Electronics and Communications in Japan, Part 3, Vol. 84, No. 2, 2001
Translated from Denshi Joho Tsushin Gakkai Ronbunshi, Vol. J81-A, No. 12, December 1998, pp. 1783–1791

# An Efficient Hybrid Genetic Algorithm for the Traveling Salesman Problem

Kengo Katayama and Hiroyuki Narihisa

Department of Information and Computer Engineering, Okayama University of Science, Okayama, Japan 700-0005

## SUMMARY

This paper describes an efficient hybrid genetic algorithm (HGA) for the traveling salesman problem. In general, a genetic algorithm (GA) combined with other algorithms (e.g., a local search) is well known to be a powerful approach. The other algorithms are divided into local search heuristics and metaheuristics. In incorporating the metaheuristics, it is reported that a difficult problem of changing processes between the GA process and the metaheuristic search process appears. To avoid the difficult problem using simulated annealing as one of the metaheuristics, we investigate an efficient HGA that does not involve the problem. © 2000 Scripta Technica, Electron Comm Jpn Pt 3, 84(2): 76–83, 2001

**Key words:** Hybrid genetic algorithm; traveling salesman problem; complete subtour exchange crossover.

## 1. Introduction

Many combinatorial optimization problems are $\mathcal{NP}$-hard. It is believed that these problems cannot be solved to optimality within polynomially bounded computation times. However, heuristic algorithms can obtain near-optimal solutions within reasonable times. As one such algorithm, the genetic algorithm (GA) [1–3], based on the evolutionary principle of natural chromosomes, is well known. The GA has been successfully used in a wide variety of problem domains because of its simplicity, ease of operation, minimal requirements, and global perspective. Generally, the GA consists of operators such as selection, crossover, and mutation. A local search or metaheuristic algorithm is incorporated into these genetic operators to make a more powerful algorithm. Such an algorithm is called a "hybrid GA" (HGA) [9–14, 26, 24]. In general, the HGA is based on applying the genetic operators to a set of local optima found by local search algorithms.

Kido [9] combined simulated annealing (SA) and tabu search (TS), which are metaheuristics and contain many parameters, with the GA search process independently, and reported that the HGA obtained better results than the individual algorithms in the traveling salesman problem. Moreover, he reported that a "timing" problem occurred in the HGA. This problem occurs when switching a search process from the metaheuristic to the GA. In this paper we call the problem the "switching-time problem." To avoid this, he adopted a fixed search time that depended on setting the values of the parameters in the metaheuristic by user's experience. That is, to independently search by the metaheuristic in each generation, the search process is changed to the GA process when the process of the metaheuristic is terminated by the parameter values set by a user. This problem is considered a difficult problem in using the HGA.

In this paper, we employ the complete subtour exchange crossover [26, 27], and present an efficient HGA that does not involve the "switching-time problem." The metaheuristic incorporated is the simulated annealing algorithm, which incurs the problem in a general case of combining with the GA process. In our HGA with the SA, we combined only a part of the SA process, which depends on

© 2000 Scripta Technica

the search time for a given instance size, with the GA process, particularly the mutation process among all of the genetic processes; in the SA process a temperature parameter of the SA is fixed for each generation, but the temperature is decreased after each individual is subjected to the SA. The HGA(SA) is tested on a benchmark set [22] of the traveling salesman problem, and we investigate the efficiency of our HGA(SA).

## 2. Combinatorial Optimization Problem and Genetic Algorithm

The GA has been typically applied to combinatorial optimization problems. Here, we describe combinatorial optimization problems and the GA.

A combinatorial optimization problem is specified by a set of problem instances and is either a minimization problem or a maximization problem. Here, we consider the minimization problem.

The objective of a combinatorial optimization problem is to find a minimum cost element in the set of feasible solutions:

$$\min \{ f(x) : x \in F, F \subseteq X \}$$

where $F$ denotes the set of feasible solutions and $f$ is a mapping $f : F \to Z$. The problem is to find a globally optimal solution, that is, the solution $x \in F$ such that $f(x) \leq f(y), \forall y \in F$.

