Journal of the Korean Institute of Industrial Engineers
Vol. 33, No. 4, pp. 469-479, December 2007.

A Voronoi Tabu Search Algorithm for the Capacitated Vehicle Routing Problem

Yong-Ju Kwon[1] · Jun-Gyu Kim[2] · Jeongyeon Seo[2] · Dong-Ho Lee[2†] · Deok-Soo Kim[2]

[1]PIC center, Nexerve Co., Ltd.
[2]Department of Industrial Engineering, Hanyang University, Seoul 133-791, Korea

차량경로 문제에 관한 보로노이 다이어그램 기반 타부서치 알고리듬

권용주[1] · 김준규[2] · 서정연[2] · 이동호[2] · 김덕수[2]

[1](주)넥서브 업무혁신센터 / [2]한양대학교 산업공학과

This paper focuses on the capacitated vehicle routing problem that determines the routes of vehicles in such a way that each customer must be visited exactly once by one vehicle starting and terminating at the depot while the vehicle capacity and the travel time constraints must be satisfied. The objective is to minimize the total traveling cost. Due to the complexity of the problem, we suggest a tabu search algorithm that combines the features of the existing search heuristics. In particular, our algorithm incorporates the neighborhood reduction method using the proximity information of the Voronoi diagram corresponding to each problem instance. To show the performance of the Voronoi tabu search algorithm suggested in this paper, computational experiments are done on the benchmark problems and the test results are reported.

Keywords: Capacitated Vehicle Routing; Tabu Search; Voronoi Diagram

1. Introduction

Logistics, generally defined as the provision of goods and services from supply points to demand points, includes supply of raw materials to manufacturers, transfer of products to warehouses or depots, delivery of products to customers, and collection of reusable and repairable products. Among various decision issues in logistics systems, we focus on vehicle routing, which is the problem of designing a set of routes for a fleet of vehicles. There are various types of vehicle routing problems according to the forms of vehicles (fleet size and capacity, type), customer information (demand size, demand type, and demand frequency), side constraints (time windows, the number of depots, and backhaul), etc.

Among various vehicle routing problems, this paper focuses on the capacitated vehicle routing problem (CVRP) that determines a set of routes by a fleet of vehicles, starting and terminating at the depot, to serve a given set of customers under the vehicle capacity and the travel time restrictions. From the theoretical viewpoint, the CVRP is an extension of the traveling salesman problem

This work was supported by Brain Korea 21 Grant funded by Korean Government. This is gratefully acknowledged.
† Corresponding author : Dong-Ho Lee, Department of Industrial Engineering, Hanyang University, Seongdong-gu, Seoul 133-791, Korea
Tel : +82-2-2220-0475, Fax : +82-2-2296-0471, E-mail : leman@hanyang.ac.kr
Received April 2007; revision received July 2007; accepted August 2007.

(TSP). This implies that the CVRP is NP-hard (Lenstra and Rinnooy Kan 1981). On the other hand, from the practical viewpoint, the CVRP can be found commonly in the real fields. For example, a distribution company in Turkey transporting electronic household commodities from plants to dealers (Barbarosoglu and Ozgur 1999) and a distribution company in Greece transporting fresh milk in the Athens area (Tarantilis and Kiranoudis 2002). Also, see Lee and Lee (2005), Seong and Moon (2006), Oh *et al.* (2006) for recent studies on the CVRP and its variations in Korea.

The CVRP was first introduced by Dantzig and Ramser (1959). Since then, the CVRP and its variants have been studied extensively. In the 1960s, the CVRP was formulated as an integer programming model. In the 1970s, the CVRP research focused on the two-phase heuristics, i.e., route building and improvement. The most widely well-known route-building heuristic is the saving algorithm by Clarke and Wright (1964) that considers the saving cost if routes are merged, and the well-known two-phase heuristic is the sweep algorithm by Gillett and Miller (1974) that suggested the first clustering and second routing method. At this time, 2-opt and 3-opt algorithms, proposed by Lin and Kernighan (1973), were applied to the CVRP for route improvement. In the 1980s, mathematical programming-based algorithms and interactive heuristics were proposed. These heuristics required more computational effort and time, but gave very high-quality solutions. In the 1990s, the research focus shifted to applying meta-heuristics such as simulated annealing, deterministic annealing, neural network, ant colony, genetic algorithm and tabu search. Some of these algorithms produced highly accurate solutions for the benchmark problems. Osman (1993) suggested the first-best-admissible strategy and the best-admissible strategy for the tabu search algorithm with λ-interchange mechanism. Furthermore, he showed that tabu search is better than simulated annealing for the vehicle routing problem. Gendreau *et al.* (1994) suggested the taburoute algorithm considering infeasible solutions using the genius algorithm proposed by Gendreau *et al.* (1992). Rochat and Taillard (1995) developed a tabu search algorithm using the concept of adaptive memory, and Toth and Vigo (2003) suggested the granular tabu search (GTS) that reduces the solution space effectively. See Laporte (1992) and Laporte *et al.* (2000) for literature reviews on the CVRP.

