2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Knowledge Acquisition Issues for Intelligent Route Optimization by Evolutionary Computation

Masaki Suzuki and Setsuo Tsuruta
School of Information Environment
Tokyo Denki University
Inzai, Japan

Rainer Knauf
Faculty of Computer Science
and Automation
Ilmenau University of Technology
Ilmenau, Germany

Yoshitaka Sakurai
The School of Interdisciplinary
Mathematical Sciences
Meiji University
Nakano, Japan

***Abstract*—The paper introduces a Knowledge Acquisition and Maintenance concept for a Case Based Approximation method to solve large scale Traveling Salesman Problems in a short time (around 3 seconds) with an error rate below 3 %. This method is based on the insight, that most solutions are very similar to solutions that have been created before. Thus, in many cases a solution can be derived from former solutions by (1) selecting a most similar TSP from a library of former TSP solutions, (2) removing the locations that are not part of the current TSP and (3) adding the missing locations of the current TSP by mutation, namely Nearest Insertion (NI). This way of creating solutions by Case Based Reasoning (CBR) avoids the computational costs to create new solutions from scratch.**

I. Introduction

Due to the complicated road network, the product distribution is less efficient than that of the USA, which disadvantages the productivity of the Japanese industry. This inefficiency also causes social problems and economical losses. Namely, we are facing the necessity of urgently reducing the volume of car exhaust gases to meet environmental requirements as well as curtailing the transport expenses in Japan.

There are many distribution systems that should be optimized, including the delivery of parcels, letters and products distribution across multiple ventures. In order to improve their efficiency, it is necessary to optimize the delivery routes, or the delivery order of multiple delivery locations (addresses). One round delivery route includes more than several tens or hundreds different locations. Thus, the optimization of a delivery route can be modeled as a large scale Traveling Salesman Problem (TSP).

However, the TSP is a combinatorial problem that causes computational explosion due to the $n!$ order of combinations for a n city TSP. Therefore, to practically obtain an efficient delivery route of such a distribution system, a near optimal solving method of TSP is indispensable.

Yet, the practical use of such a solving method on an actual site needs human confirmation of the solution, since social and human conditions are involved. Namely, human users should verify that the solution is practical. Users sometimes need to correct manually or select an alternative solution to meet miscellaneous technical and social side conditions. Therefore, TSP solving methods are required to ensure a response time necessary for the above human interaction.

By the way, solutions generated by domain experts may have 2-3% of deviation from the mathematical optimal solution, but they never generate worse solutions, which may cause practical problems. On the other hand, conventional approximate methods for solving TSP [9][14][25] may generate even mathematically optimal solutions in some particular cases but cannot ensure that the amount of errors is below 2-3%. Errors more than 4% possibly discourage users, which makes those conventional methods not practically useful for delivery route schedule.

There are some rather simple ways to solve a TSP heuristically. For instance, users create a delivery schedule by repeating simple methods such as stepwise inserting new delivery locations at a position that minimizes the gain of travelling distance. This is a method called "Nearest Insertion" (NI)[1]. Also, users improve the delivery route by repeating exchanges sub-routes in the delivery schedule. This is a method called "2-opt".

To cope with this, we proposed some types of Genetic Algorithms (GAs) [20]. These GAs incorporated some simple heuristics aiming at interactive real-time response as well as avoiding significant errors[2] for any kinds of delivery location patterns.

Our first approach, a GA incorporating 2-opt type heuristics, tends to fall into a local minimum solution with certain delivery location patterns though it performs well for other patterns. Therefore, we proposed a multi-outer-world GA (Mow-GA) [20]. This method overcomes some 2-opt type GA's drawbacks by NI type GAs with 2-opt type GAs.

However, the Mow-GA has also some defects. The Mow-GA starts an NI type GA from the first generation without inheriting the information of the elite individuals generated by the 2-opt type GA. Within about 3 seconds, it cannot have enough performance for large scale TSPs. Moreover, an attempt to compensate this drawback by adding a complementary approach did not work as well [20].

To overcome this problem, as an improvement of the Mow-GA, a Multi-inner-world Genetic Algorithm (Miw-GA) was proposed [21]. In this GA, the search efficiency was improved by supplements that address the weak points by processing two types of heuristics (2-opt and NI) in each generation.

[1]Some references call it "Cheapest Insertion".
[2]By "error", we denote deviations from the optimal solution.

