Application of Genetic Recombination to Genetic Local Search in 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

In this paper, an approach based upon Genetic Recombination is proposed, and applied to the Traveling Salesman Problem (TSP). The algorithm is composed of two Sample Genetic Algorithms (SGAs) which only consist of the basic genetic operators such as selection, crossover and mutation. One of the SGAs is named as the Global Genetic Algorithm (GGA), and encompasses the main tours where it is designed to search for the global optimal solutions. Another one is named as the Local Genetic Algorithm (LGA) and traverses over the sub tours to find the local optimal solutions. The LGA is combined to the GGA as an operator. The local optimal solutions are recombined to the main tours for improving the search quality. To investigate the features of the proposed algorithm, it is applied to a small double circles TSP, and some interesting results are presented in our experiments.

Keyword: genetic algorithm, local search, genetic recombination, traveling salesman problem

I. Introduction

TSP is one of the well-studied combinatorial optimization problems [8], [21]. Many researchers from various fields have devoted to developing new algorithms for solving it. In the TSP, each distance between two cities is given for a set of n cities. The goal is to find the shortest tour. There are currently three general classes of heuristics for the TSP: classical tour construction heuristics such as the Nearest Neighbor method, the Greedy algorithm, and local search algorithms based on re-arranging segments of the tour [18]. Many progressive results have been presented in the previous studies during the recent years even though there are still improvable spaces with the search algorithms, which have been applied to the TSP, such as ant colonies[15], local search[6], neural

networks[13], simulated annealing [20], tabu search [3], and genetic algorithms [12]. It has been proved that the hybrid of different algorithms is more effective than a single algorithm. For example, the local search has been successful for improving GAs in the search processes [1], [19], [22]. Most of the works on solving the TSP are focused on the efficiency of how to solve the larger TSP instances. Some of the works aim at expanding the theories of search algorithms, especially in the GA domain.

Our attempt is to make a discussion on the GAs based on Genetic Recombination in this paper. The algorithm is composed of two SGAs which only contain basic genetic operators. One is the GGA which is applied to the main tours, for searching for the global optimal solutions. Another is the LGA which is applied to the sub tours for searching for local optimal solutions. The SGA was developed by John Holland, and his original algorithm is approximately the same as shown in the Fig.1. [10]. In the early studies, the SGA played an important role in the development of GAs, and attracted the researchers' attentions widely. Goldberg made a detailed discussion on how the SGA works with some simple optimal mathematical functions and other problems in his book [5]. Reeves discussed the differences, and similarities between the SGA and the neighborhood search [4]. Michael D. Vose provided an introduction to what is known about SGA theory. He also made available algorithms for the computation of mathematical objects related to SGA [16]. All the studies have shown that the SGA is still valuable in heuristic search algorithms.

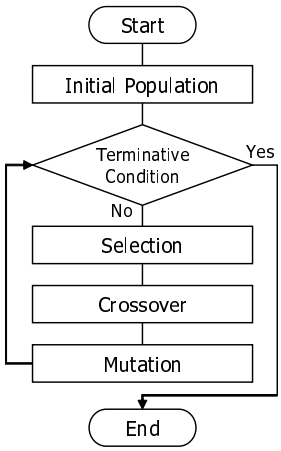


Fig.1 Simple GA

II. Local Search in GAs

Every successful strategy to produce near-optimal solutions necessarily relies upon the local search algorithm. All these algorithms differ with respect to their neighborhood structures. Any such structure specifies a set of neighboring solutions that are in some sense close to that solution. The associated local improvement operator replaces a current solution by a neighboring solution of better value if possible. The local search algorithm is repeated several times, retaining the best local optimum found. Our works focus on the local search algorithms. In the hybrid of the Local Search and the GAs, the design of the local operator is very important. There are many local search heuristics which have been combined into GAs for the TSP. For example, the well-known 2-Opt heuristic has been used to optimize the TSP tours in connection with the GAs [21]. 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 2-Opt results in an improved tour, the change is preserved. Otherwise, the tour is returned to the original form. The 2-Opt uses a pair of edges which is formed from 4 cities in the tour. It is a powerful local search operator used in the GAs for TSP. Compared with the 2-Opt, k-Opt (k=3, 4, …) uses more edges to rearrange the tours.