In many cases, in order to obtain the optimal solution of a problem instance, it is possible to apply exact methods to the combinatorial optimization problem. However, due to the computational complexity of the problem, heuristic algorithms may be useful to find near-optimal solutions for large problems within reasonable running times, including simulated annealing and the genetic algorithm.

The GA is a heuristic algorithm based on the evolutionary principle, and maintains a population of candidate solutions (individuals) and a fitness function to evaluate the individuals. The general flow of the GA is shown in Fig. 1. In each generation $k$, the size of the population $P(k) = \{I_1^k, \ldots, I_{PS}^k\}$, where $PS$ denotes the number of individuals, is fixed. To obtain better individuals, genetic operators such as selection, crossover, and mutation are applied in each generation. To measure and select how well the individual fits in the current generation, the selection operator is employed with the fitness function. The crossover operator consists of choosing a pair of individuals among those selected in the previous generation, and offspring are then generated using a mechanism that inherits valid characters of two parents. The mutation operator is usually considered a secondary search operator, but the operator may lead to other regions of the search space of the problem.

```
procedure   genetic algorithms
 01: begin
 02:   k := 0
 03:   P(k) := a set of initial feasible solutions
 04:   evaluate structures in P(k)
 05:   while stopping-criterion ≠ yes do
 06:     begin
 07:       k := k + 1
 08:       P(k) := select from P(k − 1)
 09:       alter structures in P(k) by crossover
 10:       alter structures in P(k) by mutation
 11:       evaluate structures in P(k)
 12:     end
 13:   return the best solution
 14: end.
```

Fig. 1. General flow of genetic algorithms.

## 3. Traveling Salesman Problem

### 3.1. TSP and coding

Given a graph $G = (V, E)$ which consists of a set of $n$ nodes (cities) and a distance function $d : E \to Z^+$, the traveling salesman problem (TSP) is to find a shortest tour (Hamilton cycle) visiting each of the set of cities exactly once, starting from and returning to a start city [4, 20]. In particular, we use the abbreviation TSP only for the *symmetric* traveling salesman problem, in which the distance satisfies $d_{ij} = d_{ji}$ for $1 \leq i, j \leq n$. The number of Hamiltonian cycles, that is, the size of feasible solutions in the TSP, is actually $(n - 1)!/2$.

In the case of applying the GA to the TSP, the genes are city names (i.e., path representation). The finite length

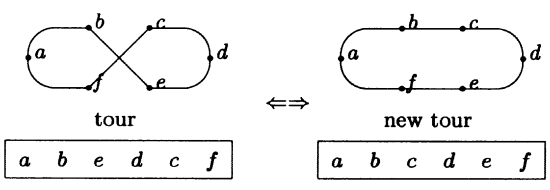


Fig. 2. An example of 2-Opt neighborhood.

of the chromosome corresponds to the size of a given TSP. The fitness is computed by the sum of all of the distances between cities on the Hamiltonian cycle.

### 3.2. Neighborhood

Given a set of feasible solutions $F$, the neighborhood is the mapping $N : F \rightarrow 2^F$. A local optimal solution $x \in F$ is satisfied $f(x) \leq f(y)$, $\forall y \in N(x)$. In many cases, the strategy for combinatorial optimization problems is based on a local search heuristic. The local search starts from an arbitrary solution $x \in F$ and the neighborhood solution $x' \in N(x)$ from the current solution is created. If $f(x') < f(x)$, $x'$ is set to $x$. The local search is performed until the local optimal solution is found.

Typical neighborhood structures such as 2-Opt, 3-Opt, Or-Opt, Lin-Kernighan Opt have been proposed for the TSP [5, 8]. In this paper, we employ the 2-Opt neighborhood method for the local search in our experiments. Figure 2 displays an example of the 2-Opt neighborhood. Given a set of all edges $T(\rho)$ for a permutation $\rho$ and a set of Hamilton cycle $H$ on $G = (V, E)$, 2-Opt neighborhood $N_{2opt}$ is defined as follows: $N_{2opt}(\rho) = \{\rho' \in H : T(\rho') = T(\rho) \setminus \{e_1, e_2\} \cup \{e_3, e_4\}, e_1, e_2 \in T(\rho), e_3, e_4 \in E \setminus T(\rho)\}$ [20].