In this paper, we suggest a new tabu search algorithm that combines the features of the existing search heuristics. In particular, the new tabu search algorithm incorporates the method to reduce the number of neighborhood solutions using the proximity information of the Voronoi diagram corresponding to each problem instance. Here, the Voronoi diagram, which will be explained later, is a powerful tool in computational geometry which provides all spatial information among geometric objects in a system with an efficient data structure (Kim *et al.* 2001a, b). To show the performance of the tabu search algorithm using the Voronoi diagram, computational experiments were done on the classic and the large-scale benchmark problems, and the test results show that our algorithm is competitive to the existing search heuristics, especially for the classic benchmark problems.

This paper is organized as follows. In the next section, the CVRP considered here is described in more detail, and an overview of the Voronoi diagram is presented in Section 3. The new tabu search algorithm is explained in Section 4, and the results of computational experiments are reported in Section 5. Finally, Section 6 concludes the paper with a short summary and discussions on possible extensions.

2. Problem Description

The CVRP considered here is the problem of determining a set of routes by a fleet of vehicles to serve a given set of customers under the vehicle capacity and the route duration restrictions. The objective is to minimize the total traveling cost. More formally, the CVRP may be described as follows. Let $G = (V, E)$ be a complete graph, where $V = \{0, 1, 2, \cdots, n\}$ is the vertex set and $E = \{(i, j) : i \neq j\}$ is the edge set. The vertex 0 denotes the depot, whereas the other vertices $V \backslash \{0\}$ correspond to the customers. Customer i is associated a demand q_i and requires a service time δ_i. A non-negative cost c_{ij}, denoting the distance or travel time between customers i and j, is associated with each edge $(i, j) \in E$. It is assumed that the costs are symmetric, i.e., $c_{ij} = c_{ji}$ for all i, j, and satisfy the triangle inequality, i.e., $c_{ik} + c_{kj} > c_{ij}$ for all $i, j, k \in V$. A set of m identical vehicles, each with capacity Q, is available at the depot. To ensure the feasibility of the problem, we assume that $q_i \leq Q$ for all $i \in V \backslash \{0\}$. Also, each vehicle may travel at most one route, and the number of vehicles must not be smaller than m_{min}, where m_{min} is the minimum number of vehicles needed to serve the total demand of customers.

A solution for the CVRP can be represented as a set of m routes $R_1, R_2, \cdots, R_m$, and the problem is to determine the vehicle routes for the objective of minimizing the total traveling costs. Here, the rth route R_r can be represented as $R_r = (0, i_{r1}, i_{r2}, \cdots, 0)$, where i_{rk} denotes the index for the kth customer in route r. Besides the decision variable, the CVRP has the following constraints.

(a) All vehicles start and end at the depot.
(b) Every vertex of $V \backslash \{0\}$is visited exactly once by exactly one vehicle.
(c) Each customer has a known demand that must be satisfied.
(d) The total demand of any route should not exceed the vehicle capacity Q.
(e) The total length (or sum of service and travel times) of any route should not exceed L, where L is a preset route duration.

3. Voronoi Diagram

Before presenting the solution algorithm suggested in this paper, this section presents the basic concept of the Voronoi diagram and the method to reduce the neighborhood solutions using the Voronoi diagram corresponding to each problem instance. We start with the Voronoi diagram structure that gives topological information among nodes in order to reduce the search space and hence to get good solutions quickly. Since a Voronoi diagram provides the most compact and concise representation of the proximity information in Euclidean space, we adopt this structure in our algorithm.

The Voronoi diagram adopted here is defined on the two-dimensional Euclidean space. Consider a finite set of n points on the two-dimensional space. The n points are denoted by $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$, with the Cartesian coordinates $(x_{11}, x_{12}), (x_{21}, x_{22}), \cdots, (x_{n1}, x_{n2})$. Note that the n points are distinct in the sense that $\mathbf{x}_i \neq \mathbf{x}_j$ for $i \neq j$, $i, j \in \{1, 2, \cdots, n\}$. Then, the planar ordinary Voronoi diagram can be defined as follows. (See Okabe *et al.* (1992) for more details on the Voronoi diagram.) In the definition, $\| \mathbf{x}_i - \mathbf{x}_j \|$ denotes the Euclidean distance between $\mathbf{x}_i$ and $\mathbf{x}_j$.

Definition. (Planar ordinary Voronoi diagram) Let $X = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \subset \mathbf{R}^2$, where $2 < n < \infty$ and $\mathbf{x}_i \neq \mathbf{x}_j$ for $i \neq j$, $i, j \in \{1, 2, \cdots, n\}$. We call the region given by

$$\mathrm{V}(\mathbf{x}_i) = \{\mathbf{x} \mid \| \mathbf{x} - \mathbf{x}_i \| \leq \| \mathbf{x} - \mathbf{x}_j \| \text{ for all } j \neq i, j \in \{1, 2, \cdots, n\}\}$$

the *planar ordinary Voronoi polygon* associated with $\mathbf{x}_i$ (or the Voronoi polygon of $\mathbf{x}_i$), and the set given by

$$\mathrm{V} = \{\mathrm{V}(\mathbf{x}_1), \mathrm{V}(\mathbf{x}_2), \cdots, \mathrm{V}(\mathbf{x}_n)\}$$

the *planar ordinary Voronoi diagram* generated by X (or the Voronoi diagram of X).

