An Evolutionary Multiple Heuristic with Genetic Local Search for Solving TSP

Peng Gang[1], Ichiro Iimura[2], and Shigeru Nakayama[3]

[1] Department of Information and Computer Science,
Faculty of Engineering, Kagoshima University
1-21-40 Korimoto, Kagoshima 890-0065 Japan
gang901@hotmail.com

[2] Department of Administration, Faculty of Administration,
Prefectural University of Kumamoto
3-1-100 Tsukide, Kumamoto 862-8502 Japan
iiimura@pu-kumamoto.ac.jp

[3] Department of Information and Computer Science,
Faculty of Engineering, Kagoshima University
1-21-40 Korimoto, Kagoshima 890-0065 Japan
shignaka@ics.kagoshima-u.ac.jp

Abstract

Traveling salesman problem (TSP) is well studied as one of the combinatorial optimization problems. In this paper we propose an evolutionary multiple heuristic combined with Genetic Local Search (GLS) for solving TSP. Two main operations of Complete 2-Opt (C2Opt) and Smallest Square (SS) are combined in the new algorithm and applied to solve TSP. Another two operations of Deletion and Best Part Collector (BPC) are discussed. A reasonable result is presented by combining these operations with genetic operators in the proposed algorithm for the TSP in our experiments.

Keyword: Heuristic, Genetic Local Search (GLS), Traveling Salesman Problem (TSP), Fractal TSP.

I. Introduction

The Genetic Algorithm (GA) is an optimizing algorithm that models the processes of natural evolution [1]. The traditional GA usually consists of some operators such as crossover, selection and mutation which are simulated from biological and genetic processes. However, the traditional GA is inefficient for solving large optimization problems. It is well known that most of the successful algorithms have often incorporated the GLS [2], [3], [4], [5], [6].

The GA is applied to the TSP which is a well known and important combinatorial optimization problem [7], [8]. In the TSP, each distance between two cities is given for a set of n cities. The goal is to find the shortest tour that visits each city exactly once and then returns to the starting city. Many books and papers introduce the procedure of how to find the shortest tour usually called as the optimum solution in TSP search algorithms [9], [10], [11]. There are currently three general classes of heuristics for solving TSP: classical tour construction heuristics such as the Nearest Neighbor heuristic, the Greedy algorithm and local search algorithms based on re-arranging segments of the tour [12]. The nearest-neighbor heuristic can be solved easily in quadratic time and linear space. However, Bentley [13] has shown that this simple technique is outperformed by other, seemingly harder to compute methods, such as the multiple fragments heuristic and cheapest insertion [14].

Multiple fragments heuristic consider all edges one at a time in sorted order, and include an edge if it connects the endpoints of two fragments of tours connected components of previously added edges. Cheapest insertion maintains a tour of a subset of the sites, and at each step adds a site by replacing an edge of the tour by two edges through the new site. Each successive insertion is chosen as the one causing the least additional length in the augmented tour. These methods concentrate on reconstruction of the edges in the TSP tours.

In this paper we propose an evolutionary multiple heuristics combined with Genetic Local Search (GLS) to solve the TSP. Two main operations of Complete 2-Opt (C2Opt) and Smallest Square (SS) are combined in our algorithm and applied to the TSP. The C2Opt is based on the 2-Opt heuristic search, and can remove all crossed edges in the tour if the repetition is sufficiently great. The SS selects shorter edges than the C2Opt. The problem with the SS is that the city orders of the original tour are changed whilst it is applied. Therefore, the crossed edges cannot be removed completely. However, it presents a good result combining the C2Opt and the SS with genetic operators for the TSP in our experiments. Another two operations of Deletion and Best Part Collector (BPC) are employed in the proposed algorithm. The Deletion is effective for removing duplicates from the population and the BPC is effective for collecting the best part among the individuals to cultivate the elite individual.

II. Outlines of Evolutionary Multiple Heuristics

The heuristic 2-Opt has been used to optimize TSP tours in connection with the GA [8]. Usually, the 2-Opt is randomly applied by the GA to 20–30% of the individuals in a population. K. Mathias et al. made a discussion on the topic of 2-Opt [15]. As described in their paper, the 2-Opt removes two edges in a tour, and then one of the resultant segments is reversed and the two segments are reconnected. If the 2-Opt results in an improved tour, the change is kept. Otherwise, the tour is returned to its original form. The 2-Opt is typically applied to all $n(n-1)/2$ pairs of edges. If one or more improvements are found, the process can be re-applied to the set of all edges. When no more further improvements are found, the tour has converged on a local optimum with respect to the 2-Opt. K. Mathias et al. also stated that it is computationally expensive to run the 2-Opt until convergence [15].