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

To simplify the search processes and to make them clearly understandable, the searching operator of the GA was limited to the mutation to improve individuals. The Miw-GA can obtain high accurate solutions for various kinds of delivery location patterns within interactive response time. However, when the number of cities in the TSP became more than 200, the average error rate from the optimal solution exceeded 4% in some cases.

To overcome this drawback, a so-called Backtrack and Restart GA (BR-GA) [22] was proposed, which maintains the diversity of a population by conducting a random restart and fostering new children. This GA achieves below 3% level error rate for less than 1000 cities TSPs within 3 seconds.

Even the world's fastest exact algorithm called Concorde [1] needs 10-100 seconds to solve the same size TSPs. As to approximation algorithms, LKH [17][10] can solve below 1000 cities TSPs within about 3 seconds. This is the world's top level compromise between both accuracy and speed in solving TSPs by an effective branch & cut search.

Our BR-GA can also solve below 2000 cities TSPs within 10 seconds. However, LKH is an improved version of the fairly complex method called LK. This is not easy for application field experts to understand and modify solutions according to domain conditions. Experts in the delivery application field are apt not to use complex methods that are too difficult to understand.

An exact solution or too much accuracy is not valuable since 2-3% error is not recognizable for field experts. Moreover, complex methods have problems in flexibility. The developers of delivery schedules have difficulties in modifying solutions generated by complicated methods deriving solutions suitable for the deliverer's convenience.

In practice, new TSPs distinguish from formerly solved ones very slightly. The changes in the location patterns are less than 10 - 30 % and about 5 % in average. This insight opens ways to improve the computational complexity in solving large scale TSPs. The contribution of the present paper is a method to solve TSPs by Case Based Reasoning (CBR). Here, previous TSP solutions are collected in a case base (CB). A new TSP is solved by

1) retrieving the most similar TSP in the CB,
2) mutating its solution towards a solution of the current TSP by (a) removing the redundant locations from the library solution and (b) adding the deficient locations of the current TSP by Nearest Insertion (NI), and,
3) in case the newly created solution TSP meets some fitness requirements, adding it to the case library.

This paper is organized as follows. In the next (second) section, the delivery route optimization problem and its technical issues are described. In the third section, related work is summarized. Section four introduces our method for solving the problem.

The main focus in this paper is a concept for the Case Base maintenance. In the fifth section, experiments to validate the solving method, its effects and its results are shown and section six concludes the paper.

II. Delivery Route Optimization

The delivery route optimization problem of these distribution systems is formulated as follows.

The delivery network is represented by a weighted complete graph $G = (V, E, W)$. V is a node set. A node $v_i \in V$ $(i = 1, \cdots, n)$ represents a delivery location. $N = |V|$ is the number of nodes. $E \subseteq V \times V$ is a set of edges. An edge e_{ij} represents a route from v_i to v_j. W is an edge weight set. An edge weight d_{ij} represents a distance (or the costs to go) from v_i to v_j. Here, we presume $d_{ij} = d_{ji}$. The problem to find the minimal-length Hamilton path in such a graph $G = (V, E, W)$ is called Traveling Salesman Problem (TSP).

Delivery zones that are covered by one vehicle are different according to the region. Delivery locations are comparatively overcrowded in the urban area, whereas scattered in the rural area. Therefore, the number of locations for delivery differs - over several tens or hundreds up to 2000 or so - depending on the region and period of time.

It is necessary to compose and optimize a new delivery route for each round delivery since delivery locations change frequently. Though human or social factors should be considered, this is a problem to search the shortest path or route, modeled as a famous "Chinese Postman Problem" or "Traveling Salesman Problem (TSP)".

The computer support by nearly optimal solving methods is quite useful even though the method is an approximation algorithm. This reduces the burden and time loss of workers as well as costs and car exhaust gases in distribution networks.

III. Related Work

Simulated Annealing (SA) [15] and Tabu Search (TS) [4][6][7] are known as meta-heuristics search techniques.

Theoretically, SA is said to be able to find very near-optimal solutions (below 3% worst error) by decreasing the risk of falling into a local minimum [8]. But practically, it is very difficult to adjust SA's parameters such as cooling speed for coping with various location patterns. Furthermore, SA usually takes a long computation time to get nearly optimal solutions.