To illustrate the above definition more clearly, we generated the Voronoi diagram for a classic benchmark problem with 100 vertices using the algorithm suggested by Kim *et al.* (2001a, b), and it is shown in <Figure 1>. In this figure, we can see that the Voronoi diagram is the set of points that have the same Euclidean distance among the corresponding points.

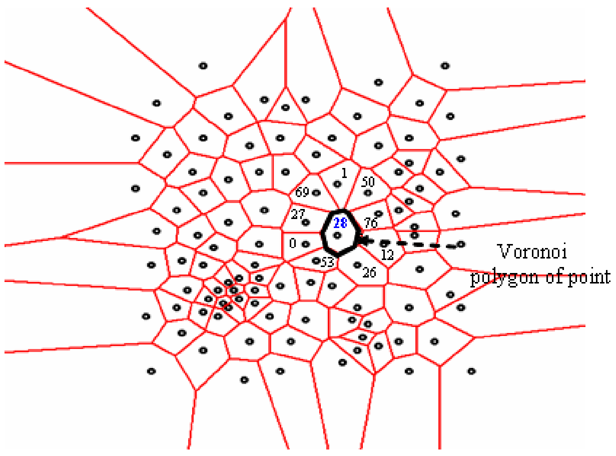

Figure 1. Voronoi diagram: and example

A number of useful information can be obtained from the Voronoi diagram. Among them, in this paper, we use the proximity information to reduce the neighborhood solutions in the tabu search algorithm suggested in this paper. Here, the proximity information for a point implies the set of points associated with those adjacent to the Voronoi polygon including that point. For example, in <Figure 1>, the proximity information for point 28 can be represented as $P_{28} = \{0, 1, 12, 26, 27, 50, 53, 69, 76\}$. Note that the proximity information implies that the points in P_i are relatively closer to point i than the others.

As stated earlier, the neighborhood generation method suggested in this paper uses the proximity information for each point. In other words, the complete graph $G = (V, E)$ for the CVRP is reduced to $G' = (V, E')$, where G' is a sub-graph of G in the sense that E' is the set of edges obtained from reducing the original edge set E using the proximity information of the corresponding Voronoi diagram for the vertex set V.

To show the effect of the reduction based on the proximity information, we compare the number of edges of the original graph and the reduced sub-graph, i.e., $|E|$ and $|E'|$, for each of the fourteen classic benchmark problems of Christofides *et al*. (1979) and the twenty large-scale benchmark problems of Golden *et al*. (1998) and the results are summarized in <Table 1>. For example, in the CMT1, the original complete graph has 2450 edges while the reduced sub-graph has 135 edges, which shows 94.5% reduction in the number of edges. Moreover, this table provides some information that even though the size of problems is larger, the number of edges to E' slightly increases. In overall, it can be seen from that table that the percentage of reduction is quite high and hence it can be used to reduce the neighborhood solutions while applying the tabu search algorithm. However, the reduction method does not guarantee that the reduced edge set always contains the optimal solutions.

Table 1. Comparison of the numbers of edges: original and reduced graphs

Classic benchmark problems					Large-scale benchmark problems				
Problem ID	$\|V\|^1$	$\|E\|^2$	$\|E'\|^3$	% reduction	Problem ID	$\|V\|$	$\|E\|$	$\|E'\|$	% reduction
CMT1	50	2450	135	94.5	G1	240	57360	640	98.9
CMT2	75	5550	209	96.2	G2	320	102080	880	99.1
CMT3	100	9900	284	97.1	G3	400	159600	1120	99.3
CMT4	150	22350	424	98.1	G4	480	229920	1360	99.4
CMT5	199	39402	582	98.5	G5	200	39800	560	98.6
CMT6	50	2450	135	94.5	G6	280	78120	1348	98.3
CMT7	75	5550	209	96.2	G7	360	129240	1008	99.2
CMT8	100	9900	284	97.1	G8	440	193160	1232	99.4
CMT9	150	22350	424	98.1	G9	255	64770	702	98.9
CMT10	199	39402	582	98.5	G10	483	232806	898	99.6
CMT11	120	14280	335	97.7	G11	252	63252	1118	98.2
CMT12	100	9900	272	97.3	G12	483	232806	1362	99.4
CMT13	120	14280	335	97.7	G13	252	63252	688	98.9
CMT14	100	9900	272	97.3	G14	320	102080	884	99.1
					G15	396	156420	1104	99.3
					G16	480	229920	1348	99.4
					G17	240	57360	702	98.8
					G18	300	89700	882	99.0
					G19	360	129240	1062	99.2
					G20	420	175980	1242	99.3

1 number of vertices
2 number of edges
3 number of edges reduced using the proximity information

4. Voronoi Tabu Search Algorithm

The algorithm suggested in this paper, called the Voronoi tabu search (VTS) hereafter, consists of two phases: obtaining an initial solution and improvement. The improvement phase incorporates the method to reduce the number of edges using the proximity information of the Voronoi diagram when generating the neighborhood solutions. Each of the two phases is explained below.