## 4. Complete Subtour Exchange Crossover

Here, we describe a complete subtour exchange crossover (CSEX) [26, 27] since the CSEX is incorporated into our HGA.

The CSEX inherits as many good subtours as possible because they are worth preserving for descendants. The CSEX enumerates the subtours that have the same direction (or opposite direction) on two permutations as common subtours, and can create feasible offspring solutions. In this enumeration, the CSEX handles simplified common subtour used in the SXX [6] or Brady's [5].

Figure 3 shows an example of the CSEX. The common subtours are $\boxed{b, c}$ and $\boxed{f, g, h}$ in ParentA, and $\boxed{h, g, f}$ and $\boxed{c, b}$ in ParentB. In the case of the SXX, $\boxed{b, c, d}$ of ParentA and $\boxed{c, b, d}$ of ParentB are enumerated as a common subtour. However, in the case of the CSEX, these are not enumerated as a common subtour. That is, $\boxed{b, c}$ of ParentA and $\boxed{c, b}$ of ParentB excluding $d$ are the common subtour. The common subtour handled in the CSEX is for the direction of the subset (subtour), which is completely the same or symmetrical. Consequently, it is possible to enumerate all of the common subtours with $O(n)$ time. In Ref. 6, the SXX needed $O(n^5)$ time, but Yagiura and colleagues showed that it was possible to enumerate the subtour with $O(n^2)$ [7] and $O(n + K)$ [8], where $K$ denotes the number of the subtours in the SXX.

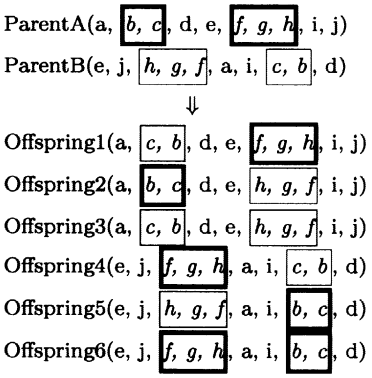


Fig. 3. An example of complete subtour exchange crossover.

In Fig. 3, offspring can be generated, for example, six offspring. In this context, if $K$ common subtours are included among the parents, a maximum of $2 \times 2^K - 2$ offspring are generated. However, in this paper the best cost offspring among all of the offspring survive to the next search process (see Ref. 27 for more details).

## 5. Hybrid Genetic Algorithm

In this section, we describe the difficult problem in the standard HGA and an implementation of our HGA(SA).

### 5.1. Standard HGA and its problem points

In general, the standard HGA with the metaheuristic algorithm, both the GA process and the process of the metaheuristic are divided independently. After the genetic operators such as selection, crossover, and mutation, the metaheuristic algorithm performs a search, and when the search of the metaheuristic is terminated with a "stopping rule," the GA process starts again. However, since the stopping rule to terminate the process of the metaheuristic algorithm depends on the parameters of the metaheuristic, the "switching-time problem" occurs between the two independent processes [9]. In Ref. 9, the switching time was set by the user's experience using a fixed parameter value. Figure 4 shows the difference between the standard HGA (a) and our HGA (b).

In this paper, we investigate an HGA incorporating a part of the metaheuristic (SA) process into the mutation process of the GA. The stopping rule of the SA process in the HGA(SA) depends on the size of a given input instance. By using this rule, the HGA(SA) does not incur the switching-time problem.

