A Hybrid Genetic Algorithm for the Vehicle Routing Problem with Time Windows

Soonchul Jung and Byung-Ro Moon
School of Computer Science and Engineering
Seoul National University
Seoul, 151-742 Korea
{samuel,moon}@soar.snu.ac.kr

Abstract

This paper suggests a new hybrid genetic algorithm for the 2D Euclidean vehicle routing problem with time windows. The natural crossover, proposed for the 2D Euclidean traveling salesman problem, was adopted with some modification in the suggested genetic algorithm. The most notable feature of the natural crossover is that it uses the 2D image of a solution itself for chromosomal cutting. We also investigate the usefulness of parents' decision variables such as arrival times during recombination. The suggested genetic algorithm found optimal solutions for 26 out of 31 instances with known optimal solutions.

1 Introduction

The vehicle routing problem (VRP) is the problem of finding a set of minimum-cost vehicle routes which start at a central depot, serve a set of customers with known demands, and return to the depot without any violation of constraints [6], [24]. There are several variants of VRP depending on their constraints. The vehicle routing problem with time windows (VRPTW) is an extension of VRP. In VRPTW, time-window constraints are added to the basic constrains of VRP. Each customer must be served only once by one vehicle, and the total demands of the customers served by a particular vehicle must not exceed the capacity of the vehicle. Each customer must be served within his/her time window. A vehicle must wait until the service is possible if the vehicle arrives at a customer earlier than the lower bound of his/her time window – the earliest arrival time. The depot also has a time window, and all the vehicles must return by the latest arrival time of the depot. In VRPTW, the objective can be the minimization of the travel distance, the travel time, the number of vehicles, or their combinations.

VRPTW has shown its usefulness in the area of distribution-related systems — school bus routing, newspaper delivery, garbage collection, fuel oil delivery, dial-a-ride service, etc. If a new routing plan of vehicles is more efficient than before, we can save fuel, money, and/or time.

Various algorithms for VRP and its variants have been studied intensively for decades. There are some of exact methods to solve VRPs and VRPTWs to the optimality [10], [12], [19], [21]. Although the speed-up techniques for exact methods have been introduced, the NP-hardness of VRPTW [27] still makes the required computational time prohibitive. Local heuristic methods often produce good near-optimal solutions in short computational time. They are divided into two classes: route construction heuristics [13], [11] and route improvement heuristics [7], [32]. Solomon [26] designed and reviewed several route construction heuristics. In [18], A number of route improvement heuristics are clearly described. Although these heuristics were able to run separately to solve VRPTW, they have also been incorporated in meta heuristics like tabu search, simulated annealing, genetic algorithm (GA), etc [4], [5], [30], [23], [15].

Blanton and Wainwright [2] introduced a genetic algorithm for VRPTW that used a sequence of customers as a chromosome. A greedy insertion-based heuristic interprets a chromosome (sequence), and calculates the fitness of the chromosome. The sequence means the insertion order of customers for the heuristic. GIDEON, suggested by Thangiah *et al.* [31], [30], is a framework for VRPTW, adopting a cluster-first route-second strategy. Their GA was used in the clustering phase. The chromosome represents angles whose origin is the depot, in order to define sectors to which customers will belong. Customers within a sector are assigned to one vehicle, and routed by the cheapest insertion method

[14]. In another system of Thangiah [29], a chromosome represents circles (by defining an origin and a radius). Customers within or near a circle are assigned to one vehicle. GENEROUS of Potvin and Bengio [23] uses a set of routes themselves as a chromosome. The crossover merges two parents heuristically, then the repair operator is applied to the offspring. Most approaches use a set of routes themselves as a chromosome after Potvin and Bengio's work [23]. Recently, Tan *et al.* [28] introduced a messy genetic algorithm that a chromosome is a sequence of (customer number, vehicle number) pairs.