Moreover, some crossover methods also emphasize on the use of local operations. Crossover methods usually recombine two individuals to reproduce the new individuals, but most of them seldom utilize the shorter edges while they are applied to the tour. Some crossover methods aimed at the phenotype of the edges use more cities or a set of cities according to the visiting order in the tour. For example, YAMAMURA proposed the sub tour exchange crossover (SXX) which mutually exchanges the parts which contain the same continuous cities in two individuals [17]. MAEKAWA proposed the edge exchange crossover (EXX) which utilizes more edges from different individuals [14]. Both of the methods are powerful for solving the TSP. However, there is a problem with few variations of the child individuals because the SXX is only available when the parts which composed of the same cities in two various individuals are found.

Except for the TSP, local search is also used in many fields [2], [7], [23]. For example, H. Suzuki et al. proposed a local search algorithm using a vision-related technique for a real-time visual servoing [11]. The method utilizes the global search feature of a GA and a local search technique of the GA. Their local operations are very similar to our LGA. They used a sub population generated from elite individual of the global GA, then only applied the mutation to the sub population to improve the local optimal solution.

Our approach is to use a sub tour which contains a set of continuous cities ordered according to the visiting order. The basic idea is to find a better sub tour to replace the original one. This operation acts like the Genetic Recombination in the Genetic Engineering field. Genetic Recombination is the process by which the combination of genes in an organism's offspring is different from the combination of genes in that organism. This definition is commonly used in classical genetics, evolutionary biology, and population genetics. Commonly, one gene or a set of a few foreign genes is taken out of the DNA of one organism and inserted into the DNA of another organism by an artificial manipulation of genes. The manipulation disrupts the ordinary command code sequence in the DNA. This disruption may make the individual better if it is applied judiciously. As we knew, there are many good phenotypes of plants that have been created in biological field with Genetic Recombination. The technical problem is how to find out which are the better GENES and to replace the original ones with them. In the TSP, it is the technique related to the local search. To find the better cities in the tour, we chose a set of contiguous cities from the main tour to form a sub tour, and then apply the LGA to the sub tour to find better solutions to feed into the main tour. This operation cultivates fitter GENES in a way that mimics biology, the details of which are described in the following section.

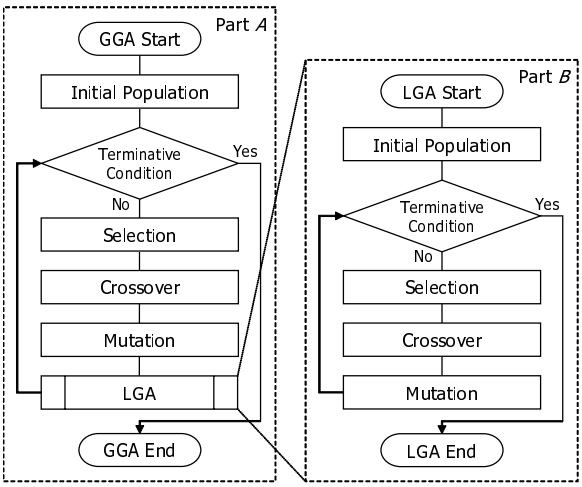


Fig.2 Proposed Algorithm

III. Proposed algorithm

The proposed algorithm is shown in Fig.2. The parts A and B in the figure are both separately a SGA before part B is intercalated into part A. Part B is the LGA in the algorithm. The two parts form the new algorithm-GGA. Part B performs as a local search operator in the GGA. The GGA is applied to the main tour for searching for global optimal solutions. To obtain the global optimal solutions in the process of the TSP, the GGA may just need to improve the order of a set of cities in the best tour, especially in the latter stages of the process. A set of continuous cities are chosen to create the sub tour in the experiment. The LGA is applied to the sub tour, for finding the local optimal solution to replace the original part chosen from the main tour.