4.1 Obtaining an Initial Solution

The initial solution is obtained with the savings algorithm of Clarke and Wright (1964). The savings algorithm starts with n routes that correspond to each of the n vertices. Then, the saving for each pair of vertices is computed and the pairs are sorted in the non-increasing order of the savings. Here, the saving between vertices i and j is defined as

$$s_{ij} = c_{0i} + c_{j0} - c_{ij},$$

where c_{ij} denotes the cost (distance or travel time) between vertices i and j. Finally, the vertices are merged according to this order until no further merges are possible while considering the vehicle capacity and the travel time restrictions.

4.2 Improvement

In this phase, the initial solution is improved by the ordinary tabu search procedure, together with the features of the existing search heuristics and the neighborhood reduction method using the Voronoi diagram. Note that the tabu search is selected in this study since it works better than other search heuristics (Cordeau and Laporte 2002).

The tabu search heuristic, proposed by Glover (1989, 1990), is a well-known search technique to escape from terminating at prematured local optimum. The tabu search heuristic starts with an initial solution. For each alternative S for vehicle routes, a new alternative S' is obtained with a function that transforms S into S'. This transformation is called a *move* in the tabu search literature, which can be made to the neighboring solution even though it is worse than the given solution. To avoid cycling, the tabu search heuristic defines a set of moves that are tabu (forbidden), and these moves are sorted in a set Λ, called the *tabu list*. Elements of Λ define all tabu moves that cannot be applied to the current solution. The size of Λ is bounded by a parameter θ called the *tabu list size*. If $|\Lambda| = \theta$ before adding a move to Λ, one must remove an element in it, generally the oldest one. Note that a tabu move can be always allowed to be chosen if it creates a solution better than the incumbent solution, i.e., the best objective value obtained so far. This is called the *aspiration criterion* in the literature.

An application of tabu search can be characterized by: (a) representing solutions; (b) generating neighborhood solutions, and (c) termination condition(s). In the following, we explain the details of these in the implementation of the tabu search heuristic suggested in this paper.

4.2.1 Representing solutions

A solution of the CVRP is encoded by multiple strings, each of which represents an ordered set of vertices assigned to each vehicle. More formally, the solution is a set S of m routes R_1, $R_2,\cdots$, R_m, where $R_r = (0, i_{r1}, i_{r2}, \cdots, 0)$. As defined earlier, 0 and i_{rk} denote the depot and the index for the kth customer in route r, respectively.

4.2.2 Generating neighborhood solutions

To generate the neighborhood solutions, we use the λ-interchange method proposed by Osman and Christofides (1989). In this method, the neighborhood solutions are generated by changing up to λ customers between two routes R_a and R_b. In other words, each change between the two routes is described using a couple (λ_1, λ_2) (with $\lambda_1 \leq \lambda$ and $\lambda_2 \leq \lambda$), where λ_1 vertices are moved from R_a to R_b, and λ_2 vertices are moved from R_b to R_a. Since the value of Λ is often restricted to 1 or 2 in order to limit the number of possible neighborhoods, we consider the cases with $\lambda = 1$, which results in two types of moves, the swap with (1, 1) and the insertions with (1, 0) or (0, 1). In this study, all pairs of vehicle routes are considered for the 1-interchange method.

Although the value of λ is restricted to 1, the number of neighborhoods to be searched is still large. Therefore, we use the method to reduce the neighborhoods. As explained earlier, the reduction method is based on the proximity information of the Voronoi diagram. Consider an example with two routes, R_1 = (0, 10, 5, 3, 2, 4, 6, 0) and R_2 = (0, 8, 7, 1, 9, 12, 11, 0). Suppose that vertex 1 (in R_2) with the proximity information of P_1 = {0, 3, 5} is to be inserted into R_1. Then, we can see that there exist seven possible positions to which vertex 1 can be inserted. Among them, we consider those in the proximity information P_1. See <Figure 2(a)> for its pictorial description. (In the figure, the dashed arrows denote the possible positions for insertions considering the proximity information.) Like the insertion method, the swap method also reduces the number of neighborhood solutions using the proximity information. Consider the above example with R_1 and R_2. Suppose that vertex 1 (in R_2) has the proximity information of P_1 = {0, 3, 5} and vertex 5 (in P_1) has P_5 = {0, 1, 9}. Then, vertices 1 and 5 can be swapped since P_1 and P_5 have 5 and 1, respectively. See <Figure 2(b)> for its pictorial description.

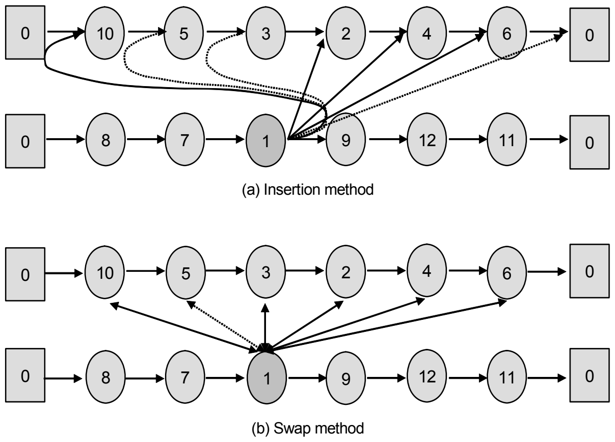

Figure 2. Generating neighborhood solutions : examples

While the (reduced) neighborhoods are generated using the 1-interchange method, we allow the infeasible solutions with respect to the vehicle capacity and the travel length (or time) restriction in order to extend the search space. To do this, we handle the infeasible solutions through the penalized objective function suggested by Gendreau *et al.* (1994). The mathematical description of the penalized objective function is given below. In this description, $[x]^+ = \max(0, x)$ and $x_{ij} = 1$ if a vehicle route contains the edge (i, j), and 0 otherwise.