In our previous works, we presented a Multiple Heuristic Search Algorithm and applied it to the TSP [16], [17], [18]. The result shows that the algorithm is efficient for local improvement in the TSP instances, but it just emphasizes on the performance of the multiple search process which uses some local search techniques. Since no genetic operators are employed to the search process, it is not an evolutionary algorithm and easily reaches the local convergence. This paper discusses an evolutionary multiple heuristics algorithm combined with the genetic operators based on our previous works. The genetic operators, such as crossover and mutation, explore the diversity of search space and additionally improve the convergence speed to avoid converging prematurely to some sub-optimal solutions. The operations of C2Opt and SS are combined in the algorithm for improving tours in the search space. The C2Opt means the 2-Opt is completely applied to an initial tour according to the operations described in section 3, until no crossed edges are found in the tour. The C2Opt is based on the 2-Opt heuristic search method. The tour created by the C2Opt usually is a local optimum in which, probably, some of the city orders are the same as they are in the optimal tour. This is a desirable attribute for local search operations.

On the other hand, some shorter edges might have been missed while the 2-Opt is being applied to remove the two crossed edges. In a TSP tour, two edges determine a square except for the neighboring ones. The 2-Opt only checks crossed edges and a pair of side edges of a square. Hence, another pair of side edges will be missed. The SS checks the crossed edges and the two pairs of the side edges of a square which is the first square in our search algorithm, and then it chooses the shortest pairs to improve the tour. If the tour is separated as two sub tours, another two edges which form the second square between the two sub tours must be found to reconnect the tour. The two edges should not make the tour longer than the previous one. If such edges do not exist, the 2-Opt is applied to the first square. In the algorithm, the SS creates a tour shorter than C2Opt, but it cannot completely remove all the crossed edges in the tour, because the original city orders are changed whilst the tour is reconnected in the second square. A better tour can be created if the C2Opt is set after the SS (which we refer to as SS+C2Opt).

Another two operators used in our algorithm are Deletion and Best Part Collector (BPC). Deletion keeps an efficient space for local search methods by removing the individual which have the same city orders as other individuals among the population. The individual removed is then re-generated by a preset mutation operator. The BPC is used to collect the best parts from different individuals and used to groom the elite individual. All the parts collected contain the same cities with the same start and end cities. If one part is found shorter than that in the elite individual, the part selected in the elite individual is recombined with the shorter one. The BPC can remarkably improve the tour by this process.

In the traditional GA, the selection is an important operation. It reproduces the elite individuals as new offspring. With the increase of the elite individuals from the selection, the population loses its diversity, resulting in the search space of other genetic operators becoming narrow. The selected elite usually forms a local optimum at the expense of finding an optimal solution. That is why we rejected employing the selection operator in our algorithm.

The crossover and the mutation are implemented in our algorithm. The crossover operator is an Edge Exchange Crossover (EXX) [19]. The mutation operator is a Two Point Change method which randomly changes the order of 2 cities. The EXX and the mutation intercalate a new diversity into the population. The individuals, recombined by the crossover and the mutation, can contain new crossed edges and in their turn become the new candidates for the C2Opt and SS operations.

III. Multiple Heuristic Algorithms as Local Search

Our new algorithm of multiple heuristics is shown in Fig. 1. The terminative condition of the algorithm is set to the total number of generations required. The order of the operations in the algorithm is significant and related to the results. The C2Opt is chosen, empirically, to be set after the SS. A tour which consists of *n* cities is expressed as $c=\{c_0,\cdots,c_i,\cdots,c_j,\cdots,c_{n-1}\}$. Each distance $d(c_i,c_j)$ is given for each pair of cities c_i and c_j. All cities are coded using a path representation method.

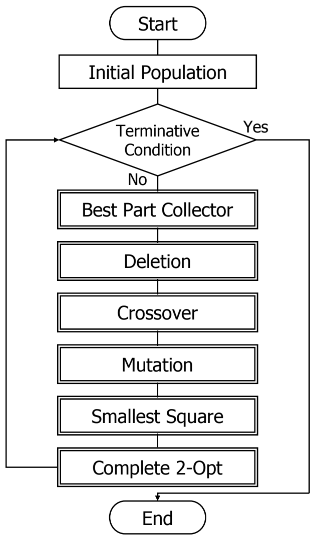


Fig. 1. Proposed algorithm of Evolutionary Multiple Heuristics with GLS.

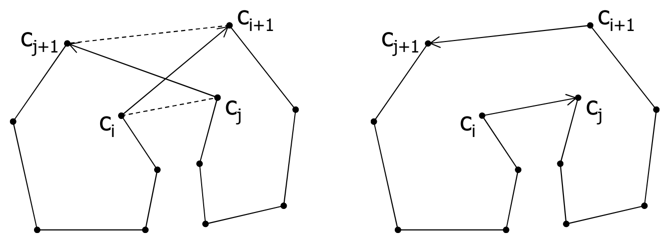


Fig. 2. Effect of C2Opt (Left figure: Before application of C2Opt, Right figure: After application of C2Opt).