There are two methods for initializing the sub tour: (1): Keep the original part from the main tour as an individual in the population of the LGA, and then randomly initialize the sub tour, except for the start and end cities to match the population size of the LGA. (2): Only reproduce the original part from the main tour to the population size of the LGA. It means the population is composed of same individuals as the original part. The same operation connoted in both initialization methods (1) and (2) is that the original parts from the main tours are preserved in the population of the LGA. If the solution of the LGA is shorter than the original part, the main tour will be improved when it is recombined. If no solution is found to be shorter than the original part in the LGA, then the original part will be recombined back to the main tour. Both of the initialization methods are available for a small sub tour in the LGA. But, the first method is not suggested for a big sub tour as it takes too long a time to initialize. The local operations described in Suzuki's paper are the same as the second initialization method used in our algorithm for initializing the LGA. The difference is that we used a SGA which contains crossover and selection except for the mutation and applied the local search to more elite individuals. Only one elite individual is selected to create the sub population in Suzuki's algorithm.

In the LGA, the sub tour is an open tour. The length of the tour is calculated from the start city to the end city, not including the distance between the end city and the start city. Another important point is that all individuals in the population of the LGA have the same start and end cities during the processing. This imperative is for avoiding the main tour becoming longer at the connection points after the recombination of the sub tour.

GGA and LGA are complete genetic algorithms; hence all genetic operators for solving the TSP are applicable to them. In the experiments, only three genetic operators have been selected: Crossover, Mutation and Selection. Crossover is a one point crossover method with two random individuals. Mutation is randomly carried out on the individuals by changing the order of two cities chosen randomly within a tour. Selection is carried out to replace the four longest individuals with the two shortest ones in the population. In the LGA, the mutation and crossover keep the diversity of the population, so the mutation rate is set higher than in the GGA. The terminative conditions are set to the total number of generations in the GGA and the LGA.

Obviously, it is computationally expensive running a big LGA in every generation of the GGA. But, it is believed that the LGA could be seen as an island for improving the elite individuals of the main population in distributed or parallel processing (This will be discussed in our future works). A small LGA is effective as an operator in the GGA, which is confirmed latter in the results.

Local Operations: The tour which consists of n cities is expressed as $c=c_0, \ldots, c_i, c_{i+1, \ldots,} c_{n-1}$ (n>=4). The distance $d(c_i, c_{i+1})$ is given for the pair of cities c_i and c_{i+1}. All cities are coded using path representation.

1. Randomly choose one main tour $c= c_0, \ldots, c_i, c_j, \ldots, c_{n-1}$ from population of the GGA.

2. Randomly choose one sub tour which contains N_s continuous cities in the main tour c. The sub tour is $c_s=c_{i, \ldots,} c_j$. The start city is c_i and the end city is c_j. j-i > 4. This is the first individual of the LGA. The length of the first individual is set as d_0.

3. Reproduce the first individual to the population size of the LGA. The start city c_i and the end city c_j are kept unchanged. The population of the LGA is created with P_s same individuals.

4. Run the LGA in every generation of the GGA. The start and end cities of all individuals of the LGA are kept unchanged during the processing.

5. The best individual in the population of the LGA is obtained when the LGA stopped. Its length is set as d_1.

6. The best individual is recombined back into the main tour to replace the original part.

Let the ratio $Ratio=d_0/d_1$, where the range is $Ratio > 1.0$. The Ratio is discussed again in the next section. The cities of sub tour in the LGA ranges from 4 to n. The minimum size of the sub tour is set to 4 cities due to the start and end cities being fixed in the sub tour during the processing. The maximum size is set to n because it is the largest improvable part in the main tour. The LGA performs as the reversion operator in the GGA when the sub tour only contains 4 cities, and the start and end cities are fixed.