$$F(S) = \sum_{i \in V}\sum_{j \in V} c_{ij}x_{ij} + \alpha_C\left[\left(\sum_{i=1}^{n} q_i \sum_{j=0}^{n} x_{ij}\right) - Q\right]^+$$

$$+\alpha_D\left[\left(\sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{i=1}^{n}\delta_i\sum_{j=0}^{n}x_{ij}\right) - L\right]^+,$$

where α_C and α_D are positive parameters representing the penalties associated with the violations of the vehicle capacity and the travel length constraints, respectively. The penalty parameters are updated periodically, i.e., every t iterations. More specifically, α_C and α_D are multiplied by 2 if a feasible solution can be found during t iterations, and divided by 2, otherwise. In our implementation, α_C and α_D were initially set to 100, and t to 10 from a preliminary test.

There may be several ways of selecting a move. Among them, this paper uses the method of examining the reduced neighborhood solutions (for the 1-interchange method) and taking the best move that is not tabu since it generally performs well. After the best move is done, the current solution is additionally improved by applying the 3-opt procedure.

4.2.3 Defining tabu moves

The tabu moves are defined as follows. If vertex i on route R_p is moved to route R_q at iteration t, the move of i from R_p to R_q is declared tabu during $t+\theta$ where θ is the tabu list size. Here, the tabu list size θ is determined using the *simple dynamic tabu term rule* of Glover and Laguna (1993). That is, θ is set to an integer uniformly distributed over the interval $[\theta_{min}, \theta_{max}]$. As in Gendreau *et al.* (1994), the parameters θ_{min} and θ_{max} for our VTS algorithm were fixed to 5 and 10, respectively. Note that the oldest tabu move is removed before adding a new one if the tabu list is full. As stated earlier, a tabu move can be always allowed to be chosen if it creates a solution better than the incumbent solution, i.e., the best objective value obtained so far.

4.2.4 Stopping condition

The improvement phase is terminated if no improvements have been made for a certain number t_1 of iterations. In our implementation, the t_1 was set to $50 \cdot n$ from a preliminary test, where n is the number of vertices.

4.3 Intensification

After a solution is obtained using the improvement method explained earlier, it can be improved once more in the intensification phase. The intensification method is the same as that of the above improvement method except that the neighborhood reduction method using the proximity information of the Voronoi diagram is not used when generating the neighborhoods. That is, the entire neighborhood solutions (for the 1-interchange method) are examined and the best move that is not tabu is taken. Finally, as in the improvement phase, the intensification phase is terminated if there are no improvements for a certain number t_2 of iterations. In our implementation, the t_2 ($< t_1$) was set to $10 \cdot n$ from a preliminary test.

5. Computational Experiments

To show the performance of the VTS algorithm, computational tests were done on the benchmark problems, the fourteen classic benchmark problems of Christofides *et al.* (1979) and the twenty large-scale benchmark problems of Golden *et al.* (1998). The classic benchmark problems contain between 50 and 199 vertices in addition to the depot. Here, problems 1 to 5 and 11 to 12 have the vehicle capacity restriction, while problems 6 to 10 and 13 to 14 have the vehicle capacity and the travel distance (time) restrictions. On the other hand, the large-scale benchmark problems contain between 200 and 483 vertices in addition to the depot. Here, problems 1 to 8, generated in concentric circles around the depot, have the vehicle capacity and the travel distance (time) restrictions, while problems 9 to 12, generated in concentric squares with the depot located in one corner, have the vehicle capacity restriction. Also, problems 13 to 20 were generated in concentric squares with the depot located in center.

The VTS algorithm was coded in C, and the test was done on a workstation with an Intel Xeon processor operating at 3.20 GHz clock speed. The performance measures used are: (a) gap from the best known solution value and (b) CPU seconds. Here, the gap can be represented as

$$100 \cdot (Z_{\text{VTS}} - Z^*) / Z^*,$$

where Z_{VTS} is the objective value obtained using the VTS algorithm and Z^* is the best known solution value.