A. Complete 2-Opt (C2Opt)

N_p tours are randomly initialized as one population. Here follow the steps to C2Opt:

(i) Choose one tour $c = \{c_0, \cdots, c_i, c_{i+1}, \cdots, c_j, c_{j+1}, \cdots, c_{n-1}\}$ (the left figure of Fig. 2). The i and the j are initialized to zero.

(ii) Choose the edge No. 1 (c_i, c_{i+1}) for $i < n$.

(iii) Choose the edge No. 2 (c_j, c_{j+1}) for $j < n$.

(iv) If $|j - (i+1)| \geq 2$, and $d(c_i, c_j) + d(c_{i+1}, c_{j+1}) < d(c_i, c_{i+1}) + d(c_j, c_{j+1})$, the 2-Opt is applied, namely, the edges (c_i, c_{i+1}) and (c_j, c_{j+1}) are removed, and the edges (c_i, c_j) and (c_{i+1}, c_{j+1}) are connected. The segment between c_{i+1} and c_j is reversed.

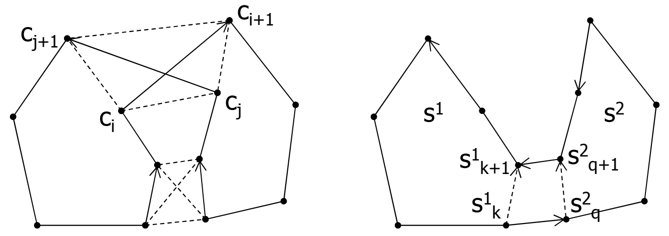


Fig. 3. Effect of SS (Left figure: Before application of SS, Right figure: After application of SS).

(v) The starting city c_j of the edge No. 2 is changed to the next city, namely, $j + 1$, and step (iii) and (iv) are repeated until $j = n - 1$. If $j = n - 1$, then $j + 1 = 0$.

(vi) The starting city c_i of the edge No. 1 is changed to the next city, namely, $i + 1$, and step (ii), (iii), (iv) and (v) are repeated until $i = n - 1$. If $i = n - 1$, then $i + 1 = 0$.

(vii) The steps of (ii), (iii), (iv), (v) and (vi) are repeated for N times. If N is sufficient, a tour without crossed edges will be created as shown in the right figure of Fig. 2.

B. Smallest square (SS)

In the TSP, a set of every two edges forms a square except for the neighboring edges. There are only three ways for the traveling salesman to pass through the 4 cities at the four corners of every square (the left figure of Fig. 3): $(c_i, c_{i+1}) + (c_j, c_{j+1})$, $(c_i, c_j) + (c_{i+1}, c_{j+1})$ and $(c_i, c_{j+1}) + (c_{i+1}, c_j)$. Obviously the path of $(c_i, c_{j+1}) + (c_{i+1}, c_j)$ is the shortest route in the square. However, if the salesman takes $(c_i, c_{j+1}) + (c_{i+1}, c_j)$, the tour will be separated into two sub tours. Another two edges have to be found for reconnecting the tour in the second square.

N_p tours are randomly initialized as one population. Here follow the steps to SS:

(i) An initial tour $c = \{c_0, \cdots, c_i, c_{i+1}, \cdots, c_j, c_{j+1}, \cdots, c_{n-1}\}$ (the left figure of Fig. 3) is chosen in the population. The i and the j are initialized to zero. The first square is searched.

(ii) Choose the edge No. 1 (c_i, c_{i+1}) for $i < n$.

(iii) Choose the edge No. 2 (c_j, c_{j+1}) for $j < n$.

(iv) If $|j-(i+1)| \geq 2$, and $d(c_i,c_{j+1})+d(c_{i+1},c_j)<d(c_i,c_j)+d(c_{i+1},c_{j+1})<d(c_i,c_{i+1})+d(c_j,c_{j+1})$, the edges (c_i,c_{i+1}) and (c_j,c_{j+1}) are temporarily removed, and the edges (c_i,c_{j+1}) and (c_{i+1},c_j) are connected. The tour is separated into two sub tours s^1 and s^2 (the right figure of Fig. 3). The s^1 and s^2 consist of t and u cities, respectively where $t + u = n$. The city orders of s^1 and s^2 are $c_{s^1} = \{s_0^1,\cdots s_k^1,s_{k+1}^1,\cdots,s_{t-1}^1\}$ and $c_{s^2} = \{s_0^2,\cdots s_q^2,s_{q+1}^2,\cdots,s_{u-1}^2\}$, respectively.

(v) The second square is searched. The k and the q are initialized to zero.

(a) Choose the edge No. 3 (s_k^1,s_{k+1}^1) from sub tour s^1 where $k<t$.