TS usually needs a long computation time to get practically optimal solutions [4]. Moreover, TS is said to be weak in maintaining solution diversity though it has a strong capability for local search.

However, such weaknesses of TS were improved by Kanazawa et al. [12]. So-called random restart methods [26], which apply local search such as 2-opt for improving random initial solutions, can obtain near-optimal solutions. Other methods to guarantee responsiveness by limiting the number of repetitions are the Greedy Randomized Adaptive Search Procedure (GRASP) [5] or the elaborated random restart method [16]. This elaborated random restart method is the basic method of MowGA [20].

However, these methods cannot guarantee the required accuracy level for over 200 cities TSPs though it obtained below 3% error solution for 40 cities TSP within 100 milliseconds.

As to using Genetic Algorithms (GA) to efficiently solve TSPs, various techniques are proposed. GA applications' solving methods using the edges assembly crossover (EAX) [18] and the distance-preserving crossover (DPX) [24] could get highly optimized solutions in case of very-large-scale TSPs (with 1000-10000 cities) [2][19].

These crossover methods examine the characteristics of parent's tour edge to strictly inherit it to children. However, since these crossover operations take a long computation time for analyzing edges, using it for large-scale TSP is often inefficient.

The Lin-Kernighan (LK) method [17] is a very famous heuristic search technique for TSPs. However, LK and its improved variants [25] are too complex methods though the optimality of obtained solutions is high. These methods are often incorporated with the meta-heuristics search such as SA, TA, and GA.

Especially, LKH [10] with a performance of the world's top class for approximation algorithm can solve a 1000 cities TSP within about 3 seconds and an about 2000 cities TSP within about 10 seconds . However, flexibility corresponding to peculiar condition is not favorable, since the algorithm is complex.

As to the approximate solution method, various techniques are proposed. In [2], two kinds of methods are compared for many cases. It shows that CGA-LK is advantageous to 300 - 10000 cities TSP, but Random-LK is advantageous to 198 - 268 cities TSP. Therefore, the solution that can efficiently solve TSP of 1000 cities or more can not necessarily efficiently solve TSP of about 100 cities.

As to TSP of lower scales (with 10s to 100 cities), in [2], the TSP lin105 benchmark test was solved with 1.77% average error rate in 231 seconds. In [3], the performance comparison experiments were conducted using various crossover operators.

A GA method aiming at obtaining high quality approximate solution as fast as possible for 10s - 100s cities TSPs is proposed by Van et al. [23].

In [20] and [21], a comparison experiment for the Mow-GA, the Miw-GA, the GA that used the best crossover in [3], the Random-LK, the GA by Van, and the TS by Kanazawa was conducted and the advantage of our proposed methods according speed and accuracy has been proven.

As to the exact algorithm for TSP, ABCC (Concorde) [1] is the world's fastest algorithm that needs 10s - 100s seconds to solve below 1000 cities TSPs.

All these approaches aim at constructing a TSP solution without considering former TSP solutions. In many applications, however, a new TSP does not differ a lot from a former one. In these cases, a Case Based Reasoning (CBR) [11] technology saves a lot of computational costs.

IV. The Proposed technique

The contribution of the present paper is a method to solve the TSP by CBR. CBR is a usual problem solving method in fields, were creating a new solution to a problem from scratch is expensive and solutions to similar problems are available: justice, medicine, architecture, and even bigger programming projects are partially performed in such a way.

Generally, CBR holds a case base (CB) with pairs [problem, solution] of formerly solved problems and consists of the steps (1) case retrieval, (2) case reuse, (3) case revision, and (4) case retaining.

In the case retrieval step, a case of the CB needs to be identified, which is "most similar" to the present case. The way to define "similarity" and to quantify it in a way that at "more similar or identical" can be decided (or better, quantified on a scale of values with an ordering relation in-between), depends on the kind of problem to be solved.

In the reuse step, the solution of the retrieved case needs to be adapted to the needs of the current problem.

The revision step aims at validating the adapted solution in the real world application and revising it according to the results of the real world application.

The retain step crucial. The question, whether or not a new case should be included as a new case into the CB, may be not easy to answer in many application fields. Also, adding new cases without considering the removal of (other) cases (1) "blows up" the CB and (2) bags the risk to keep old solutions that are outperformed by more recent ones.

In our particular application, we hold a CB with TSP problems along with their solutions sorted by their problem scale (number n of locations to visit).