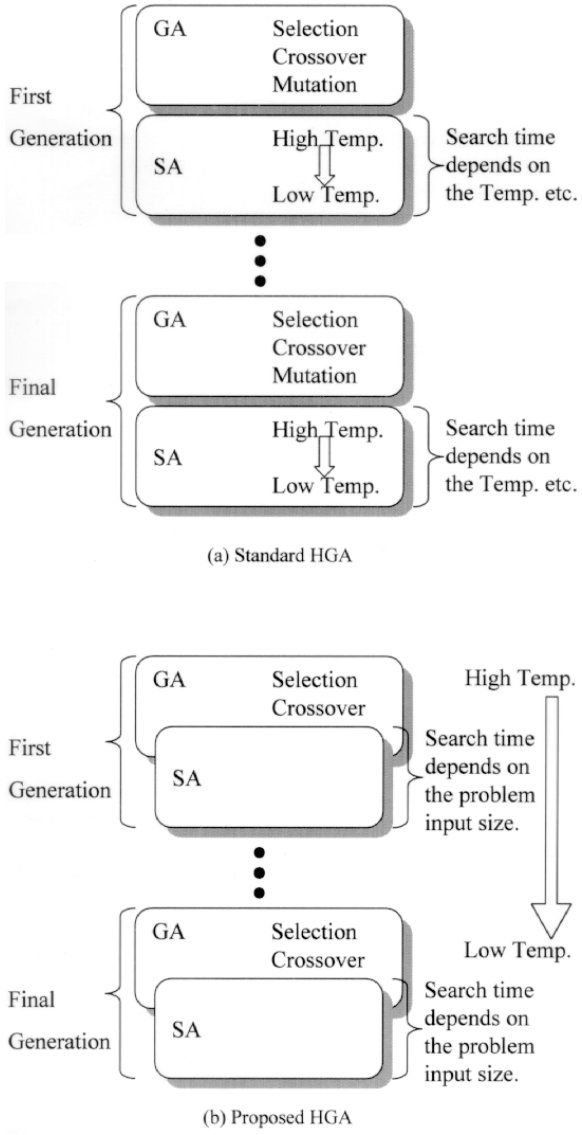


Fig. 4. Standard HGA and proposed HGA.

### 5.2. Hybrid GA with SA

Simulated annealing (SA) [16, 20] is based on an analogy with the physical process of annealing, in which a pure lattice structure of a solid is made by heating the solid in a heat bath until it melts, then cooling it down slowly until it solidifies into a low-energy state. In the combinatorial optimization, the SA is a randomized algorithm to search the solution neighborhood. Many variations of the SA have been proposed in the past; however, we investigate a basic version of SA as shown in Fig. 5. Step05 is a process that depends on an instance input size and that determines the search order for creating the neighborhoods of the current solution with $O(n^2)$ (in the case of the 2-Opt neighborhood). Generally, the search order is random in the SA. Therefore, Step07 is an operation that chooses the neighborhood solutions from the current solution. After that, if the better solution $x'$ is found, that is, $f(x') < f(x)$, solution $x'$ becomes $x$; otherwise, that is, $f(x') \geq f(x)$, the solution $x'$ becomes $x$ with a probability $\exp(-(f(x') - f(x))/T)$, where $T$ denotes a temperature in the SA algorithm.

As parameters of the SA, in general, an initial temperature $T$, a final temperature $FT$, and a cooling ratio $\alpha$ ($0 < \alpha < 1$) for the decrease in the current temperature $T$ are required. In our SA algorithm, we adopt a general method to decrease the current temperature in Step13 [9, 10].

By setting the parameters such as $\alpha$, $T$, and $FT$, we can obtain different solutions. Generally, the stopping rule of the SA process is a state of parameter values of $T <= FT$. If $\alpha$ and $T$ are set to relatively large values, we can obtain better solutions than the other setting cases.

In our HGA(SA) shown in Fig. 4(b), the temperature $T$ is decreased in each generation, and with the fixed temperature the process of the SA depends on the problem input size rather than the parameters as described above. The flow of the HGA(SA) is shown in Fig. 6. The processes from Step05 to Step12 in Fig. 5 depend on the problem input size, and these processes are incorporated into the mutation process of the GA. Step13 and Step14 in Fig. 5 are a control operation for the temperature $T$, and are incorporated into Step23 and Step24 in Fig. 6. Therefore, the temperature $T$

```
procedure    simulated annealing algorithm
01: begin
02:   set α(0 < α < 1), T to suitably high value,
      and FT as termination temperature value;
03:   x := a feasible solution
04:   repeat
05:     while does not reach the number of times do
06:       begin
07:         choose an x' ∈ N(x) randomly
08:         if  f(x') < f(x) then
09:           x := x'
10:         else
11:           x := x' with prob. exp(−(f(x') − f(x))/T)
12:       end
13:     T := T * α
14:   until T <= FT;
15:   return the solution x
16: end.
```