(b) Choose the edge No. 4 (s_q^2,s_{q+1}^2) from sub tour s^2 where $q<u$.

(c) The three pairs of the edges in the second square are checked to find the shortest pair for reconnecting the tour.

(1) If $d(s_k^1,s_q^2)+d(s_{q+1}^2,s_{k+1}^1)<d(s_k^1,s_{k+1}^1)+d(s_q^2,s_{q+1}^2)<d(s_k^1,s_{q+1}^2)+d(s_q^2,s_{k+1}^1)$, the edges (s_k^1,s_{k+1}^1) and (s_q^2,s_{q+1}^2) are removed and the edges (s_k^1,s_q^2) and (s_{q+1}^2,s_{k+1}^1) are connected. A better tour is formed. Then, go to the step (vi).

(2) If $d(s_k^1,s_{q+1}^2)+d(s_q^2,s_{k+1}^1)<d(s_k^1,s_{k+1}^1)+d(s_q^2,s_{q+1}^2)<d(s_k^1,s_q^2)+d(s_{q+1}^2,s_{k+1}^1)$, the edges (s_k^1,s_{k+1}^1) and (s_q^2,s_{q+1}^2) are removed and the edges (s_k^1,s_{q+1}^2) and (s_q^2,s_{k+1}^1) are connected. A better tour is formed. Then, go to the step (vi).

(3) If $d(s_k^1,s_{k+1}^1)+d(s_q^2,s_{q+1}^2)<d(s_k^1,s_q^2)+d(s_{q+1}^2,s_{k+1}^1)<d(s_k^1,s_{q+1}^2)+d(s_q^2,s_{k+1}^1)$, the start city of edge No. 4 is changed to the next one, namely, $q + 1$. If $q = u - 1$, then $q + 1 = 0$, and go to step (b). When $q = u - 1$, the starting city of the edge No. 3 is changed to the next one, namely $k + 1$, and go to step (a). If no pair of edges in the second square is found to make the tour shorter between c_{s^1} and c_{s^2} until $k = t - 1$ and $q = u - 1$, the 2-Opt is applied to the first square (Refer to the step (iv)), namely, the edges (c_i,c_{i+1}) and (c_j,c_{j+1}) are removed and the edges (c_i,c_j) and (c_{i+1},c_{j+1}) are connected. The segment between c_{i+1} and c_j is reversed.

(vi) The starting city c_j of the edge No. 2 is changed to the next city, step (iii), (iv) and (v) are repeated until $c_j = c_{n-1}$. If $j = n - 1$, then $j + 1 = 0$.

(vii) The starting city c_i of the edge No. 1 is changed to the next city, step (ii), (iii), (iv), (v) and (vi) are repeated until $c_i = c_{n-1}$. If $i = n - 1$, then $i + 1 = 0$.

(viii) The steps of (ii), (iii), (iv), (v), (vi) and (vii) are repeated N times. If N is sufficient, a tour close to the optimal solution will be obtained.

In fact, for some TSP instances, the optimal solution can be directly obtained from C2Opt or SS+C2Opt, for example, in our previous work [17], the optimal solution of C type of Concentric Circles TSP was obtained easily by a single CPU even though the number of cities reaches to 20,000. The repeat times (N) were experientially set to different numbers in our experiments. It is difficult to decide how many repeat times are sufficient for running the C2Opt or SS+C2Opt to remove all the crossed edges in the tour, because the amount of crossed edges depend on the TSP instance itself. But we found this problem could be solved while C2Opt or SS+C2Opt is applied to the same individual randomly for $2 \times N$ or more times.

C. *Deletion and best part collector (BPC)*

Deletion is effective for removing the individual which has the same city orders as the other individuals among the population. Deletion is applied to the six elite individuals. The individual removed is re-generated by the mutation operator using another individual selected randomly from the population. Removing the duplicates from the population helps to keep sufficient search spaces for the operation BPC as shown in Fig. 4.

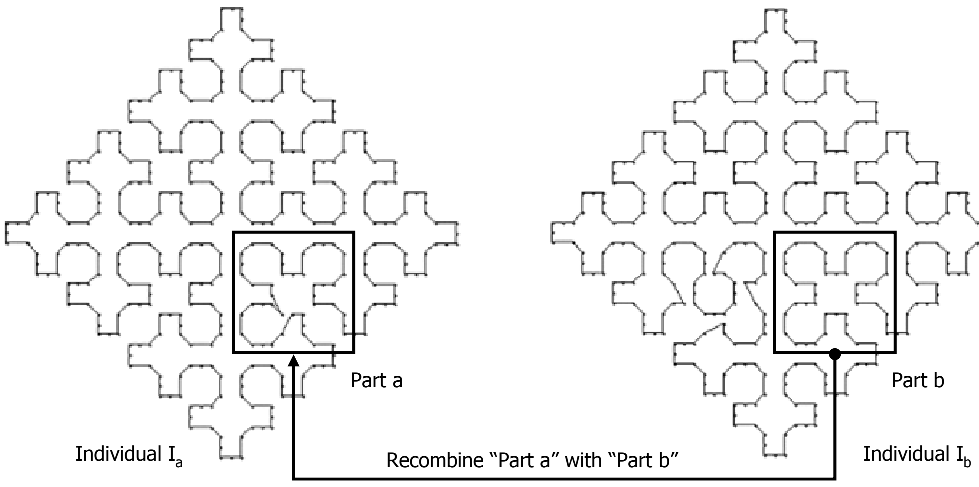


Fig. 4. Effect of BPC.