Table 2. Test results on the classic benchmark problems

Problems	Number of nodes	Without intensification phase				With intensification phase				Best known
		Value	Gap (%)	CPU to best	CPU to end	Value	Gap (%)	CPU to best	CPU to end	
CMT 1	50	524.61	0.00	0.6	2.5	524.61	0.00	0.6	4.4	524.61
CMT 2	75	844.45	1.10	7.3	10.2	844.45	1.10	7.3	11.3	835.26
CMT 3	100	828.74	0.31	16.8	25.2	828.74	0.31	16.8	35.9	826.14
CMT 4	150	1042.80	1.40	15.5	37.8	1040.22	1.15	61.1	86.8	1028.42
CMT 5	199	1324.01	2.53	179.0	212.3	1324.01	2.53	179.0	257.2	1291.29
CMT 6	50	555.43	0.00	0.3	2.6	555.43	0.00	0.3	5.2	555.43
CMT 7	75	918.60	0.98	3.4	6.3	918.60	0.98	3.4	8.8	909.68
CMT 8	100	875.33	1.08	19.1	28.9	875.33	1.08	19.1	43.4	865.94
CMT 9	150	1177.14	1.25	61.6	83.7	1177.14	1.25	61.6	115.2	1162.55
CMT1 0	199	1441.88	3.30	132.5	165.9	1431.39	2.55	198.7	218.0	1395.85
CMT 11	120	1043.90	0.17	26.4	47.8	1043.90	0.17	26.4	101.6	1042.11
CMT 12	100	819.56	0.00	0.3	8.2	819.56	0.00	0.3	16.5	819.56
CMT 13	120	1567.26	1.69	28.1	42.9	1566.36	1.64	63.2	83.5	1541.14
CMT 14	100	866.37	0.00	2.5	9.7	866.37	0.00	2.5	23.5	866.37

Table 3. Test results on the large-scale benchmark problems

Problems	Number of nodes	Without intensification phase				With intensification phase				Best known
		Value	Gap (%)	CPU to best	CPU to end	Value	Gap (%)	CPU to best	CPU to end	
G 1	240	5813.77	3.31	388.3	495.2	5812.30	3.28	803.1	1115.7	5627.54
G 2	320	9085.65	7.55	463.5	670.8	9072.35	7.39	1766.2	2528.8	8447.92
G 3	400	12087.60	9.53	845.3	1253.4	12080.70	9.46	1281.6	3072.3	11036.22
G 4	480	15977.10	17.27	1317.6	1878.7	15847.00	16.31	8517.8	11194.0	13624.52
G 5	200	6752.91	4.52	93.2	179.9	6752.91	4.52	93.2	770.2	6460.98
G 6	280	9014.94	7.16	656.7	863.2	8992.02	6.88	1029.0	1935.6	8412.80
G 7	360	11472.40	12.68	735.2	1030.9	11459.90	12.55	1044.3	2243.4	10181.75
G 8	440	13039.80	11.80	2405.6	2820.6	13034.50	11.75	4314.2	6154.8	11663.55
G 9	255	590.41	1.20	216.8	315.2	589.60	1.06	317.1	558.5	583.39
G 10	323	755.33	1.86	577.3	737.1	749.78	1.11	1159.7	1577.1	741.56
G 11	399	970.99	5.72	1585.3	1882.8	961.64	4.70	5420.6	6409.8	918.45
G 12	483	1173.37	5.98	184.4	430.6	1173.37	5.98	184.4	1086.1	1107.19
G 13	252	904.86	5.33	93.5	146.8	904.86	5.33	93.5	251.5	859.11
G 14	320	1147.13	6.09	140.3	291.3	1143.47	5.75	464.3	689.0	1081.31
G 15	396	1448.23	7.66	155.1	287.7	1439.71	7.02	293.8	641.0	1345.23
G 16	480	1709.84	5.37	136.5	638.4	1706.25	5.15	644.3	1632.2	1622.69
G 17	240	717.59	1.38	165.2	223.3	717.26	1.34	271.9	354.1	707.79
G 18	300	1025.26	2.78	240.7	308.4	1023.64	2.62	477.3	565.3	997.52
G 19	360	1405.09	2.80	729.6	873.1	1403.39	2.67	913.6	1102.8	1366.86
G 20	420	1879.95	3.29	1721.3	1907.2	1879.44	3.26	1907.6	2107.1	1820.09

Table 4. Comparison of the results: classic benchmark problems

Problem	VST				Osman1		TABUROUTE2		GTS3		Best known
	Without intensification		With intensification								
	Value	CPU	Value	CPU	Value	CPU	Value	CPU	Value	CPU	
CMT1	524.61	2.47	524.61	4.38	524.61	67.2	524.61	360	524.61	48.6	524.61
CMT2	844.45	10.17	844.45	11.31	844.00	70.8	835.32	3228	838.60	132.6	835.26
CMT3	828.74	25.24	828.74	35.86	835.00	675	826.14	1104	828.56	143.4	826.14
CMT4	1042.80	37.75	1040.22	86.80	1044.35	3075	1031.07	2528	1033.21	270.6	1028.42
CMT5	1324.01	212.25	1324.01	257.20	1334.35	1972.7	1311.35	5454	1318.25	450	1291.29
CMT6	555.43	2.61	555.43	5.20	555.43	140.2	555.43	810	555.43	51.6	555.43
CMT7	918.60	6.25	918.60	8.78	911.00	203	909.68	3276	920.72	165	909.68
CMT8	875.33	28.91	875.33	43.44	866.75	1200	865.94	1536	869.48	174	865.94
CMT9	1177.14	83.70	1177.14	115.20	1184.00	2443.6	1162.89	4260	1173.12	340.2	1162.55
CMT10	1441.88	165.92	1431.39	217.99	1417.85	3310.1	1404.75	5988	1435.74	546.6	1395.85
CMT11	1043.90	47.83	1043.90	101.56	1042.11	1398.4	1042.11	1332	1042.87	190.8	1042.11
CMT12	819.56	8.16	819.56	16.50	819.59	407.5	819.56	960	819.56	66	819.56
CMT13	1567.26	42.94	1566.36	83.47	1547.00	1343	1545.93	3552	1545.51	560.4	1541.14
CMT14	866.37	9.74	866.37	23.53	866.37	5579	866.37	3942	866.37	84.6	866.37
Gap(%)	1.21		1.11		0.94		0.27		0.79		
CPU(s)	48.85		72.23		1563.25		2737.86		230.31		

