A Case Based Approach for an Intelligent Route Optimization Technology

Masaki Suzuki, Takaaki Motomura, Taro Matsumaru and Setsuo Tsuruta
School of Information Environment
Tokyo Denki University
2-1200 Muzai Gakuendai
Inzai, Chiba, 270-1382, Japan
+81 476 46 8493
13jkm15@ms.dendai.ac.jp

Rainer Knauf
Faculty of Computer Science and Automation
Ilmenau University of Technology
PO Box 10 05 65
98684 Ilmenau, Germany
+49 3677 69 1445
rainer.knauf@tu-ilmenau.de

Yoshitaka Sakurai
School of Interdisciplinary Mathematical Sciences
Meiji University
4-21-1 Nakano Nakano-ku
Tokyo, 164-8525, JAPAN
+81 3-5343-8307
sakuraiy@meiji.ac.jp

ABSTRACT
The paper introduces a Case Based Approximation method to solve large scale Traveling Salesman Problems in a short time with a low error rate. It is useful for domains with most solutions being similar to solutions that have been created. Thus, a solution can be derived 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.

Categories and Subject Descriptors
D.2.6 [**Evolution Support Environment**]: Evolutionary Computation – *Genetic Algorithms, Case Based Reasoning, fitness, diversity*

General Terms
Algorithms

Keywords
Traveling Salesman Problems (TSP), Genetic Algorithm (GA), Heuristics, Case Based reasoning (CBR), Knowledge Maintenance

1. INTRODUCTION
The efficiency of product distribution in Japan is low compared to other industrialized countries. 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 transport expenses in Japan. Thus, it should be optimized.

A round delivery comprises more than several tens or hundreds of different locations. The optimization of a delivery route can be modeled as a large-scale of Traveling Salesman Problem (TSP).

GECCO'14, July 12–16, 2014, Vancouver, BC, Canada.

The TSP is a combinatorial problem that causes computational explosion due to the *n!* order of combinations for an *n*-city TSP. Thus, to practically obtain an efficient delivery route of such a distribution system, a near optimal solving method of TSP is indispensable. Moreover, the practical use of such a solving method on an actual site needs human confirmation (which is difficult to formulate) of the solution, since social and human conditions are involved. Human users should verify the practicability of a solution. Users sometimes need to adjust manually or select an alternative solution to meet miscellaneous technical and social side conditions. Therefore, the TSP solving methods are required to ensure a response time necessary for the above human interaction. Moreover, it is necessary to use comparatively simple strategies (heuristics) that users and human specialists can optimize the delivery schedule.

Interestingly, 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 the TSP [13][7][3] may generate even mathematically optimal solutions in particular cases but cannot ensure that the amount of errors is below 2-3%.

Simple strategies to create a delivery schedule are repeated methods such as stepwise inserting new delivery locations near the roughly imaged delivery route until the route includes all delivery locations, and modifying the traveling order of the delivery locations in order to decrease the total traveling distance. This is a method called “Nearest Insertion” (NI). Another strategy is repeating exchanges of combinations of routes in the delivery schedule. This is a method called “2-opt”.

Complex strategies such as the Lin-Kernighan (LK) [8] method or Karp’s partitioning [6] are not considered as practical methods from the above-mentioned viewpoint. This is because these methods are not familiar to field experts though theoretically their way of modification matched to deliverer's convenience.

In our former work, we proposed some types of Genetic Algorithms (GAs) [9]. These GA incorporated simple heuristics aiming at interactive real-time response as well as avoiding significant errors for any kinds of delivery location patterns.

A simple GA incorporating 2-opt type heuristics tends to fall into a local optima with certain delivery location patterns though it performs well for others. Therefore, we proposed a multi-outer-

world GA (Mow-GA) [10], which overcomes some 2-opt type GA's drawbacks by cascading NI type GA to 2-opt type GA.

However, the Mow-GA has also some defects. It starts NI type GA from the first generation without inheriting the information of the elite individuals generated in the 2-opt type GA. It cannot have enough performance for large scale problems within about 3 seconds. Moreover, the complementary effects caused by combination of such heuristics did not work [10].

As an improved version of the Mow-GA, Multi-inner-world Genetic Algorithm (Miw-GA) was proposed [11]. It improved the search efficiency through complementary effects that supplement the weak points mutually by processing two types of heuristics (2-opt and NI) in each generation.

To simplify the searching processes, the searching operator of our 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 TSP became more than 200, the average error rate from the optimal solution exceeded 4% in some cases.

Therefore, we proposed a so called Backtrack and Restart GA (BR-GA) [12], which maintains a population's diversity by conducting random restart and fostering new children. It achieves below 3% error rate for less than 1000 cities TSPs within 3 seconds.

Even the world's fastest exact algorithms called Concorde [1] needs 10- 100 seconds to solve the same size of TSPs. As to approximation algorithms, LKH [8][4] can solve below 1000 cities TSPs within about 3 seconds. This marks the world's top level accuracy and speed in solving TSPs by implementing an effective branch & cut in the search process.

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 domain engineers to understand and to modify a solution depending on field social conditions, and to practically use.