In our algorithm, the C2Opt and the SS+C2Opt directly create tours in which crossed edges are completely removed and partially contain the same city orders as they are in the optimal tour. The BPC is used to collect the partially optimal parts for the elite individual from the other individuals. Firstly, we randomly and continuously select *P* cities in the elite individual, $4 < P < n/2$ (Fig. 4, Part a). The minimum range of *P* is set to 4 cities because it is the minimal improvable part which contains the same starting city and ending city. $n/2$ is set as the maximum range because it is large enough to cover all improvable parts in the tour which crossed edges are removed. Secondly, the individuals are examined for parts which contain the same cities, with the same starting and ending cities as the part that is selected from the elite individual. When one part is found to be shorter than that in the elite individual, the part selected in the elite individual is recombined with the shorter one (Fig. 4, Part b). The BPC is applied to the elite individuals from No. 0 to No. 5.
The EXX and mutation operators are randomly applied to each percentage of the population; 60% and 5%, respectively, except for the 6 chosen elite individuals who remain unchanged to preserve their integrity.

IV. Experiments and Considerations

A. Experimental conditions

The tour which contains *n* cities $c = \{c_0, \cdots, c_i, \cdots, c_j, \cdots, c_{n-1}\}$ is given and the distance between two cities c_i and c_j is $d(c_i, c_j)$. Total distance d_{total} is given by

$$d_{total} = \sum_{i=0}^{n-1} d(c_i, c_{i+1}), \tag{1}$$

where $c_n = c_0$. d_{total} is the same as the fitness of the individual. The shorter the distance is, the higher the fitness is. The population size is set to $N_p = 100$.

Our source code is written in Java and runs on a PC (CPU: Pentium III 1.0GHz, RAM: 256MB) with Microsoft Windows 2000 Operating System.

The experiments are designed with the following seven cases and each case is applied to the TSP instances independently and numbers in the parentheses are the repeat times in each generation;

Table 1. Seven cases in the experiments.

ALGO No.	Different local searches			Fixed operations		
No. 1	BPC (10)			Deletion	EXX	Mutation
No. 2			C2Opt (10)	Deletion	EXX	Mutation
No. 3		SS (10)		Deletion	EXX	Mutation
No. 4		SS (5)	C2Opt (5)	Deletion	EXX	Mutation
No. 5	BPC (10)		C2Opt (10)	Deletion	EXX	Mutation
No. 6	BPC (10)	SS (10)		Deletion	EXX	Mutation
No. 7	BPC (10)	SS (5)	C2Opt (5)	Deletion	EXX	Mutation

The BPC is applied to the 5 shortest tours in every case. The parameters of EXX, Deletion and mutation are fixed in all cases.

Two instances of the Euclidean Traveling Salesman Problem (ETSP) and three TSP instances from TSPLIB are used in our experiments. One of the ETSP instances is the David tour as shown in Fig. 5 [20]. The instance from TSPLIB are att48 (Fig. 6), kroC100 (Fig. 7) and ch130 (Fig. 8). Another one of the ETSP is the MNPeano tour as shown in Fig. 9 [21]. The coordinates of ETSP instances are generated with L-Systems [22].

The David tour and the instance att48 are used for examining the efficiency of the algorithm. The case of BPC, SS and C2Opt is applied to the MNPeano tour and two other TSP instances (kroC100 and ch130) to confirm the robustness of the algorithm. Every optimal solution is confirmed with the solution provided in TSPLIB.

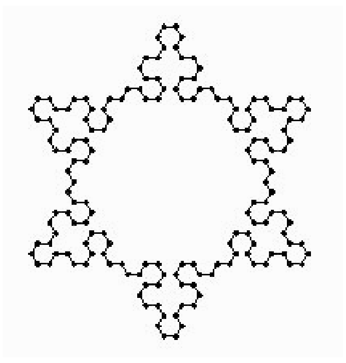


Fig. 5. The David tour.

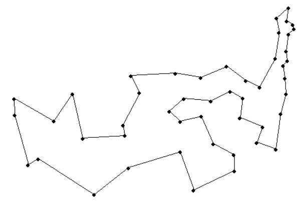


Fig. 6. The shortest tour of att48.