The natural crossover, introduced by Jung and Moon [17], [16] is a crossover manipulating chromosomes in which genes are laid on a 2D space. Because 2D chromosomes can preserve problem information with less distortion, two-dimensional chromosomes are often used in genetic algorithms for 2D problems [8], [1], [3]. The natural crossover was originally devised for the 2D Euclidean traveling salesman problem (TSP), and produced better experimental results than state-of-the-art GAs for TSP [17]. In most VRPTW instances including Solomon's benchmark instances [26], customers are located on a 2D Euclidean space. Therefore, only if a 2D form of chromosomes are defined for VRPTW, it is possible for a genetic algorithm to use the natural crossover. This paper provides an extension of the natural crossover to VRPTW, and investigates its competence.

There are variables created during the evaluation of a route. Waiting times, arrival times, and travel-so-far distances are some of such decision variables. Almost all genetic algorithms for VRPTW did not utilize parents' decision variables in the course of recombination and mutation. In this paper, we utilize parents' decision variables during crossover.

The paper is organized as follows. Section 2 describes the mathematical formulation of VRPTW. Section 3 explains the genetic operators used in the proposed genetic algorithm. Section 4 presents the experimental results. Finally Section 5 makes conclusions.

2 Formulation of VRPTW

Table 1 represents the meanings of terms related to VRPTW. For a given route $R_k = (v_1, v_2, ..., v_m), v_1 = c_0$, decision variables are calculated as follows:

$$
\begin{aligned}
a_{v_i} &= \begin{cases} 0, & i = 1 \\ tt_{v_{i-1}} + t_{v_{i-1}v_i}, & i > 1 \end{cases} \\
w_{v_i} &= \begin{cases} 0, & i = 1 \\ \max(0, e_{v_i} - a_{v_i}), & i > 1 \end{cases} \\
tt_{v_i} &= \begin{cases} 0, & i = 1 \\ a_{v_i} + w_{v_i} + s_{v_i}, & i > 1 \end{cases}
\end{aligned}
$$

Table 1: Terminologies

constants	meaning
N	number of customers
Q	capacity of vehicles
C	set of all customers including the depot
c_i	customer i, $0 \le i \le N$ (c_0 is the depot.)
d_{ij}	distance from customer i to customer j
t_{ij}	travel time from customer i to customer j
q_i	demand of customer i
s_i	service time of customer i
e_i	earliest arrival time to customer i
l_i	latest arrival time to customer i
variables	**meaning**
$n(R)$	number of routes
$n(R_k)$	number of customers in route k
v_{jk}	j^{th} customer of route k
R_k	route $k = (v_{1k}, v_{2k}, ...)$
S_k	set of customers in route k
RD_k	travel distance of route k
RT_k	travel time of route k
RL_k	total load of route k
a_i	arrival time at c_i
al_i	adjusted latest arrival time at c_i
w_i	waiting time before servicing c_i
tt_i	travel-so-far time after servicing c_i
td_i	travel-so-far distance when arriving at c_i
ad_i	accumulated demands of customers after servicing c_i

$$
\begin{aligned}
td_{v_i} &= \begin{cases} 0, & i = 1 \\ td_{v_{i-1}} + d_{v_{i-1}v_i}, & i > 1 \end{cases} \\
ad_{v_i} &= \begin{cases} 0, & i = 1 \\ ad_{v_{i-1}} + q_{v_i}, & i > 1 \end{cases} \\
RT_k &= tt_{v_m} + t_{v_m v_1} \\
RD_k &= td_{v_m} + d_{v_m v_1} \\
RL_k &= ad_{v_m}
\end{aligned}
$$

The objective of our genetic algorithm is to find a set of routes having the minimal travel distance. In other words, we have to minimize

$$\sum_{k=1}^{n(R)} RD_k$$

subject to

$$S_1 \cup S_2 \cup \cdots \cup S_{n(R)} = C, \quad (1)$$

$$S_i \cap S_j = \{c_0\}, \quad 1 \le i, j \le n(R), i \ne j \quad (2)$$

$$v_{ik} \ne v_{jk}, \quad 1 \le i, j \le n(R_k), i \ne j \quad (3)$$

$$a_i \le l_i, \quad 1 \le i \le N \quad (4)$$

```
GA()
{
    initialize population P of size N;
    while ( stopping condition is unsatisfied ) {
        select parent_1 and parent_2 from P;
        offspring ← crossover(parent_1, parent_2);
        if ( random number is larger than mutation rate )
            mutate offspring;
        local-optimize offspring;
        replace an individual in P with offspring;
    }
    return the best individual;
}
```