Fig. 5. Simulated annealing algorithm.

```
procedure   hybrid GA(SA)
01: begin
02:   set α, T, FT, and some parameters of GA;
03:   k := 0;
04:   P(k) := a set of initial feasible solutions;
05:   evaluate structures in P(k)
06:   repeat
07:     k := k + 1;
08:     P(k) := select from P(k − 1);
09:     alter structures in P(k) by crossover (CSEX)
10:     while does not reach the number of offspring do
11:       begin
12:         x := one of offspring generated by CSEX
13:         while does not reach the number of times do
14:           begin
15:             choose an x' ∈ N(x) randomly
16:             if  f(x') < f(x) then
17:               x := x'
18:             else
19:               x := x' with prob. exp(−(f(x') − f(x))/T)
20:           end;
21:       end
22:     evaluate structures in P(k);
23:     T := T * α
24:   until T <= FT;
25:   return the best solution
26: end.
```

Fig. 6. Proposed hybrid GA(SA).

is fixed for each generation, but $T$ is decreased by Step23 in the end process of each generation. Furthermore, since $T$ is fixed for each generation, the running search times of the SA process for each individual are mostly the same. By such a hybrid algorithm, the HGA performs without the switching-time problem, and an efficient HGA can be achieved.

### 5.3. Relation between proposed HGA and crossover

Here, we describe the relation between HGA(SA) and the crossover operator. According to Kido [9], in a process of local search, it is efficient to change to the GA process from the metaheuristic process before the local optimal solution is found. In our HGA, a similar idea is used. In early generations, our HGA does not obtain local optimal solutions after the SA process; however, in the generations in which the HGA(SA) terminates, local optimality is satisfied because the SA is the same as a process of local search when the temperature $T$ is very low. To achieve this optimality in the HGA, it is important to choose the type of crossover operators or other operators. The simple crossover technique has a tendency to move to a different space of a given problem from the current solution states or is very disruptive. However, we have already confirmed that a new solution created by our adopted crossover, the complete subtour exchange crossover (CSEX), does not go too far from the parent solutions [27]. Analogously to the CSEX, the distance preserving crossover DPX taking account of this property has been developed by Freisleben and Merz [24]. Moreover, Kido [9] adopted the heuristic crossover [25]. Thus, we consider that such crossovers are able to create new solutions that are near local optima or parent solutions. Therefore, candidate solutions in a final generation of the HGA(SA) using the CSEX satisfy the local optimality given by the local search algorithm.

## 6. Experimental Results

To confirm the efficiency of the proposed HGA, we compare it with other approaches such as multistart local search and iterated local search algorithms and also we observe whether good subtours are broken by the SA process in the HGA(SA) or not.

### 6.1. Experimental details

In this experiment, all computations of algorithms written in C were performed on an S-4/5 workstation (microSPARC II, 110 MHz). We tested the HGA(SA) algorithm on 15 benchmark instances of 76 to 532-city instances taken from the TSPLIB [22]. Each algorithm performs 10 runs for each instance.

Next, we describe the setting of parameter values and operators in the HGA. The GA or SA requires many parameters, such as the population size $PS$, the crossover rate $p_c$, the mutation rate $p_m$, the initial temperature $T$, the cooling ratio $\alpha$, and so forth. In this experiment, we set $PS$ = 10, and all solutions in the initial population are created randomly. The HGA(SA) is performed until $NG$ generations in which $T$ is very low are reached, provided that we preset suitable values of the initial temperature $T$ and the final temperature $FT$ for sufficient cooling. $NG$ is given in Table 1. The selection operator is the elite strategy [1, 3], which has been a useful method for the TSP or other problems [1, 9, 21]. The population has no identical individuals during operation of the HGA. For the crossover operator, we adopt the CSEX as described above with $p_c$ = 1.0, and the determination of parents is performed randomly from the candidate individuals in the population. The

mutation operator is not used. For the metaheuristic, the SA using the 2-Opt neighborhood structure is combined as described in Section 5. All individuals created by the CSEX are applied to the SA process. The cooling ratio $\alpha$ is set to 0.98. The quality of the solution is calculated by

$$\frac{\text{Obtained Length} - \text{Optimal Length}}{\text{Optimal Length}} \times 100\ (\%)$$

### 6.2. Results and discussions

Table 1 shows the results (best, worst, average percent excess over the optimal length) of the HGA(SA) for several instances of the TSPLIB. "Opt/10 Runs" denotes the number that obtained the optimal solution in 10 runs.