IV. Results and Discussions

The main tour which contains n cities is given as $c=c_0, \ldots, c_i, c_{i+1}, \ldots, c_{n-1}$ where the distance between two cities c_i, c_{i+1} is $d(c_i, c_{i+1})$. A sub tour which contains m cities from the main tour is $c=c_0, \ldots, c_k, c_{k+1}, \ldots, c_{m-1}$ where the distance between two cities c_k, c_{k+1} is $d(c_k, c_{k+1})$. Total distances of the main tour and the sub tour are d_m and d_s respectively:

$$d_m = \sum_{i=0}^{n-1} d(c_i, c_{i+1}) \qquad (1)$$

where $c_n=c_0$.

$$d_s = \sum_{k=o}^{m-2} d(c_k, c_{k+1}) \qquad (2)$$

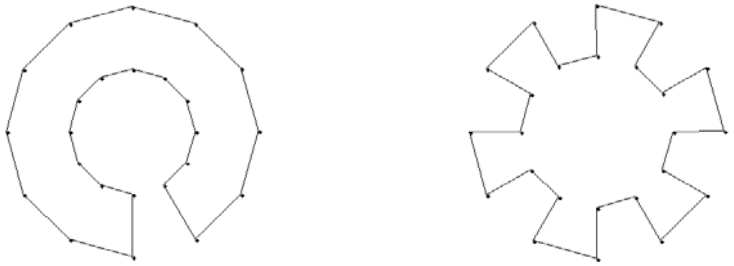
The d_m and d_s are defined as the Fitness of the individuals. The shorter the distance is, the higher the Fitness is, in other words, the fitness is inversely proportional to the distance. The parameters are shown in Table 1.

Our source code is written in Java and run on a PC (CPU: Pentium III 1.0GHz, RAM: 256MB) with Microsoft Windows2000 Operating System.

The TSP instance of the double concentric circle, which contains only 24 cities, is used in the experiments (Fig.3). The ratio of the inner radius (R_i) and the outer radius (R_o) is Ri/Ro. If $R_i/R_o < 0.58879$, the optimal tour is C-type (Fig.3-a). If $Ri/Ro > 0.58879$, the optimal tour is O-type (Fig.3-b).

Table 1 Parameters in the GGA and the LGA.

	GGA (Main tour)	LGA (Sub tour)
Population size	100	80
Cities	24	4 ~24
Generations	1,000 0	10, 20, 30, 40, 50, 60, 70, 80, 90, 100
Crossover rate	75.0%	75.0%
Mutation rate	2.0%	2.5%
Selection	2 shortest individuals replace the 4 longest ones	2 shortest individuals replace the 4 longest ones



a. C Type, Ri/Ro<0.58879 b. O Type, Ri/Ro>0.58879

Fig.3 TSP instances

A. Distributions of Optimal Solutions

Fig.4 shows the number of the optimal solutions which is obtained in 20 runs. A peak appears around the mid point, where the number of cities in the sub tour is about half that of the main tour. The Fitness is shown in Fig.5. The process is stopped at the 1,000th generation. The process converges faster with the increase of the number of cities in the sub tour at the beginning. This happens because there are many more improvable spaces in a big sub tour of the LGA. Stable convergences appears when the city numbers of the sub tour range around half that of the main tour. The result is not satisfied when the LGA performs as the reversion operator in the experiments.

B. Computation Time

The GAs usually take a long time to run to reach a good result. Consequently it increases the computation time greatly by combining the LGA into the GGA. The computation time was examined by increasing the city number and the generations in the LGA respectively.

The results are shown in Fig6. and Fig.7. The computation time linearly became longer with the increase of the number of cities and the generations in the LGA. Decreasing the number of cities and generations in the LGA took a shorter time, but it might not be sufficient for finding a better sub tour to improve the main tour.

C. Time and Generations for obtaining the Optimal Solutions

Fig.8 is the figure of the computational time for obtaining the optimal solutions. There are 3 points on every vertical line; the low, upper, and central points show the shortest, longest, and mean time, respectively for obtaining the optimal solutions in 20 runs. The time for finding the optimal solutions increases with the number of cities in the sub tour. In Fig.9, the three points on the lines show the earliest, latest, and mean generations for reaching the optimal solutions respectively in 20 runs. The earliest generations appeared when the number of cities in the sub tour ranged around half the number of cities of the main tour. Some optimal solutions appeared early when the sub tours contained more cities. But, it takes a far longer computational time when the number of cities of the sub tour is increased as stated above.

