighbor-join mutation are local search strategies. Our experiments indicated that the edge assembly crossover and the neighbor-join mutation are able to preserve good edges and add new edges. These strategies seem to be able to cooperate closely with each other to improve the overall search performance.

Experiments on 17 benchmark TSPs have verified that the proposed approach is robust and is very competitive with algorithms from the best surveys. Our approach is able to find stable optimum solutions for all test TSPs; specifically, it found the optimum over 27 times in 30 independent runs for hard problems, such as *att532*, *vm1432*, and *pcb3038*. We believe that the robustness of our approach makes it an effective tool for TSPs and potential applications.

In the future, the HSGA research will be pursued in three directions: (1) development of the HpS with some well-known crossover operators; (2) development of the HSGA on several bioinformatics applications; and (3) study of a more diverse set of TSPs to determine the limits of our HSGA.

References

[1] Applegate, D., Bixby, R., Chvatal, V. and Cook, W. J. (1999). Finding tours in the TSP. *Tech. Rep. TR99-05*, Dept. Comput. Appl. Math., Rice Univ., Houston, TX 77005.

[2] Alizadeh, F., Karp, R. M., Newberg, L. A. and Weisser, D. K. (1993). Physical mapping of chromosomes: a combinatorial problem in molecular biology. *Proceeding of the Fourth ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 52–76.

[3] Baraglia, R., Hidalgo, J. I. and Perego, R. (2001). A hybrid heuristic for the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 5(6), 613–622.

[4] Baeck, T. (1996). *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York.

[5] Chen, S. H., Smith, S. F. and Guerra, S. C. (1999). The GENIE is out! (Who needs fitness to evolve?). *Proceedings of the Congress of Evolutionary Computation (CEC)*, 2102–2106.
[6] Dorigo, M. and Gambardella, L. M. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, **1**(1), 53–66.
[7] Fogel, D. B. (1995). *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence.* IEEE Press, Piscataway, NJ.
[8] Freisleben, B. and Merz, P. (1996). New genetic local search operators for the traveling salesman problem. *Parallel Problem Solving from Nature IV (PPSN IV)*, 890–899.
[9] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, MA.
[10] Hart, W. E. (1994). Adaptive global optimization with local search. *PhD thesis*, University of California, San Diego.
[11] Hopfield, J. J. and Tank, D. W. (1985). Neural computation of decisions in optimization problems. *Biological Cybernetics*, **52**, 141–152.
[12] Johnson, D. S. and McGeoch, L. A. (1997). The traveling salesman problem: A case study in local optimization. In: Aarts, E. H. L. and Lenstra, J. K. (Eds.), *Local Search in Combinatorial Optimization.* John Wiley and Sons, Ltd., pp. 215–310.
[13] Jung, S. and Moon, B. R. (2000). The nature crossover for the 2D Euclidean TSP. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 1003–1010.
[14] Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, **200**, 671–680.
[15] Korostensky, C. and Gonnet, G. H. (2000). Using traveling salesman problem algorithms for evolutionary tree construction. *Bioinformatics*, **16**(7), 619–627.
[16] Krasnogor, N. and Smith, J. (2000). A memetic algorithm with self-adaptive local search: Tsp as a case study. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 987–994.
[17] Lin, S. (1965). Computer solutions of the traveling salesman problem. *Bell Systems Technical Journal*, **23**, 2245–2269.
[18] Lin, S. and Kernighan, B. (1973). An effective heuristic algorithms for the traveling salesman problem. *Operations Research*, **21**, 498–516.
[19] Mulhem, M. and Maghrabi, T. (1998). Efficient convex-elastic net algorithm to solve the Euclidean traveling salesman problem. *IEEE Transactions on Systems, Man and Cybernetics -Part B*, **28**(4), 618–620.
[20] Nagata, Y. and Kobayashi, S. (1997). Edge assembly crossover: A high-power genetic algorithm for the traveling salesman problem. *Proceeding of the Seventh International Conference on Genetic Algorithms (ICGA)*, 450–457.
[21] Nagata, Y. and Kobayashi, S. (1999). An analysis of edge assembly crossover for the traveling salesman problem. *IEEE International Conference on Systems Man and Cybernetics*, pp. 628–633.
[22] Nagata, Y. and Kobayashi, S. (1999). A proposal and evaluation of new crossover "Edge Assembly Crossover" for the traveling salesman problem. *Journal of Japanese Society for Artificial Intelligence*, **14**(5), 848–859.
[23] Padberg, M. and Rinaldi, G. (1987). Optimization of a 532-city symmetric traveling salesman problem by branch and cut. *Operation Research Letters*, **6**, 1–7.
[24] Reinelt, G. (1991). TSPLIB- A traveling salesman problem library. *ORSA Journal on Computing*, **3**(4), 376–384.
[25] Stützle, T. and Dorigo, M. (1999). ACO algorithms for the traveling salesman problem. *Evolutionary Algorithms in Engineering and Computer Science*, 163–183.
[26] Tao, G. and Michalewicz, Z. (1998). Inver-over operator for the TSP. *Parallel Problem Solving from Nature V (PPSN V)*, 803–812.
[27] Tsai, H. K., Yang, J. M. and Kao, C. Y. (2001). A genetic algorithm for traveling salesman problems. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 687–693.
[28] Watson, J., Ross, C., Eisele, V., Denton, J., Bins, J., Guerra, C., Whitely, D. and Howe, A. (1998). The traveling salesrep problem, edge assembly crossover, and 2-opt. *Parallel Problem Solving from Nature V (PPSN V)*, 823–832.
[29] Yang, J. M., Horng, J. T. and Kao, C. Y. (2000). A genetic algorithm with adaptive mutations and family competition for training neural networks. *International Journal of Neural Systems*, **10**(5), 333–352.
[30] Yang, J. M., Horng, J. T., Lin, C. J. and Kao, C. Y. (2001). Optical coating designs using an evolutionary algorithm. *Evolutionary Computation*, **9**(4), 421–443.
[31] Yang, J. M. and Kao, C. Y. (2000). Integrating adaptive mutations and family competition into genetic algorithms as function optimizer. *Soft Computing*, **4**(2), 89–102.
[32] Yang, J. M. and Kao, C. Y. (2000). A family competition evolutionary algorithm for automated docking of flexible ligands to proteins. *IEEE Transactions on Information Technology in Biomedicine*, **4**(3), 225–237.
[33] Yen, J., Yip, J. C. and Pao, Y. H. (1998). Combinatorial optimization with use of guided evolutionary simulated annealing. *IEEE Transactions on Systems, Mans, and Cybernetics -Part B*, **28**(2), 173–191.
[34] Zachariasen, M. and Dam, M. (1995). Tabu search on the geometric traveling salesman problem. *Proceedings from Metaheuristics International Conference*, pp. 571–587.
[35] Zhenya, H., Chengjian, W., Bingyao, J., Wenjiang, P. and Luxi, Y. (1999). A new population-based incremental learning method for the traveling salesman problem. *Proceedings of the Congress on Evolutionary Computation (CEC)*, 1152–1156.