Figure 1: A typical steady-state hybrid genetic algorithm

$$RT_k \leq l_0, \qquad 1 \leq k \leq n(R) \qquad (5)$$
$$RL_k \leq Q, \qquad 1 \leq k \leq n(R). \qquad (6)$$

Restriction (1) ensures that all the customers are necessarily visited. Restriction (2) means that all the customers must be partitioned disjoint, and the depot is included in all routes. Restriction (3) ensures that every customer is visited only once. Restriction (4) and (5) take care of time constraints. Restriction (6) prevents the overload of vehicles.

3 Hybrid Genetic Algorithm

We use a typical steady-state hybrid genetic algorithm (Figure 1). Local optimization algorithms help GAs fine-tuning around local optima. The following subsections describe our genetic algorithm in detail.

3.1 Initialization of Population

A lot of route construction heuristics have been proposed for VRPTW. In [26], several heuristics are carefully designed and compared with one another. According to [26], the insertion heuristic 1 (I1) overall beat other route construction heuristics such as savings, nearest neighbor, etc. The core part of I1 is the routine inserting a new unrouted customer into the current route, between two adjacent customers on the route. If there is no feasible customer to insert, a new route is created. I1 repeats the loop until there are no unrouted customers.

We create a population of solutions using a stochastic version of I1. The existence of adjustable weights in the cost function of I1 makes the creation of various solutions possible. In calculating an insertion cost of a customer, the weights determine the balance between the spatial aspect and the temporal aspect of the problem instance. The values of weights are changed at random in the ranges of $\mu \in [0.1, 0.9], \lambda \in [0, 1], \alpha_1 \in [0.1, 0.9]$, and $\alpha_2 \in [0.1, 0.9] (\alpha_1 + \alpha_2 = 1)$. This is expected to be helpful in creating a robust population whose solution qualities do not depend on specific aspects of the problem instance. The first customer for a new route is chosen at random among the farthest unrouted customer, the unrouted customer with the earliest deadline, and a random unrouted customer.

3.2 Selection and Crossover

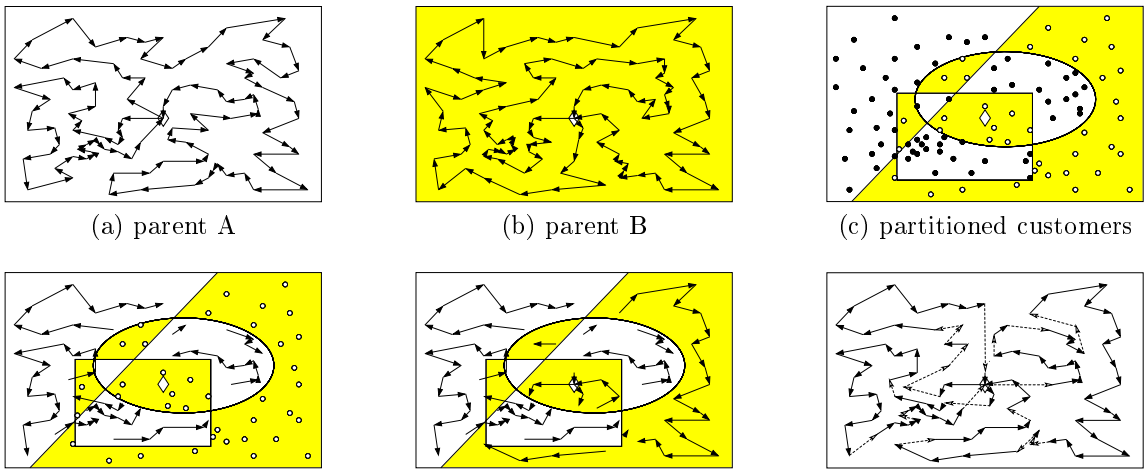
We use the typical binary tournament selection.