We compare the abilities of the HGA(SA) with Multi-start Local Search (MSLS) [18, 20, 27] and Iterated Local Search (ILS) [18, 20, 27] based on the 2-Opt neighborhood structure. The computation times to terminate the MSLS and ILS are set larger than that spent by HGA(SA). The average computation times (in seconds) that obtained best solutions by the HGA(SA) are as follows: pr76: 118 s, lin105: 232 s, pr107: 130 s, pr124: 126 s, pr152: 361 s, kroA200: 806 s, pr226: 840 s, pr264: 1029 s, pr299: 1537 s, lin318: 1233 s, pr439: 1805 s.

Table 2 shows the results of MSLS and ILS for instances of pr76, lin105, kroA200, and lin318. The MSLS is a method that repeats the 2-Opt local search algorithm many times starting from random solutions. The ILS is a method that repeats the 2-Opt local search starting from new solutions obtained by using a random 4-Opt move [17] to the best local optimal solution previously found by the local search procedure.

Table 1. Experimental results of hybrid GA

| TSPLIB | | HGA(SA) (%) | | | Opt |
|---|---|---|---|---|---|
| Instances | NG | min. | max. | avg. | 10 Runs |
| pr76 | 200 | 0.00 | 0.86 | 0.24 | 7 |
| kroA100 | 200 | 0.00 | 0.38 | 0.12 | 7 |
| rd100 | 200 | 0.01 | 0.70 | 0.12 | 0 |
| lin105 | 200 | 0.00 | 0.41 | 0.04 | 9 |
| pr107 | 200 | 0.00 | 0.40 | 0.04 | 9 |
| pr124 | 200 | 0.00 | 1.37 | 0.19 | 5 |
| kroA150 | 200 | 0.00 | 2.35 | 1.36 | 4 |
| pr152 | 200 | 0.00 | 0.82 | 0.29 | 6 |
| kroA200 | 200 | 0.33 | 2.36 | 1.21 | 0 |
| pr226 | 200 | 0.01 | 0.84 | 0.32 | 0 |
| pr264 | 200 | 0.00 | 1.28 | 0.42 | 3 |
| pr299 | 200 | 0.47 | 2.33 | 0.95 | 0 |
| lin318 | 200 | 0.53 | 2.86 | 1.80 | 0 |
| pr439 | 200 | 0.31 | 3.34 | 1.36 | 0 |
| att532 | 300 | 0.65 | 2.91 | 2.15 | 0 |

Table 2. Experimental results of MSLS and ILS

| | MSLS (%) | | | CT (sec) | Opt |
|---|---|---|---|---|---|
| | min. | max. | avg. | avg. | 10 Runs |
| pr76 | 0.11 | 0.95 | 0.43 | 300 | 0 |
| lin105 | 0.23 | 1.11 | 0.66 | 500 | 0 |
| kroA200 | 1.71 | 5.11 | 3.67 | 1000 | 0 |
| lin318 | 4.56 | 6.17 | 5.61 | 2000 | 0 |
| | ILS (%) | | | CT (sec) | Opt |
| | min. | max. | avg. | avg. | 10 Runs |
| pr76 | 0.00 | 0.41 | 0.24 | 300 | 5 |
| lin105 | 0.00 | 0.79 | 0.17 | 500 | 7 |
| kroA200 | 0.37 | 1.57 | 1.39 | 1000 | 0 |
| lin318 | 1.36 | 2.69 | 2.00 | 2000 | 0 |

Next, we compared our HGA(SA) and a GA reported in the past. Ulder and colleagues evaluated a GenLK based on the LK heuristic algorithm and a Gen2-Opt based on 2-Opt local search applied to instances of the TSPLIB [12]. The crossover used in both algorithms was the maximal preservative crossover [13], which is a promising crossover operator [17, 18, 20, 23]. For lin318, GenLK obtained solutions of 0.13%, and Gen2-Opt obtained solutions of 2.02% on the average. In Table 1 our HGA(SA) obtained a better average solution of 1.80% than that of Gen2-Opt. Thus, the HGA(SA) could not obtain better solutions than the GenLK, but could obtain better results than that of the Gen2-Opt. This result produced by our HGA(SA) is not a limit. Since the SA is a metaheuristic and can be based on more powerful local search algorithms such as 3-Opt or LK heuristics, if such a powerful heuristic is incorporated into the SA, even better results by the HGA(SA) can be expected.