A. Case Retrieval

If a new problem has to be solved, a "most similar" TSP needs to be retrieved from the CB. Here, we define similarity as the fraction of locations (cities) in the actual TSP, which are also in TSP of the CB. For example, if we have to solve a TSP with the scale 200 and 180 out of these 200 cities are in a TSP solution of the CB, its similarity is 180 / 200 = 0.9. If there are several solutions with the highest degree of similarity, the fittest one will be defined as "most similar".

However, a less similar TSP solution may be appropriate, too, in case its fitness is better than the fitness of the most similar one. There might be a TSP in the CB, which even covers all locations (cities) of the current TSC, i.e. with a similarity of 1, but the scale of this "very similar" TSP is 10 times bigger and thus, it is not a good candidate to derive a solution for the current TSP from it.

Therefore, we consider fitness, too. We retrieve the case with the next lower degree of similarity (respective the fittest one among them, if there are several ones) and look, whether this case has a higher fitness than the most similar case.

We define the fitness gain of a TSP solution s_1 with the fitness f_1 compared to a solution s_2 with a fitness f_2 as $(1 - f_2/f_1)$. If, for example, the most similar solution s1 has a fitness of $f_1 = 100$ (units of length for the complete distance of the tour) and the solution with the next better degree of similarity s_2 has a fitness of $f_2 = 80$, the fitness gain of s_2, related to solution s_1 is $(1 - 80/100) = 0.2$.

Also, we define the similarity loss of a TSP solution s_1 with a similarity sim_1 to the actual TSP, related to the solution

s_2 with the next lower degree of similarity to the actual TSP of sim_2 as $(sim_1 - sim_2)$. If, for example, the most similar solution s_1 has a similarity of 0.9 to the actual TSP and the solution with the 2nd highest similarity s_2 has a similarity of 0.8 to the actual TSP, the similarity loss is 0.1.

In our case retrieval strategy, we continue considering the next similar TSPs (to the currently considered one in the CB) as long as the fitness gain is higher than the similarity loss. When we come to a point, at which the next similar solution has a higher similarity loss then fitness gain (or even has a negative fitness gain), we refuse the next similar solution and retrieve the currently considered solution.

B. Case Reuse and Revision

After retrieving a solution from the CB, the redundant places (locations that are not in the current TSP) are removed from it.

After removal, the deficient locations (locations of the current TSP, which are not in the TSP retrieved from the CB) are added by the Nearest Insertion (NI) technique. This technique finds the position of a new location to insert, at which the increment of the complete tour length is minimal.

Since this reuse method already includes optimization issues by finding the optimal place for each inserted location (and a rejection of the non-optimal ones), there is no explicit revision of this result in our CBR application here.

C. Case Retaining and Case Base Maintenance

First, we determined a reasonable number of TSP solutions in the CB. By experiments with various TSP scales n we found that

- in CBs with less than $n/2$ TSP solutions there is not much hope to find a similar case,
- in CBs with around n TSP solutions we always found a similar TSP case in a reasonable time of $50-100ms$, which is negligible compared to the permitted solution time $3s$, and
- in CBs $2n$ cases and more the search time for a similar solution is not acceptable any more ($9s$, in some cases),

where n is the scale of the largest TSP to be solved.

There are two important CB maintenance issues, namely (1) it should contain the fittest individuals (shortest TSP tours), but also (2) diversity, i.e. individuals, which are not too similar to each other to ensure not to fall into local optima.

For ensuring both fitness and diversity, we hold the fitness (tour length) of each individual in the CB as well as a representation of the similarity of the current population in the CB.

To quantify both, we use a parameter, which is currently fixed by 5:

- If a new TSP solution is more than $1/5$ worse in terms of fitness than the CBs fittest element, it is accepted for CB inclusion only, if improves the CBs diversity.
- For checking the diversity, a pattern set is hold, which describes all $4/5$ (80%) similarities of the current population in the CB.

Technically, we represent the similarity (common 4/5 sub-routes of the TSP solutions in the CB) by a set of patterns, which describes the commonalities of a population by using

- non-variables (as lower case letters), which represent all 80% sub-routes that occur in the current CB) and
- variables (as upper case letters), which represent 20% sub-routes, which have to be exceeded by a new TSP solution to be accepted for inclusion into the CB.

Such a pattern set represents the similarities (sub-routes that some TSP solutions have in common) of a population, if each individual is covered by at least one pattern.