Most genetic algorithms for VRPTW do not consider the representation of chromosomes as important, and they recombine the new offspring by heuristically interpreting the two parents. In recombining the offspring, they consider a number of criteria like distances between customers, ranges of time windows, sizes of routes, distribution of distances, etc; but they do not consider the physical locations of customers. In GIDEON system of Thangiah *et al.* [31], [30], each chromosome contains a set of numbers representing the angles defining sectors (centered at the depot) instead of routes themselves. Customers in a sector basically belong to the same route. Namely, this system considers the locations of customers to be more important than other elements.

Multi-dimensional chromosomes were suggested for problems with multi-dimensional characteristics. A two-dimensional crossover, introduced by Cohoon and Paris [8], chooses a small rectangle from one parent and then copies the genes in the rectangle into the offspring with the rest of genes copied from the other parent. Anderson *et al.* [1] suggested a block-uniform crossover which tessellates a 2D chromosome into $i \times j$ blocks, and copies the genes block by block from a uniformly selected parent. Bui and Moon [3] proposed a generalization of crossovers to n dimensions. Jung and Moon [17], [16] introduced an encoding/crossover pair for the 2D Euclidean traveling salesman problem which uses a 2D image as a chromosome, and performs crossover on the chromosome. Since 2D Euclidean VRPs and 2D Euclidean TSPs share a lot of characteristics, we inherit the natural encoding/crossover pair with some modification.

In this paper, we use the 2D image of routes as a chromosome, where each gene is located at the coordinate of the corresponding customer. We describe the natural crossover for the 2D Euclidean VRPTW in the following:

1. The 2D image of two solutions are selected as parents (Figure 2 (a),(b)).



(a) parent A (b) parent B (c) partitioned customers

(d) inherited arcs from parent A (e) adding arcs from parent B (f) new offspring

◊ The diamond represents the depot.

Figure 2: An example of the natural crossover for VRPTW

2. Free curves or figures are drawn on the 2D space where customers are located[1]. It is proven that they always partition the chromosomal space into two equivalent classes [17] (marked white and gray in Figure 2 (c)). Every customer belongs to one of the classes. Customers in the white class are marked black and customers in the other gray class are marked white.

3. For every arc of the parent A, if both of the start-point and the end-point are marked black[2], it survives in the offspring (Figure 2(d)); for every arc of the parent B, if both are marked white, it survives in the offspring (Figure 2(e)). Then we have a number of disconnected segments.

4. The decision variables of the parent A such as a_i, w_i, tt_i, and td_i are saved as a_i^A, w_i^A, tt_i^A, and td_i^A, respectively. The decision variables of the parent B are saved in the same way. They are used in repairing the offspring later.

5. A valid solution is made by adding arcs by the repair algorithm in Section 3.2.1 (Figure 2(f)).

Because we only have to calculate the class of every customer, the time complexity of the crossover grows linearly with respect to the number of customers.

[1]We do not have an efficient implementation for drawing fully free curves. Instead, we use four types of curves — straight line, triangle, quadrangle, and ellipse. Two curves are chosen at random among them allowing multiple occurrences. Refer to [16] for more information.

[2]It is possible that the arc passes through the gray region even when both points are marked black; there are a few arcs in Figure 2 (d). For efficient implementation, we ignore classes of arcs.

3.2.1 Repair Algorithm

The step 5 in the previous section repairs the intermediate offspring to a valid solution. We utilize the parents' decision variables in this process. Figure 3 represents the repair algorithm. Its key routine is connecting the last customer on the current partial route to the minimum-cost start-point of a segment in a nearest-neighbor manner.