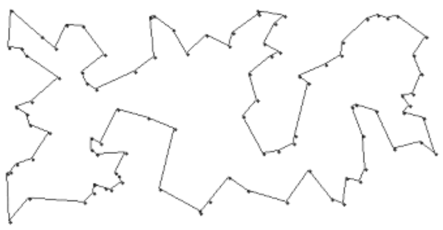


Fig. 7. The shortest tour of kroC100.

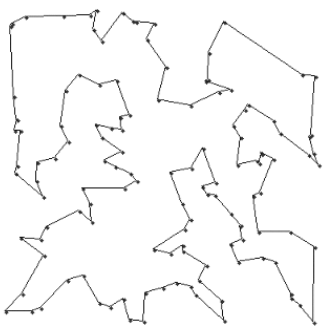


Fig. 8. The shortest tour of ch130.

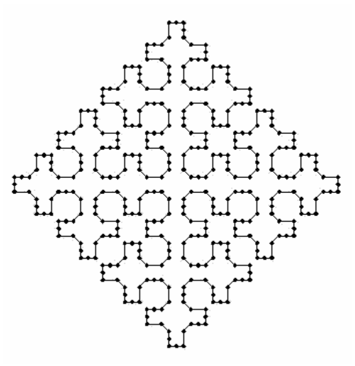


Fig. 9. The MNPeano tour.

B. *Experiments of David tour and att48*

Table 2 and 3 show the results of different cases which are applied to the David tour and the att48, respectively. The BPC is used to collect the optimal parts from other individuals and apply them to cultivate the elite individual. It is repeated 10 times and applied to the 5 shortest tours in each generation in the algorithm. The result indicates that it is impossible to obtain the optimal solutions when only the BPC is set with the EXX, Deletion and mutation. Six optimal solutions are obtained in the att48 and none is found in the David tour in the case of BPC and C2Opt. The results of combination of BPC and SS are not also satisfied in both David tour and att48. The case of SS and C2Opt presents a significant result: eleven optimal solutions in the David tour and sixteen in the att48 are obtained. As mentioned in section 3, the C2Opt could completely remove the crossed edges, it is beneficial to create the candidates for the BPC. If compared with the C2Opt, the SS took a much longer time to run the same number of generations. This happens because the SS checks three pairs of the edges in the square and one more square is checked in the same generation, the tour is reconnected with the shortest pair of edges. The problem is that the city orders of the tour are changed while the tour is reconnected in the second square. Some crossed edges in the tour could not be completely removed. The case of SS and C2Opt presents a better result than single SS or C2Opt because C2Opt could remove the crossed edges which remain in the tour after the application of SS. The best combination is BPC, SS and C2Opt which form the new algorithm with the EXX, Deletion and mutation. Optimal solutions are obtained for all tours in the 20 runs. The time and generations needed for producing the optimal solutions become shorter and earlier than the other cases in both instances of the David tour and the att48.

Table 2. Experimental results of David tour with different local searches.

TSP	Cities	ALGO	Gen.	MiniG	MaxG	MG	ST	LT	Mt	TT	TS
David tour	162	BPC	20	—	—	—	—	—	—	3	0
David tour	162	C2Opt	20	—	—	—	—	—	—	16	0
David tour	162	SS	20	—	—	—	—	—	—	78	0
David tour	162	SS, C2Opt	20	2	18	11	6	47	28	50	11
David tour	162	BPC, C2Opt	20	—	—	—	—	—	—	15	0
David tour	162	BPC, SS	20	10	17	15	44	67	57	75	4
David tour	162	BPC, SS, C2Opt	20	1	17	6	5	50	18	57	20

Gen.: Total generation. MiniG: Minimum generation of the optimal solution. MaxG: Maximum generation of the optimal solution. MG: Mean generation of the optimal solution. ST: Shortest time for obtaining the optimal solution. LT: Longest time for obtaining the optimal solution. MT: Mean time for obtaining the optimal solution. TT: Mean time for running the total generation. The unit of time is second. TS: Total optimal solutions obtained in the 20 runs. The "—" means no data was obtained.

Table 3. Experimental results of att48 with different local searches.

TSP	Cities	ALGO	Gen.	MiniG	MaxG	MG	ST	LT	Mt	TT	TS
att48	48	BPC	100	—	—	—	—	—	—	2	0
att48	48	C2Opt	100	32	87	52	2	6	3	7	7
att48	48	SS	100	—	—	—	—	—	—	19	0
att48	48	SS, C2Opt	100	4	66	29	1	9	3	13	16
att48	48	BPC, C2Opt	100	16	96	55	1	6	3	7	6
att48	48	BPC, SS	100	77	77	77	17	17	17	21	1
att48	48	BPC, SS, C2Opt	100	7	85	32	1	12	4	14	20

Gen.: Total generation. MiniG: Minimum generation of the optimal solution. MaxG: Maximum generation of the optimal solution. MG: Mean generation of the optimal solution. ST: Shortest time for obtaining the optimal solution. LT: Longest time for obtaining the optimal solution. MT: Mean time for obtaining the optimal solution. TT: Mean time for running the total generation. The unit of time is second. TS: Total optimal solutions obtained in the 20 runs. The "—" means no data was obtained.