D. Recombination Rates and Ratios

Fig.10 is the figure of the recombination rates. We suppose the number of the individuals which were recombined with the LGA is N_r and the number of the individuals which were searched by the LGA is N_s. The recombination rate $\mathrm{Rate} = N_r / N \times 100$. The recombination rates increase with the increase of the generations and the number of cities in the sub tours. More generations are advantageous for finding a better sub tour to carry out the recombination, but it takes a longer time to run the LGA. The same trend appears with the changes of the number of cities because there are more improvable spaces in the bigger sub tour. The part chosen from the main tour is easier to improve. Fig.11 is the ratio figure. The ratio Ratio = d_0/d_1 and Ratio > 1.0, where d_0 is the original

sub tour from the main tour and d_1 is a sub tour searched to be shorter in the LGA. The peaks of the ratios appear when the number of cities in the sub tour ranged from 8 to 13. The higher ratio means a sub tour was found to be shorter in the LGA than the original part from the main tour. The result is correlative with the distribution of the optimal solutions in Fig.4.

E. Dynamics of the LGA

Fig.12 and Fig.13 show the dynamics of the LGA. Fig.12 presents the ratio distributions of a single GGA in which the optimal solution is obtained at the 252nd generation. Because the original part from the GGA is preserved in the LGA and the ratio Ratio = d_0/d_1, the value of the Ratio > 1.0. It indicates that the LGA creates a shorter sub tour and recombines it to the GGA when Ratio > 1.0. The LGA works efficiently and more ratios are bigger than 1.0 before the optimal solution is obtained at the 252nd generation. When the population converges to the optimal solution, the LGA can not create a better sub tour than the original part from the main tour and the ratios become 1.0. Fig.13 shows the ratio distributions of another GGA in which the premature convergence occurs at the 151st generation. There are many points distributed over 1.0 after the premature convergence occurs, which indicates that the LGA still works efficiently even though the GGA reaches the premature convergence.