### 6.3. Experiment on subtours

The HGA(SA) contains the CSEX which inherits as many good subtours as possible, because they are worth preserving for descendants. However, it is considered that these good subtours are broken by the SA. To confirm this, we now investigate how many subtours are inherited in the next generation. In this experiment, we randomly choose 10 surviving subtours in which each subtour consists of three cities in a part of the optimal solution, and observe the frequency of appearance of the surviving subtours during the search process of the HGA(SA).

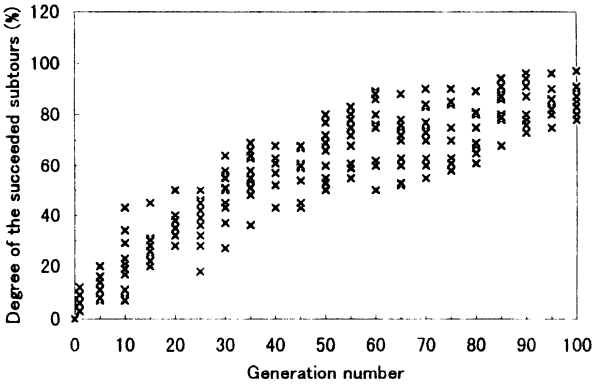


Fig. 7. Appearance degree of the succeeded subtours by hybrid GA(SA).

Figure 7 shows the frequency of the surviving subtours under the HGA(SA) for a 105-city instance, lin105. From the figure, we confirmed that the number of subtours increased in this search. In the early generations, the number of subtours was smaller because the temperature of the SA was high. However, after that, the number of surviving subtours increased gradually. We observed the same tendency in other instances. We therefore confirmed that good subtours are not likely to be broken by the process of the SA in the HGA(SA).

## 7. Conclusions

This paper presented an approach to the hybrid genetic algorithm (HGA) that did not incur the switching-time problem which occurred in the case of combining a metaheuristic (simulated annealing algorithm, SA) with a genetic algorithm. Our HGA was not a standard hybrid algorithm that executed all of the SA process in each generation, but rather an algorithm that decreases the temperature of the SA in each generation using a fixed search time depending on the given instance size, and finally reaches local optimality. In our experiment, we investigated the efficiency of our HGA applied to the traveling salesman problem, and compared it with useful approaches such as the iterated local search algorithm. Finally, we confirmed that good subtours were not broken by the SA of the metaheuristic.

**Acknowledgments.** The authors thank the referees for valuable comments and suggestions.

## REFERENCES

1. Goldberg DE. Genetic algorithms in search, optimization, and machine learning. Addison–Wesley; 1989. p 1–88.
2. Michalewicz Z. Genetic algorithms + data structure = evolution programs. Springer-Verlag; 1992. p 11–94, 165–191.
3. Kitano H, editor. Genetic algorithm. Sangyo Tosho; 1993. (in Japanese)
4. Bell MO, Magnanti TL, Nemhauser GL, Monma CL. Handbooks in operations research and management science network models. North-Holland; 1995. Vol. 7, p 225–330.
5. Brady RM. Optimization strategies gleaned from biological evolution. Nature 1985;317:804–806.
6. Yamamura M, Ono T, Kobayashi S. Character-preserving genetic algorithms for traveling salesman problem. JSAI 1992;7:1049–1059. (in Japanese)
7. Yagiura M, Nagamochi H, Ibaraki T. Two comments on the subtour exchange crossover operator. JSAI 1995;10:464–467. (in Japanese)
8. Uno T, Yagiura M. Fast algorithms to enumerate all common intervals of two permutations. Tech Rep 96015, Department of Applied Mathematics and Physics, Graduate School of Engineering, Kyoto University, 1996.
9. Kido T. Hybrid searches using genetic algorithms. In: Kitano H, editor. Genetic algorithm. Sangyo Tosho; 1993. p 61–88. (in Japanese)
10. Brown DE, Huntley CL, Spillane AR. A parallel genetic heuristic for the quadratic assignment problem. Proc 3rd ICGA, p 406–415, 1989.
11. Prinetto P, Rebaudengo M, Reordo MS. Hybrid genetic algorithms for the traveling salesman problem. In: Albrecht RF, editor. Artificial neural nets and genetic algorithms. Springer-Verlag; 1993. p 559–566.
12. Ulder NLJ, Aarts EHL, Bandelt H-J, van Laarhoven PJM, Pesch E. Genetic local search algorithms for the traveling salesman problem. In parallel problem solving from nature, p 109–116, 1991.
13. Mühlenbein H, Gorges-Schleuter M, Krämer O. Evolution algorithms in combinatorial optimization. In: Parallel computing 7. North-Holland; 1988. p 65–85.
14. Mühlenbein H. Evolution in time and space—The parallel genetic algorithm. In: Rawlins G, editor. Foundations of genetic algorithms. Morgan-Kaufmann; 1991. p 316–337.
15. Lin S, Kernighan BW. An effective heuristic algorithm for the traveling salesman problem. Oper Res 1973;21:498–516.