The shortest length of sub-routes a variable can stand for is, when related to (divided by) the complete length of the individuals, the diversity degree of the population.

To describe a population's similarities, which has to meet a certain degree of diversity, we create an according pattern set. Any newly created individual can be checked, whether or not it is covered by one of the patterns. If so, this individual is not appropriate to become a member of the population, otherwise it is.

For example, let's consider a 10 elements signature $\Sigma = \{a, b, c, d, e, f, g, h, i, j\}$ and a (to keep the example simple, small) population Pop of three individuals[3][4]: $Pop = \{abcjheidfga, acgdibhjfea, acgdhjfheba\}$. In case a diversity degree of more than 0.4, all sub-sequences with the length of $0.4 * length(gene)$, i.e. all 40% of the gene's length containing sub-sequences have to be replaced by variables.

Since our individuals have 10 elements, all 4-elemental parts of it are replaced by variables. Therefore, the pattern set Pat, that represents the 0.6 - similarity of this population is $Pat = \{aXeidfga, abXidfga, abcXdfga, abcjXfga, abcjhXga, abcjheXa, aXbhjfea, acXhjfea, acgXjfea, acgdXfea, acgdiXea, acgdibXa, aXjfheba, acXfheba, acgXheba, acgdXeba, acgdhXba, acgdhjXa\}$.

To avoid finding local optima only, the solution space should be well covered by a population, i.e. the individuals should be well distributed among the patterns.

From a diversity point of view, it is desirable to have a good coverage of the solution space by a population. In terms of our structural diversity representation it means, the individuals should be well distributed among the patterns, i.e. the number of individuals covered by a pattern is about the same for each pattern.

If, for example, a pattern covers 100 individuals and the remaining 20 patterns cover about 3 individuals each, the individuals of the population are not homogeneously distributed among the patterns and thus, not very diverse. Vice versa, in case it is about the same for each pattern, there is no

[3] Since our application of the TSP problem, the individuals contain all elements (cities to visit) of the signature, but in different sequences.

[4] For clearness, we use different colors for different individuals. The color of a pattern refers to the individual it is derived from.

concentration of many individuals for a particular pattern, i.e. in such a situation the request of being diverse is met much better.

A measure for being "well distributed" among the patterns is the entropy of the patterns based on the probability of a pattern to cover any individual of the population

$$H(Pat, Pop) = -\sum_{i=1}^{|Pat|} p_i * ld(p_i)$$

with p_i being the likelihood of a population's individual to be covered by the i-th pattern of the pattern set Pat, i.e. the number of individuals covered by the pattern related to the total number of individuals in the population.

The entropy is maximal, if all individuals are equally distributed among the patterns and becomes less, the more the distribution is unequal. This metric has the advantage that each individual counts not more than once, even in case it is covered by several patterns.

The issue, whether or not a new TSP solution improves the diversity in the CB is answered by using the pattern set:

1) In case it does not match with any of the patterns, it is accepted for inclusion into the CB, even if its fitness is more than $1/5$ worse than the fitness of the CBs fittest element and
2) in case it does match with any of the patterns it is accepted, if it improves the entropy of the distribution of the TSP solutions among the patterns, even if its fitness is more than $1/5$ worse than the fitness of the CBs fittest element and

In case of (1), the pattern set needs to be updated by adding those patterns for the new individual, which are not already in the pattern set.

V. Experiments and Results

A. Experiment

In this section, we compare the proposed method with our former one, namely BR-GA. The comparison experiments were done by solving two TSPLIB problems.

The experiments were conducted under the following computation environment. Namely, the CPU is an AMD Athlon 64 X2 3800+ 2GHz processor. It is almost the same performance as Athlon 64 3200+ 2GHz, because of its execution performance on the single core mode with 1 GB memory.

The programs were written in C language, compiled by Microsoft Visual C++ .NET 2003 version 7.1.3091 with the /02 option (directing the execution speed preference), and executed on Windows XP Professional. The experiments were conducted under the following computation environment. The CPU is Intel(R)Core(TM) i7-2600 CPU @3.40GHz processor with 4GB memory.