In calculating the cost of adding an arc, we consider terms about spatial and temporal closenesses of customers, and terms about parents. Let c_i be the last customer (point) on the current partial route, and let c_j be a candidate customer to connect c_i. The cost is the weighted sum of i) the distance between c_i and c_j, ii) the waiting time at c_j, iii) the slack time of delivery to c_j, and iv) the difference of the service completion between parents and the offspring at c_j:

$$c_{ij} = \delta_1 \cdot d_{ij} + \delta_2 \cdot W_j + \delta_3 \cdot S_j + \delta_4 \cdot P_j$$

where

$$\begin{aligned} \delta_1 + \delta_2 + \delta_3 + \delta_4 &= 1, \\ A_j &= t_i + t_{ij}, \\ W_j &= \max(0, e_j - A_j), \\ S_j &= l_j - A_j, \text{ and} \\ P_j &= \min(|tt_j^A - (A_j + W_j + s_j)|, \\ &\qquad |tt_j^B - (A_j + W_j + s_j)|). \end{aligned}$$

$\delta_1, \delta_2, \delta_3$, and δ_4 are reinitialized within respective specific ranges whenever the repair function is invoked.

To investigate the usefulness of parents' decision variables, we compare in Section 4 a GA version with $\delta_4 \neq 0$ against one with $\delta_4 = 0$.

```
repair()
{
        while( there are segments starting from the depot ) {
                randomly choose a segment among them as a partial route;
                complete_a_route(the last customer of the partial route, start-points of the remaining segments);
        }
        while( there are remaining segments )
                complete_a_route(depot, start-points of the segments);
}
complete_a_route(the last customer, candidate customers)
{
        t ← the last customer;
        do {
                find a feasible customer, say c*, among candidate customers and the depot,
                     such that the cost from t to c* is minimized;
                add an arc from t to c*;
                t ← the end-point of the segment whose start-point is c*;
        } while( t is not the depot )
}
```

Figure 3: The pseudo-code of the repair algorithm

3.3 Mutation

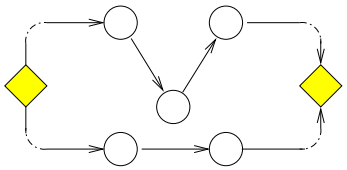
In mutation, each route of the offspring is split into at most three routes. Two cut-points are selected at random to split a route.

3.4 Local Optimization

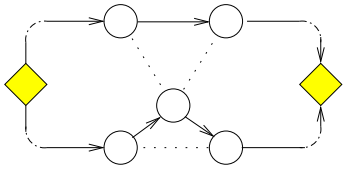
A considerable number of local optimization algorithms have been proposed to improve routes. Most of them belong to edge-exchange neighborhoods [18]. Most edge-exchange neighborhoods can be viewed as special cases of the cyclic transfer algorithm introduced by Thompson and Psaraftis [32]. Although the cyclic transfer algorithm is a generalized edge-exchange algorithm, its performance is limited due to its computational cost.

We call three local optimization heuristics in sequence — Or-opt [22], crossover [25], relocation [25] — to optimize the offspring locally. These three heuristics have different characteristics from one another; they are thought to produce synergies. The Or-opt is a vertex-based algorithm trying to move a vertex to another place within a single route. The crossover is a special type of two-edge exchange which removes the cross links of two routes. The relocation is similar to the Or-opt; it is different in that it manipulates multiple routes. Its key routine is moving a vertex in a route to another place in other routes (an example in Figure 4).

In VRPTW, it consumes considerable CPU time to check the time-feasibility of a solution. Consider the routine which checks whether an unrouted customer u can be inserted between two specific adjacent customer



(a) before relocation



(b) after relocation

For the sake of convenience, the depot was depicted as two diamonds.

Figure 4: An example of a relocation

v_{p-1} and v_p on the route $R_k = (v_1, v_2, ..., v_m)$. The routine must check the time-feasibility at $u, v_p, v_{p+1}, ...,$ and v_m, respectively. Solomon introduced *Push Forward* [26] to practically speed up this kind of operation. However, the time-feasibility checking takes $O(n)$ in the worst case even when using Push Forward (n is the number of customers in the route.).