1 tabu search algorithm of Osman (1993) (results on VAX 8600 computer)

2 TABUROUTE algorithm of Gendreau *et al.* (1994) (results on Silicon Graphics workstation, 36 MHz, 5.7 Mflops)

3 GTS (granular tabu search) algorithm of Toth and Vigo (2003) (results on Pentium 200 MHz PC)

Test results on the fourteen classic benchmark problems are summarized in <Table 2>. It can be seen from the table that the gaps of the VTS algorithm without (with) the intensification phase range from 0 (0) to 3.30% (2.55%), and the overall average of the gaps is 1.11%. In particular, the VTS algorithm found the best solutions for CMT 1, 6, 12 and 14. However, the gaps get large as the problem size increases. Also, the VTS algorithm gave the solutions within a reasonable amount of computation time. This implies that the neighborhoods are effectively reduced by the proximity information of the Voronoi diagram. Also, <Table 3> shows test results on the large-scale benchmark problems. In this table, we can see that the gaps are relatively large since the VTS algorithm considers a relatively small solution space. Compared with the existing algorithms, however, the CPU seconds were reasonable.

<Table 4> summarizes the results on the comparison between the VTS algorithm and the best existing algorithms, the tabu search algorithm of Osman (1993), the TABUROUTE algorithm of Gendreau *et al.* (1994), and the GTS (granular tabu search) algorithm of Toth and Vigo (2003), for the fourteen classic benchmark problems. It can be seen from the table that the VTS algorithm is competitive to the existing algorithms in that the differences in the overall average percentage gap is less than 1% and the CPU second is much less than those of the existing algorithms (although the computing machines were different.). Similar results can be found in <Table 5> (between the VTS algorithm and the GTS algorithm) for the twenty large-scale benchmark problems. However, the differences in the overall average percentage gap get larger.

Table 5. Comparison of the results: large-scale benchmark problems

Problem	VTS				GTS		Best known
	Improvement		Intensification				
	value	CPU	value	CPU	value	CPU	
Golden 1	5813.77	495.23	5812.30	1115.69	5736.15	298.8	5627.54
Golden 2	9085.65	670.77	9072.35	2528.83	8553.03	496.8	8447.92
Golden 3	12087.60	1253.42	12080.70	3072.31	11402.75	776.4	11036.22
Golden 4	15977.10	1878.70	15847.00	11194.00	14910.62	907.8	13624.52
Golden 5	6752.91	179.92	6752.91	770.22	6697.53	142.8	6460.98
Golden 6	9014.94	863.22	8992.02	1935.56	8963.32	279.0	8412.80
Golden 7	11472.40	1030.88	11459.90	2243.36	10547.44	699.6	10181.75
Golden 8	13039.80	2820.55	13034.50	6154.77	12036.24	664.8	11663.55
Golden 9	590.41	315.16	589.60	558.53	593.35	700.2	583.39
Golden 10	755.33	737.13	749.78	1577.09	751.66	949.8	741.56
Golden 11	970.99	1882.78	961.64	6409.84	936.04	1987.2	918.45
Golden 12	1173.37	430.63	1173.37	1086.11	1147.14	2574.0	1107.19
Golden 13	904.86	146.75	904.86	251.52	868.80	685.8	859.11
Golden 14	1147.13	291.25	1143.47	688.95	1096.18	870.6	1081.31
Golden 15	1448.23	287.66	1439.71	640.97	1369.44	1107.0	1345.23
Golden 16	1709.84	638.38	1706.25	1632.23	1652.32	1384.2	1622.69
Golden 17	717.59	223.33	717.26	354.05	711.07	857.4	707.79
Golden 18	1025.26	308.44	1023.64	565.28	1016.83	1287.0	997.52
Golden 19	1405.09	873.06	1403.39	1102.83	1400.96	1801.8	1366.86
Golden 20	1879.95	1907.17	1879.44	2107.05	1915.83	2583.0	1820.09
Gap(%)	9.44		9.18		4.18		
CPU(s)	861.7		2299.5		1052.7		

See the footnotes of <Table 4>.

6. Concluding Remarks

In this paper, we considered the capacitated vehicle routing problem that determines a set of routes by a fleet of vehicles, starting and terminating at the depot, to serve a given set of customers under the vehicle capacity and the travel time restrictions for the objective of minimizing the total traveling cost. To solve the problem, a tabu search algorithm was suggested that incorporates the features of the existing algorithms. In particular, we suggested a neighborhood reduction method using the proximity information of the Voronoi diagram. To show the performances of the Voronoi tabu search algorithm suggested in this paper, computational experiments were done on various benchmark problems and the results show that our algorithm is competitive to the existing algorithms, especially in terms of computation times.