B. Results

Table I shows the error rates of our former approach BR-GA and the method proposed here for 2 instances of large scale TSP benchmarks, namely u1432+30 and rl1889+30. By

TABLE I. Experimental Results

TPS	worst error rate of 3 seconds runtime	
	BR-GA	proposed method
u1432+30	5.92	0.68
rl1889+30	10.26	0.88

error rate, we define the percentage of the TSP solution's total distance being longer that the optimal solution, which we also computed - of course with much more computation cost - for this evaluation purpose. These results are pretty convincing that deriving new TSP solutions from a good former solution of a similar TSP, is a good idea. The error rate is about 10 times better that it was with our former approach.

The search of the most similar case in the CB took between $50ms$ and $100ms$.

The performance regarding computational complexity (run time) is theoretically that of $n/5$ of BR-GA, since a maximum of 20% of a solution is created, the rest of the solution is just adopted from its "parent", a former TSP solution.

VI. Conclusion

In this paper, we introduced a Case Based approximation method to solve up to 2000 cities TSP problems in a required maximal response time of 3 seconds with a required maximum error rate of 3 %.

This method is based on the insight, that most solutions are very similar to solutions that have been created. Thus, in many cases a solution can be derived from former solutions by

1) selecting a most similar TSP from a library of former TSP solutions,
2) removing the locations that are not part of the current TSP and
3) adding the missing locations of the current TSP by mutation, namely Nearest Insertion (NI).

This way of creating solutions by Case Based Reasoning (CBR) avoids the computational costs to create new solutions from scratch.

The evaluation of this new method revealed remarkable results according the error rate of the derived solutions within a run time that is just 20 % of the time to create a new solution from scratch plus $50 - 100ms$ for searching the most similar case in the library of former TSP solutions.

As to future work, we aim at reaching the following objectives:

- First of all, we investigate the influence of the parameter for fitness and diversity requirements of a new solution to be accepted for inclusion into the CB (which is currently intuitively set to 5) fix it accordingly.
- For adapting a former solution toward a requested one, we extend the (quite simple) NI method towards a GA that uses NI (multiply, in parallel) and extend this GA later by also including edges assembly crossover (EAX) besides NI. This way, new TSP can adopt

useful properties (fitness, being divers from others in the population) from two parents and utilize the best of each parent. We have some hope, to stay within the required limit of computational costs by these extensions and thing about using parallel computation to still meet this requirement.

- Our maintenance technique for the CB is still quite simple. We think about extending it concept that ensures a well-balanced trade-off between fitness and diversity within the TSP solutions of the case base.
- Moreover, we investigate ways to extend the class of TSP problems towards practical requirements from field experts, who like to include one-way traffic (asymmetric TSPs), road conditions (instead of just considering the distance as fitness), and other issues to make the system more practicable.

ACKNOWLEDGMENT

This work was supported by KAKENHI (23500288) and Research Institute for Science and Technology of Tokyo Denki University (Q10J-04).

REFERENCES

[1] http://www.math.uwaterloo.ca/tsp/concorde/index.html, accessed March 18, 2014.

[2] R. Baragiia, J. I. Hidalgo, and R. Perego, "A hybrid heuristic for the traveling salesman problem", *IEEE Transactions on Evolutionary Computation*, vol. 5, issue 6, pp. 613-622, 2001.

[3] C. Cheng, W. Lee, and K. Wong, "A genetic algorithm-based clustering approach for database partitioning", *IEEE Transactions on Systems, Man and Cybernetics*, Part C, vol. 32, issue 3, pp. 215-230, 2002.

[4] Y. Fang, G. Liu, Y. He, and Y. Qiu, "Tabu search algorithm based on insertion method", *Proc. of the 2003 International Conference on Neural Networks and Signal Processing*, pp. 420-423, 2003.

[5] T. A. Feo, M. G. C. Recende, and S. H. Smith, "A Greedy Randomized Adaptive Search Procedure for Maximum Independent Set", *Operations Research*, vol. 42, number 5, pp. 860-878, 1994.

[6] F. Glover, "Tabu Search - Part 1", *ORSA Journal on Computing* 1 (2), pp. 190-206, 1989.

[7] F. Glover, "Tabu Search - Part 2", *ORSA Journal on Computing* 2 (1), pp. 4-32, 1990.

[8] H. Goto, Y. Hasegawa, and M. Tanaka, "Efficient Scheduling Focusing on the Duality of MPL Representatives," *Proc. of the 2007 IEEE Symp. Computational Intelligence in Scheduling (SCIS 07)*, IEEE Press, pp. 57-64, 2007.