An exact solution or too much accuracy is not valuable since 2-3% error is not recognizable for field users. Moreover, complex methods have problems in flexibility. Application field engineers have difficulties in modifying the method to derive solutions suitable for the deliverer's convenience.

However, in practice, new TSPs distinguish from formerly solved ones very slightly. The changes in the locations to visit are less than 10 – 30 % and about 5 % in average.

Here, we introduce an alternative method to solve TSPs by Case Based Reasoning (CBR) [5]. In CBR, former solutions of a problem are collected in a case base (CB) and new problems are solved by (1) retrieving the most appropriate case from the CB, (2) mutating its solution towards a solution of the actual TSP, (3) validating or revising the solution, and (4), if applicable, adding the created solution to the CB.

This paper is organized as follows. In the next section, the delivery route optimization problem and its technical problems are described. Section three introduces our method for solving the problem. In section four, experiments to validate this method are shown and section six concludes the paper.

2. DELIVERY ROUTE OPTIMIZATION

A delivery network is represented by a weighted complete graph $G = (V, E, W)$. V is a node set. A node $v_i \in V$ ($i = 1, \ldots, n$) represents a location (address) for delivery. $N=|V|$ 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 "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.

3. PROPOSED TECHNIQUE

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, architecture, and even bigger programming projects are partially done in such a way.

CBR holds a 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. An appropriate ways to define "similarity" for our application and not losing the attention to fitness issues at the same time needs to be found.

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. In some application fields, there are good reasons to omit or just simulate this step, namely (1) in case the application in the real world bags risks according security or safety of people or other important goods, but also (2) in case the reuse technology already includes a check, whether or not a potential change can improve the solution.

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

3.1 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 same 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 actual 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 related to a solution s_2 with a fitness f_2 as $(1 - f_2/f_1)$. For example, if the most similar solution s_1 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 $(sim_1 - sim_2)$. For example, if 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.

3.2 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 actual 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, there is no explicit revision of this result in our CBR application here.

3.3 Case Retaining and CB Maintenance

First, we determined a reasonable number of TSP solutions in the CB. This number should be related to the scale n of the largest TSP to be solved. By some experiments we found that

(1) in CBs with less than n/2 TSP solutions there is not much hope to retrieve an appropriate case,

(2) in CBs with around n TSP solutions we always found an appropriate case in a reasonable time of 50 – 100 ms, which is negligible, compared to the permitted solution time 3 s, and

(3) in CBs 2n cases and more the search time for an appropriate case is not acceptable any more (9 s, in some cases).

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.

We are still in the process to collect cases for the CB. Therefore, we currently add all solutions created by our technique to the CB.

Later, we have to come up with a concept for CB maintenance, which ensures that the size of the CB (and thus, the computational costs for finding a most similar case in it) stays limited and the cases in the CB are "useful" in terms of fitness and diversity.

4. EXPERIMENTS AND RESULTS

We compared 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 ver. 7.1.3091 with the /02 option (directing the execution speed preference), and executed on Windows XP Professional.

Table 1 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 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.

Table 1. Experimental results

TSP	Worst error rate after 3 seconds runtime	
	BR-GA	proposed method
u1432+30	5.92	0.68
rl1889+30	10.26	0.88

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

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.

5. SUMMARY AND OUTLOOK

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 (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.

In our former work, we tried to keep diversity in terms of solutions and defined common subsequences of TSP solutions by a pattern set. We feel, for creating new solutions that are required to be different enough from a current population's majority, this is a good idea, but for case base maintenance the computational costs are very high. Therefore, we shifted from the idea of considering diversity of solutions towards the idea of promoting diversity of problems, for which a CB provides a solution. For the solutions, we now here consider fitness as a more important point then diversity.

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 50-100 ms for searching the most similar case in the library of former TSP solutions.

For future work, we aim at reaching the following objectives.

For adapting a former solution toward an actual one, we extend the simple NI method towards a multiple NI in parallel. After that, we extend it by also including edges assembly crossover (EAX). 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 hope to stay within the required limit of computational costs by these extensions and think about using parallel computation to still meet this requirement.

Our maintenance technique for the CB is still rudimentary, because we are still suffering from haven too few. 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 TPS 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.

6. ACKNOWLEDGMENTS

This work was sponsored by KAK ENHI (23500288) and the Research Institute for Science and Technology of Tokyo Denki University

7. REFERENCES

[1] http://www.math.uwaterloo.ca/tsp/concorde/index.html.

[2] *Proc. IEEE Symp. Computational Intelligence in Scheduling (SCIS 07)*, IEEE Press, 57-64.

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

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

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

[6] Karp, R. M., 1977. "robabilistic analysis of partitioning algorithms for the traveling-salesman problem in the plane. *Math. Oper. Res.*, 2, 3, 209–224.

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

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

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

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

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

[12] Sakurai, Y., Takada, K., Tsukamoto, N., Onoyama, T., Knauf, R. 2011. A Simple Optimization Method based on Backtrack and GA for Delivery Schedule. *Proc. of the IEEE Congress on Evolutionary Computation 2011 (CEC 2011)*, 2790-2797.

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