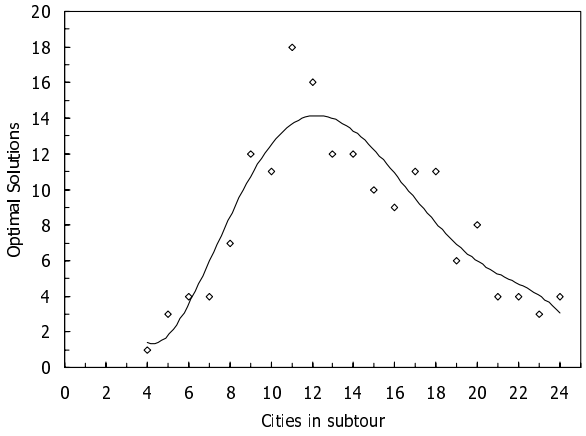


Fig.4 Distribution of optimal solutions

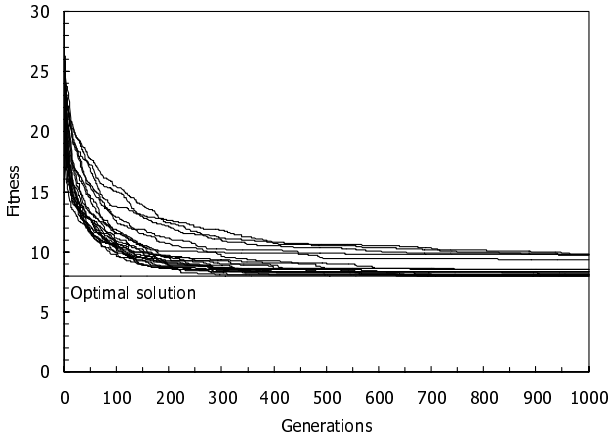


Fig.5 Fitness

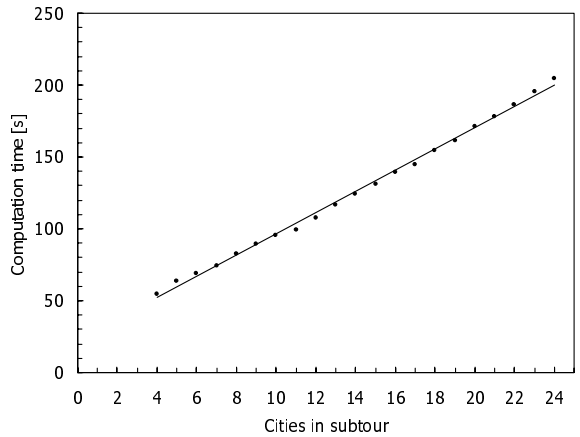


Fig.6 Computation time with cities increase

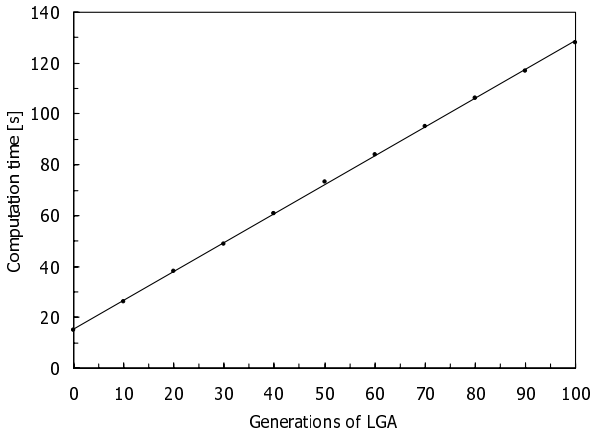


Fig.7 Computation time with generations increase.

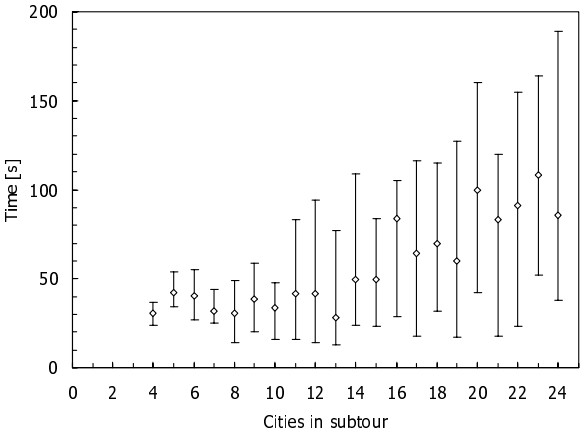


Fig.8 Time of optimal solutions

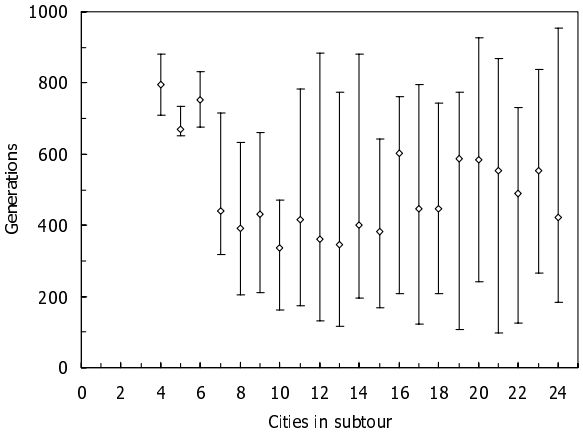


Fig.9 Generation of optimal solutions

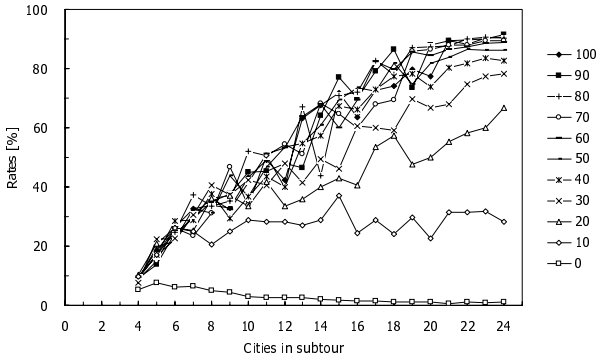


Fig.10 Rates, generations and population sizes.