16. Johnson DS, Aragon CR, McGeoch LA, Schevon C. Optimization by simulated annealing: An experimental evaluation; part I, graph partitioning. Oper Res 1989;37:865–892.
17. Johnson DS. Local optimization and the traveling salesman problem. Proc 17th Colloquium on Automata, Lang., and Prog. Springer-Verlag; 1990. p 446–461.
18. Johnson DS, McGeoch LA. The traveling salesman problem: A case study. In: Local search in combinatorial optimization. Wiley–Interscience Series in Discrete Mathematics and Optimization, p 215–310, 1997.
19. Glover F. Tabu search—part I. ORSA J Comput 1989;1:190–206.
20. Kubo M. Meta-Heuristics. Discrete structure and algorithm IV. Kindaikagakusya; 1995. p 171–230. (in Japanese)
21. Tajima T, Nakano K, Ichikawa M, Maeda H. A real-time path planning using genetic algorithms. JSAI 1995;10:94–104. (in Japanese)
22. Reinelt G. http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/TSPLIB95/TSPLIB.html
23. Mathias K, Whitley D. Genetic operators, the fitness landscape and the traveling salesman problem. In parallel problem solving from nature, p 219–228, 1992.
24. Freisleben B, Merz P. New genetic local search operators for the traveling salesman problem. In parallel problem solving from nature IV, p 890–900, 1996.
25. Grefenstette J, Gopal R, Rosmaita B, Van Gucht D. Genetic algorithms for the traveling salesman problem. Proc 1st ICGA, p 160–168, 1985.
26. Katayama K, Sakamoto H, Narihisa H. An efficiency of hybrid mutation genetic algorithm for traveling salesman problem. Proc 2nd Australia–Japan Workshop on Stochastic Models in Engineering, Technique & Management, Gold Coast, Australia, p 294–301, 1996.
27. Katayama K, Hirabayashi H, Narihisa H. Performance analysis of a new genetic crossover for the traveling salesman problem. IEICE Trans Fundam 1998;E81-A:738–750.

## AUTHORS (from left to right)

**Kengo Katayama** (member) received his B.E., M.E., and D.Eng. degrees in system science from Okayama University of Science in 1992, 1995, and 1998. He is presently a research associate in information and computer engineering, Okayama University of Science. His research interests are in the area of algorithms for the combinatorial optimization problems and VLSI routing.

**Hiroyuki Narihisa** (member) received his M.E. degree in electrical engineering and his D.Eng. degree in applied mathematics and physics from Kyoto University in 1964 and 1970. From 1983 to 1994, he headed the Information Processing Center of Okayama University of Science. Since 1980, he has been a professor in information and computer engineering, Okayama University of Science. His research interests are in the field of operations research and system analysis for optimization.

Copyright of Electronics & Communications in Japan, Part 3: Fundamental Electronic Science is the property of Wiley Periodicals, Inc. 2004 and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.

Copyright of Electronics & Communications in Japan, Part 3: Fundamental Electronic Science is the property of John Wiley & Sons, Inc. and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.