C. Experiments of kroC100, ch130 and MNPeano

Furthermore, the new algorithm is applied to the MNPeano tour which consists of 364 cities and the other two TSP instances of kroC100 and ch130. It is noted that all the solutions are optimal in the 20 runs (Table 4). For kroC100 tour the earliest generation for obtaining the optimal solution is at the 19th and the latest one, is found, at the 218th. The average value is the 87th generation. The shortest time for obtaining the optimal solution is only 13 seconds. The longest time is 142 seconds. The average value is 60 seconds. The average time for running 20 generations takes only 184 seconds. The optimal tour of the MNPeano tour is shown in Fig. 9. The earliest optimal solution is obtained at the 123rd generation and takes 1,494 seconds of single CPU time. The similar result appears in the ch130. The results of the experiments indicate that the proposed evolutionary multiple heuristics combining the GLS is an effective algorithm for the TSP.

Table 4. Experimental results of different TSP instances with all local searches.

TSP	Cities	ALGO	Gen.	MiniG	MaxG	MG	ST	LT	Mt	TT	TS
kroC100	100	BPC, SS, C2Opt	300	19	218	87	13	142	60	184	20
ch130	130	BPC, SS, C2Opt	1,000	272	857	571	260	1,031	601	907	20
MNPeano	364	BPC, SS, C2Opt	3,000	123	2,626	1,133	1,494	23,589	10,539	27,553	20

Gen.: Total generation. MiniG: Minimum generation of the optimal solution. MaxG: Maximum generation of the optimal solution. MG: Mean generation of the optimal solution. ST: Shortest time for obtaining the optimal solution. LT: Longest time for obtaining the optimal solution. MT: Mean time for obtaining the optimal solution. TT: Mean time for running the total generation. The unit of time is second. TS: Total optimal solutions obtained in the 20 runs.

V. Summary

The operations of the SS and the C2Opt are looked at in depth, revealing the pros and cons of each. The SS finds sorter edges but leaves some edges crossed. The C2Opt succeeds in removing all the crossed edges, although solutions are not as short. By combing first the SS and then the C2Opt after the BPC+Deletion and the genetic EXX+Mutation operations, an optimal tour can be found in a reasonable number of generations, which is our principal result.

In the future we plan to look at how to scale up the algorithm for solving large instances of TSP.

Acknowledgement

The authors would like to thank Mr. Christopher J. Ashley for checking the idiomatic usage very carefully in the paper.

References

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems* (Univ. of Michigan Press, 1975).

[2] A. Homaifar, S. Guan, and G. E. Liepins, A New Approach to the Traveling Salesman Problem by Genetic Algorithms, in *Proc. 5th International Conference on Genetic Algorithms* (Morgan Kaufmann, 1993) 460–466.

[3] R. M. Brady, Optimization Strategies Gleaned from Biological Evolution, Nature, **317** (1985) 804–806.

[4] T. G. Bui and B. R. Moon, A New Genetic Approach for the Traveling Salesman Problem, in *Proc. 1st IEEE Conference on Evolutionary Computation* (1994) 7–12.

[5] G. Peng, I. Iimura, H. Tsurusawa, and S. Nakayama, Genetic Local Search Based on Genetic Recombination: A Case for Traveling Salesman Problem, in *Proc. 5th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT 2004)* (Singapore, 2004) 202–212.

[6] G. Peng, T. Nakatsuru, and S. Nakayama, Discussions on LGA with Parallel System, in *Proc. Genetic and Evolutionary Computation Conference (GECCO 2005)* (Washington, DC, 2005).

[7] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, The Traveling Salesman Problem: *A Guided Tour of Combinatorial Optimization* (John Wiley & Sons, 1985).

[8] S. Lin and B. Kernighan. An Efficient Heuristic Procedure for the Traveling Salesman Problem, *Operations Research*, **21**(2) (1973) 498–516.

[9] Michael Hahsler and Kurt Hornik. TSP - Infrastructure for the traveling salesperson problem. Report 45, Research Report Series, Department of Statistics and Mathematics, Wirtschaftsuniversität Wien, Augasse 2-6, 1090 Wien, Austria.(2006).

[10] GUTIN TELOS. The Traveling Salesman Problem and its Variations. ISBN: 978-0-387-44459-8 (2007) 830 p.