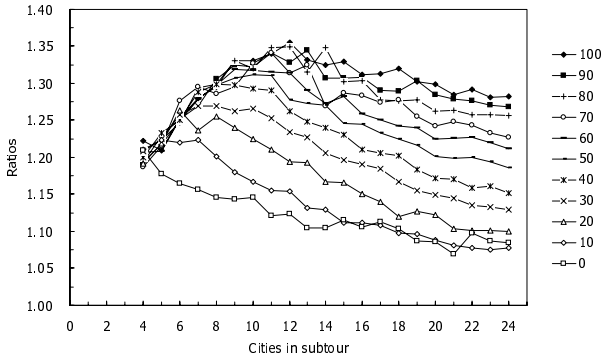


Fig.11 Ratios, generations and population sizes.

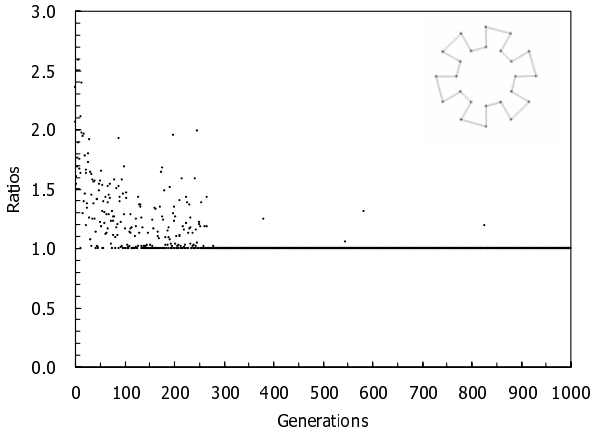


Fig.12 Generation of optimal solution: 252nd

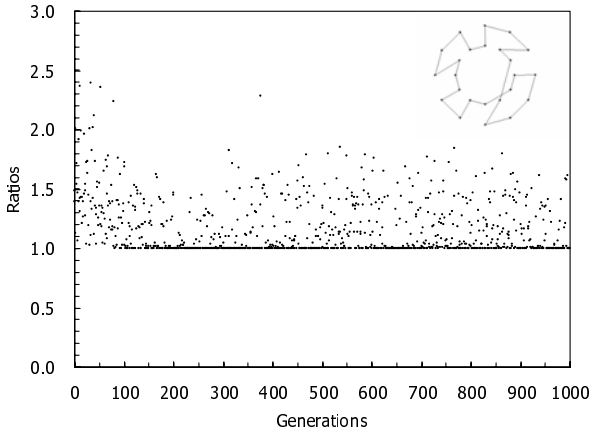


Fig.13 Generation of local solution: 151st

V. Summary

A local search algorithm based on genetic recombination is discussed in this paper. The LGA acts as a local search operator in the GGA. A good result is presented when the number of cities of the sub tour is set to around half the number of cities of the main tour. We think it would be reasonable running a small LGA for a big TSP instance. It may be more effective in distributed and

parallel processing running the LGA as an island to improve the main tour. This will be discussed in our future works.

Acknowledgement

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

References

[1] A. Homaifar, S. Guan, and G. E. Liepins, A New Approach to the Traveling Salesman Problem by Genetic Algorithms, in Proceedings of the 5th International Conference on Genetic Algorithms, pp. 460-466, Morgan Kaufmann, 1993.

[2] A. Jaszkiewicz, Genetic Local Search for Multiple Objective Combinatorial Optimization, European Journal of Operational Research, Vol. 137, No. 1, pp. 50-71, 2002.

[3] C.-N. Fiechter, A Parallel Tabu Search Algorithm for Large Traveling Salesman Problems, Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science, Vol. 51, pp. 243-267, 1994.