Although we could not suggest the best algorithm for the capacitated vehicle routing problem, this research has a certain contribution in that the Voronoi diagram was firstly adopted to solve the capacitated vehicle routing problem. Based on this, this research can be extended in several ways. First, it may be needed to develop other algorithms using the features of the Voronoi diagram for the capacitated vehicle routing problem. Second, the proximity information of the Voronoi diagram can be applied to other vehicle routing problems.

References

Barbarosoglu, G. and Ozgur, D. (1999), A tabu search algorithm for the vehicle routing problem, *Computers and Operations Research*, **26**, 255-270.

Clarke, G. and Wright, J. W. (1964), Scheduling of vehicles from a central depot to a number delivery points, *Operations Research*, **12**, 568-581.

Christofides, N., Mingozzi, A., and Toth, P. (1979), The vehicle routing problem. In: Christofides, N., Mingozzi, A., Toth, P., Sandi, C., editors, *Combinatorial Optimization*, Chichester: Wiley, 315-338.

Cordeau, J.-F. and Laporte, G. (2002), Tabu search heuristics for the vehicle routing problem, Technical Report, GERAD, École des Hautes Études Commerciales, Montréal, Canada.

Dantzig, G. B. and Ramser, J. H. (1959), The truck dispatching problem, *Management Science*, **6**, 80-91.

Gendreau, M., Hertz, A., and Laporte, G. (1992), New insertion and post-optimization procedures for the traveling salesman problem, *Operations Research*, **40**, 1086-1094.

Gendreau, M., Hertz, A., and Laporte, G. (1994), A tabu search heuristic for the vehicle routing problem, *Management Science*, **40**, 1276-1290.

Gillett, B. and Miller, L. (1974), A heuristic algorithm for the vehicle dispatch problem, *Operations Research*, **22**, 340-349.

Glover, F. (1989), Tabu Search: Part I, *ORSA Journal of Computing*, **1**, 190-206.

Glover, F. (1990), Tabu Search: Part II, *ORSA Journal of Computing*, **2**, 4-32.

Glover, F. and Laguna, M. (1993), *Tabu search*. In: Reeves, C., editor, Modern Heuristic Techniques for Combinatorial Problems, Blackwell Scientific Publications, Oxford, 70-150.

Golden, B. L., Wasil, E. A., Kelly, J. P., and Chao, I.-M. (1998), The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results. In: Crainic, T. G., Laporte, G., editors, *Fleet Management and Logistics*, Boston: Kluwer, 33-56.

Kim, D.-S., Kim, D.-U. and Sugihara, K. (2001a), Voronoi diagram of a circle set from voronoi diagram of a point set: I. Topology, Computer Aided Geometric Design, **18**, 541-562.

Kim, D.-S., Kim, D.-U. and Sugihara, K (2001b), Voronoi diagram of a circle set from voronoi diagram of a point set: II. Geometry, Computer Aided Geometric Design, **18**, 563-585.

Laporte, G. (1992), The vehicle routing problem: an overview of exact and approximation algorithms, *European Journal of Operational Research*, **59**, 345-358.

Laporte, G., Gendreau, M., Potvin, J.-Y., and Semet, F. (2000), Classical and modern heuristics for the vehicle routing problem, *International Transactions in Operations Research*, **7**, 285-300.

Lee, S.-H. and Lee, J. M. (2005), Study on the Heterogeneous Fleet Vehicle Routing Problem with Customer Restriction, *Journal of the Korean Institute of Industrial Engineers*, 31, 228-239.

Lenstra, J. and Rinnooy Kan, A. (1981), Complexity of vehicle routing and scheduling problems, *Networks*, **11**, 221-228.

Lin, S. and Kernighan, B. W. (1973), An effective heuristic algorithm for the traveling salesman problem, *Operations Research*, **21**, 503-511.

Oh, S. C., Yee, S. T., and Kim, T. Y., (2006), Agent-based Shipment Algorithm for Capacitated Vehicle Routing Problem with Load Balancing, *Journal of the Korean Institute of Industrial Engineers*, **32**, 200-209.

Okabe, A., Boots, B., and Sugihara, K. (1992), *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams,* John Wiley & Sons.

Osman, I. H. (1993), Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problems, *Annals of Operations Research*, **41**, 421-452.

Osman, I. H. and Christofides, N. (1989), Simulated annealing and descent algorithms for capacitated clustering problems, presented as EURO-XI, Beograd, Yugoslavia.

Rochat, Y. and Taillard, E. (1995), Probabilistic diversification and intensification in local search for vehicle routing, *Journal of Heuristics*, **1**, 147-167.

Seong, J. and Moon, I. K. (2006), Vehicle Routing Problem with Time Windows considering Outsourcing Vehicles, *Journal of the Korean Institute of Industrial Engineers*, **32**, 91-97.

Tarantilis, C. D. and Kiranoudis, C. T. (2002), Boneroute: an adaptive memory-based method for effective fleet management, *Annals of Operations Research*, **115**, 227-241.

Toth, P. and Vigo, D. (2003), The granular tabu search and its application to the vehicle routing problem, *INFORMS Journal on Computing*, **15**, 333-348.