[11] G.Reinelt. The Traveling Salesman: Computational Solutions for TSP Applications. Lecture Notes in Computer Science 840, Springer, (1994)

[12] P. C. Kanellakis and C. H. Papadimitriou, Local search for asymmetric traveling salesman problem, *Operations Research*, **28**(5) (1980) 1066–1099.

[13] Bentley, J. L. Experiments on traveling salesman heuristics. In *Proc. 1st Symp. Discrete Algorithms*, (1990) pp. 91-99. ACM and SIAM.

[14] Rosenkrantz, D. H., Stearns, R. E., and Lewis, P. M., II. An analysis of several heuristics for the traveling salesman problem. *SIAM J. Computing 6*, 3 (1977), 563-581.

[15] K. Mathias and Darrell Whitley, Genetic operators, landscape and the traveling salesman problem, *Parallel problem Solving from Nature 2* (1992) 219–228.

[16] G. Peng and S. Nakayama, Multiple Heuristic Search in Genetic Algorithm for Traveling Salesman Problem, in *Proc. Genetic and Evolutionary Computation Conference (GECCO 2003)*, Late-Breaking Papers (Chicago, Illinois, USA, 2003) 88–93.

[17] G. Peng, I. Iimura, and S. Nakayama, A Multiple Heuristic Search Algorithm for Solving Traveling Salesman Problem, in *Proc. 4th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT 2003)* (Chengdu, China, 2003) 779–783.

[18] G. Peng, I. Iimura, and S. Nakayama, Consideration on Multiple Heuristic Search Method in Traveling Salesman Problems, *Trans. JSCES*, **6** (2004) 155–161. (in Japanese)

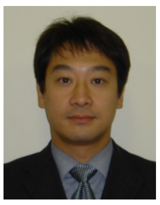
[19] K. Maekawa, H. Tamaki, H. Kita, and Y. Nishikawa, A method for the traveling salesman problem based on the genetic algorithm, *Trans. the Society of Instrument and Control Engineers*, **31**(5) (1995) 598–605. (in Japanese)

[20] P. Moscato and M. G. Norman, An analysis of the performance of traveling salesman heuristics on infinite-size fractal instances in the Euclidean plane, *Technical report* (CeTAD-Universidad Nacional de La Plata, 1994).

[21] M. G. Norman and P. Moscato, The Euclidean Traveling Salesman Problem and a Space-Filling Curve, *Chaos, Solitons and Fractals*, **6** (1995) 389–397.

[22] A. Mariano, P. Moscato, and M. G. Norman, Using L-Systems to generate arbitrarily large instances of the Euclidean Traveling Salesman Problem with known optimal tours, *Anales del XXVII Simposio Brasileiro de Pesquisa Operacional*, (Vitoria, Brazil, Nov. 1995) 6–8.

Peng Gang received the Ph.D. degree in Agriculture and Ph.D. degree in Information Science at Kagoshima University, Japan in 2001 and 2004 respectively. His research focuses on the Evolutionary Computation. He is a member of International Society for Genetic and Evolutionary Computation (ISGEC), The Institute of Electronics , Information and Communication Engineers (IEICE) and Computer Aided Diagnosis of Medical Images (CADM).



Ichiro Iimura was born in Ibaraki, Japan in 1969. He received the B. Eng. and the M. Eng. from Sophia University in 1992 and 1994, respectively and the Dr. Eng. from Kagoshima University in 2004. From 1994 to 1997, he was Research Scientist for Hitachi Research Laboratory, Hitachi, Ltd. From 1997 to 2002, he was Lecturer for Kumamoto Prefectural College of Technology. From 2002 to 2003, he was Research Associate, Prefectural University of Kumamoto. From 2003 to 2006, he was Senior Lecturer, and from 2006, he has been Associate Professor. He received “Best Paper Award for Young Researcher” from IPSJ in 2001, “Certificate of Merit for Best Presentation” from JSME in 2003, and “Best Paper Award for Young Researcher” from IPSJ Kyushu chapter in 2003, respectively. His research interests include evolutionary computation, swarm intelligence, and distributed parallel processing. He is a member of IPSJ, IEICE, and IEEJ, etc.

Shigeru Nakayama was born in Kyoto, Japan in 1948. He received the B. Sc. E. E. from Kyoto Institute of Technology in 1972, M. Sc. E. E. and Doctor of Engineering from Kyoto University in 1974 and 1977,respectively. From 1977 to 1981, he was Research Associate for Sophia University in Tokyo. From 1981 to 1987 he was Research Associate for Kyoto Institute of Technology. From 1987 to 1997, he was Associate Professor for Hyogo University of Teacher Education. From 1997, he became Professor of Information and Computer Science for Kagoshima University. His research interests include distributed parallel processing, parallel genetic algorithm, parallel image processing and distributed objects. He is a member of Inst. of Electronics, Information and Communication Engineers, Information Processing Society and J.A.P.S.