[4] C. R. Reeves, Genetic Algorithms and Neighborhood Search, in Evolutionary Computing, AISB Workshop, pp. 115-130, Leeds, U.K., 1994.

[5] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, MA, 1989.

[6] D. S. Jonhson, and L. A. McGeoch, The Traveling Salesman Problem: A Case Study in Local Optimization, in E. H. L. Aarts and J. K. Lenstra (Eds.), Local Search in Combinatorial Optimization, Wiley & Sons, New York, 1996.

[7] B. Dengiz, F. Altiparmak, and A. E. Smith, Local Search Genetic Algorithm for Optimal Design of Reliable Networks, IEEE Trans. on Evolutionary Computation, Vol. 1, No. 3, pp. 179-188, 1997.

[8] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys, The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization, Wiley & Sons, New York, 1985.

[9] G. Reinelt, The Traveling Salesman: Computational Solutions for TSP Applications, Lecture Notes in Computer Science, Vol. 840, Springer-Verlag, Berlin, Germany, 1994.

[10] J. H. Holland, Adaptation in Natural and Artificial Systems, The University of Michigan Press, 1975.

[11] H. Suzuki, and M. Minami, Visual Servoing to Catch Fish Using Global/Local GA Search, IEEE/ASME Transactions on Mechatronics, Vol. 10, No. 3, pp. 352-357, 2005.

[12] J. Grefenstette, R. Gopal, B. Rosimaita, and D. V. Gucht, Genetic Algorithms for the Traveling Salesman Problem, in Proceedings of an International Conference on Genetic Algorithms and their Applications, pp. 160-168, 1985.

[13] J.-Y. Potvin, The Traveling Salesman Problem: A Neural Network Perspective, ORSA Journal on Computing, Vol. 5, pp. 328-348, 1993.

[14] K. Maekawa, H. Tamaki, H. Kita, and Y. Nishikawa, A Method for the Traveling Salesman Problem based on the Genetic Algorithm, Transactions of the Society of Instrument and Control Engineers, Vol. 31, No. 5, pp. 598--605, 1995. (in Japanese).

[15] L. M. Gambardella, and M. Dorigo, Ant-Q: A Reinforcement Learning Approach to the Traveling Salesman Problem, in Proceedings of the 12th International Conference on Machine Learning, pp. 252-260, Morgan Kaufmann, 1995.

[16] M. D. Vose, The Simple Genetic Algorithm: Foundations and Theory, MIT Press, Cambridge, MA, 1999.

[17] M. Yamamura, T. Ono, and S. Kobayashi, Character-Preserving Genetic Algorithms for Traveling Salesman Problem, Journal of the Japanese Society for Artificial Intelligence, Vol. 7, No. 6, pp. 1049-1059, 1992. (in Japanese).

[18] P. C. Kanellakis, and C. H. Papadimitriou, Local Search for Asymmetric Traveling Salesman Problem, Operations Research, Vol. 28, No. 5, pp. 1086-1099, 1980.
[19] R. M. Brady, Optimization Strategies Gleaned from Biological Evolution, Nature, Vol. 317, pp. 804-806, 1985.
[20] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, Optimization by Simulated Annealing, Science, Vol. 220, pp. 671-680, 1983.
[21] S. Lin, and B. Kernighan, An Efficient Heuristic Procedure for the Traveling Salesman Problem, Operations Research, Vol. 21, pp. 498-516, 1973.
[22] T. G. Bui, and B. R. Moon, A New Genetic Approach for the Traveling Salesman Problem, in Proceedings of the First IEEE Conference on Evolutionary Computation, pp. 7-12, 1994.
[23] W. Han, and P. Que, Defect Reconstruction of Submarine Oil Pipeline from MFL Signals Using Genetic Simulated Annealing Algorithm, Journal of the Japan Petroleum Institute, Vol. 49, No. 3, pp. 145-150, 2006.

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. He is currently an associate professor at Oita National College of Technology in the field of Computer and Control Engineering. 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.