We use the *adjusted latest arrival time* (al_{v_i}), instead [20]. The adjusted latest arrival times at customers on a route are computed as follows:

$$al_{v_i} = \begin{cases} l_0 - t_{v_i c_0}, & i = m \\ \min(l_{v_i}, al_{v_{i+1}} - t_{v_i v_{i+1}} - s_{v_i}), & i < m. \end{cases}$$

The adjusted latest arrival time of a customer is the

time by which the vehicle must arrive at the customer to satisfy the time-feasibility with no further checking. Using the adjusted latest arrival times, the time-feasibility check is completed in constant time even in the worst case. In other words, when inserting an unrouted customer u between v_{p-1}, and v_p, it only have to check to see if $tt_{v_{p-1}} + t_{v_{p-1}u} + w_u + s_u + t_{uv_p} \leq al_{v_p}$.

The Or-opt algorithm and the relocation algorithm call the above routine very frequently, and thus the time-feasibility is checked fast using the adjusted latest arrival time described in Section 3.1.

3.5 Replacement

The offspring is compared with one of the parents. The parent is replaced to the offspring if the offspring is better. Otherwise, the other parent is replaced if the offspring is better than it. Otherwise, the worst in the population is replaced.

3.6 Stop Condition

Our GA stops when the best solution has not been broken during p consecutive generations. p was set to 2,000.

4 Experimental Results

We set a GA with $\delta_1 \in [0.4, 0.9], \delta_2 \in [0.2, 0.7], \delta_3 \in [0.1, 0.6]$, and $\delta_4 \in [0.3, 0.8]$[3] and call it VGA1.

We programmed our GA in C++ language. In the experiment, the population size and the mutation rate were set to 60 and 0.05, respectively. 100 runs were performed for each Solomon's VRPTW instance [26].

4.1 Performance

Table 2 shows the experimental results of VGA1 and TLOL [28], a recent representative paper from the field of the evolutionary computation. Inter-customer distances were calculated with real double-precision. VGA1-best and VGA1-average represent the best and the average results of VGA1 over 100 runs, respectively. For TLOL, only the best results are available [28]. The figures were rounded off to two decimal places. "#V" and "TD" mean the number of vehicles and the travel distance, respectively. "t" represents the average CPU seconds on Pentium III 1GHz.

VGA1 outperformed TLOL for 47 of the 56 instances; TLOL outperformed VGA1 for two of them; they tied for the other seven instances. Among the 47 cases that VGA1 outperformed, even the average results of VGA1 were better than TLOL (the best results) for all of them except one (R206).

In Table 3, we compared VGA1-best with the optimal solutions[4] reported in [9]. Inter-customer distances were truncated to the first decimal place to be consistent with [9]. The bold-faced numbers represent that the results of VGA1 equal the optimal solutions. VGA1 found optimal solutions for 26 out of 31 instances whose optimal solutions are known. VGA1 found most of the optimal solutions for the C and R groups, but it found the optimum for one of the five in the RC group.

4.2 The Usefulness of Parents' Decision Variables During Crossover

To test the usefulness of parents' decision variables, we set another GA with $\delta_1 \in [0.4, 0.9], \delta_2 \in [0.5, 1.0], \delta_3 \in [0.1, 0.6]$, and $\delta_4 = 0$ (VGA2). Because $\delta_4 \neq 0$, VGA1 utilized parents' decision variables (parents' travel-so-far times, in detail), while VGA2 did not.

Table 4 shows the results of VGA1 and VGA2 for each problem group. Each group of problems has about 10 instances, e.g., C1 has C101 through C109. The best ("Best") and average ("Average") results of each group are the averages of the best and average results for the corresponding instances, respectively.

According to the best results, VGA1 found better solutions more frequently than VGA2, but VGA2 was better on the average. In other words, the deviation of the best and average results in VGA1 was larger than in VGA2. VGA1 seems to be strong in instances that have long scheduling horizons and large vehicle capacities (C2, R2, and RC2 groups), although there are only slight differences compared to VGA2. Overall, the performances of VGA1 and VGA2 were comparable. It is notable that VGA1 was about 30% faster than VGA2.

5 Conclusion

In this paper, we suggested a new hybrid genetic algorithm for the 2D Euclidean vehicle routing problem with time windows. In our genetic algorithm, the 2D image itself of a solution becomes a chromosome; each gene corresponds to a customer in the 2D plane; the natural crossover cuts the chromosomal space with free curves. Because of its simplicity, the natural crossover may be applied to other variants of VRP with minor modification. The experimental results showed that the suggested hybrid genetic algorithm solved VRPTWs to

[3] If $\delta_1 + \delta_2 + \delta_3 + \delta_4 > 1$, then they are scaled down to satisfy that their sum is equal to 1.