[9] G. Gutin and A. P. Punnen (Eds.), *The Traveling Salesman Problem and its Variations*. Springer, NY, USA, 2007.

[10] K. Helsgaun, "General k-opt submoves for the Lin-Kernighan TSP heuristic," *Mathematical Programming Computation*, vol. 1, pp. 119-163, 2009.

[11] E. Huellermeier, *Case-Based Approximate Reasoning*. Springer, Berlin, 2007.

[12] T. Kanazawa, K. Yasuda, "Proximate Optimality Principle Based Tabu Search", *IEEJ Transactions on Electronics, Information and Systems*, vol. 124-C, number 3, pp. 912-920, 2004.

[13] R. M. Karp, "Probabilistic analysis of partitioning algorithms for the traveling-salesman problem in the plane", *Math. Oper. Res.* 2(3), pp. 209-224, 1977.

[14] H. Ken, I. Kokolo, S. Jun, O. Isao and K. Shigenobu, "Hybridization of Genetic Algorithm with Local Search in Multiobjective Function Optimization : Recommendation of GA then LS," *Transactions of the Japanese Society for Artificial Intelligence*, vol. 21, pp. 482-492, 2006.

[15] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, "Optimization by Simulated Annealing", *Science* 220(4598), pp. 671-680, 1983.

[16] S. Kubota, T. Onoyama, K. Onayagi, and S. Tsuruta, "Traveling Salesman Problem Solving Method fit for Interactive Repetitive Simulation of Large-scale Distribution Network", *Proc. IEEE SMC 99*, pp. 533-538, 1999.

[17] S. Lin and B. W. Kernighan, "An Effective Heuristic Algorithm for the Traveling-Salesman Problem". *Operations Research* 21(2), pp. 498-516, 1973.

[18] Y. Nagata and S. Kobayashi, "The Proposal and Evaluation of a Crossover for Traveling Salesman Problems: Edge Assembly Crossover", *Journal of Japan Society for AI*, vol. 14, number 5, pp. 848-859, 1999.

[19] H. D. Nguyen, I. Yoshihara, K. Yamamori, and M. Yasunaga, "Implementation of an Effective Hybrid GA for Large-Scale Traveling Salesman Problems", *IEEE Transactions on Systems, Man and Cybernetics*, part B, vol. 37, issue 1, pp. 92-99, 2007.

[20] Y. Sakurai, T. Onoyama, S. Kubota, Y. Nakamura, and S. Tsuruta, "A Multiworld Intelligent Genetic Algorithm to Interactively Optimize Large Scale TSP", *Proc. of the 2006 IEEE International Conference on Information Reuse and Integration (IEEE IRI2006)*, Hawaii, USA, pp. 248-255, 2006.

[21] Y. Sakurai, T. Onoyama, S. Kubota, and S. Tsuruta: "A Multi-inner-world Genetic Algorithm to Optimize Delivery Problem with Interactive-time", *Proc. of the 4th IEEE Conf. on Automation Science and Engineering (CASE 2008)*, Washington DC, USA, pp. 583-590, 2008.

[22] Y. Sakurai, K. Takada, N. Tsukamoto, T. Onoyama, and R. Knauf, "A Simple Optimization Method based on Backtrack and GA for Delivery Schedule", *Proc. of the IEEE Congress on Evolutionary Computation 2011 (CEC 2011)*, IEEE Catalog Number: CFP11ICE-CDR, ISBN: 978-1-4244-7833-0, pp. 2790-2797, 2011.

[23] X. Van, H. Liu, J. Van, and Q. Wu, "A Fast Evolutionary Algorithm for Traveling Salesman Problem", *Proc. of the Third International Conference on Natural Computation (ICNC 2007)*, vol. 4, pp. 85-90, 2007.

[24] D. Whiteley and T. Starkweather, "Scheduling Problem and Traveling Salesman: The Genetic Edge Recombination Operator", *Proc. of ICGA 89*, pp. 133-140, 1989.

[25] Y. Yamamoto and M. Kubo, *Invitation to the Traveling Salesman Problem*. Asakura Syoten, Tokyo, 1997.

[26] M. Yanagiura and T. Ibaraki, "On Metaheuristic Algorithms for Combinatorial Optimization Problems", *Transactions of the IEICE*, vol. J85-D-II, number 8, pp. 3-25, 2000.