[4] http://web.cba.neu.edu/~msolomon/problems.htm. After [9], some more optimal solutions were added.

Table 2: Experimental Results of VGA1

	TLOL		VGA1-best		VGA1-average				TLOL		VGA1-best		VGA1-average		
Instance	#V	TD	#V	TD	#V	TD	t	Instance	#V	TD	#V	TD	#V	TD	t
C101	10	**828.94**	10	**828.94**	10.00	**828.94**	6	C201	3	**591.56**	3	**591.56**	3.00	**591.56**	9
C102	10	**828.94**	10	**828.94**	10.00	**828.94**	12	C202	3	**591.56**	3	**591.56**	3.00	**591.56**	16
C103	10	859.78	10	**828.06**	10.00	**828.06**	18	C203	3	618.00	3	**591.17**	3.00	**591.17**	30
C104	10	893.23	10	**824.78**	10.00	**824.96**	29	C204	3	609.02	3	**590.60**	3.00	**591.18**	44
C105	10	**828.94**	10	**828.94**	10.00	**828.94**	7	C205	3	616.32	3	**588.88**	3.00	**588.88**	10
C106	10	**828.94**	10	**828.94**	10.00	**828.94**	7	C206	3	615.92	3	**588.49**	3.00	**588.49**	12
C107	10	**828.94**	10	**828.94**	10.00	**828.94**	7	C207	3	636.62	3	**588.29**	3.00	**588.29**	13
C108	10	830.94	10	**828.94**	10.00	**828.94**	8	C208	3	611.29	3	**588.32**	3.00	**588.32**	12
C109	10	849.03	10	**828.94**	10.00	**828.94**	9								
R101	18	1648.86	20	**1642.88**	20.00	**1643.53**	30	R201	8	1198.15	9	**1149.68**	8.29	**1153.04**	64
R102	17	1486.71	18	**1472.81**	18.50	**1479.19**	52	R202	9	1057.56	8	**1034.35**	7.40	**1038.40**	81
R103	14	1234.43	14	**1213.62**	14.81	**1222.29**	51	R203	5	922.38	6	**874.87**	6.00	**875.87**	84
R104	11	1024.38	11	**976.61**	11.70	**1001.44**	61	R204	5	791.78	4	**736.52**	4.46	**741.41**	92
R105	15	1372.71	15	**1360.78**	15.91	**1371.52**	34	R205	5	1015.99	5	**955.82**	6.05	**964.69**	70
R106	12	1271.11	13	**1240.47**	13.59	**1252.44**	48	R206	4	884.65	5	**879.89**	5.33	892.55	93
R107	12	1106.19	11	**1073.34**	11.73	**1083.10**	63	R207	4	875.76	4	**799.86**	4.66	**814.05**	97
R108	9	992.12	10	**947.55**	10.74	**959.65**	65	R208	3	778.38	4	**705.45**	3.50	**714.37**	99
R109	13	**1101.37**	13	1151.84	12.97	1157.27	41	R209	3	920.34	5	**859.39**	5.26	**867.52**	83
R110	11	1119.12	12	**1072.41**	12.00	**1082.72**	53	R210	4	961.18	5	**910.70**	6.10	**918.37**	102
R111	12	1083.05	12	**1053.50**	12.00	**1063.21**	56	R211	6	820.23	4	**755.96**	4.70	**765.64**	96
R112	11	1020.52	10	**953.63**	10.77	**971.89**	49								
RC101	14	1659.68	16	**1643.41**	16.46	**1658.34**	28	RC201	4	1354.96	9	**1265.56**	9.00	**1269.94**	50
RC102	15	1492.10	14	**1461.23**	14.65	**1480.82**	45	RC202	8	1151.46	8	**1095.64**	7.84	**1101.03**	74
RC103	11	**1249.86**	12	1277.54	12.11	1313.73	48	RC203	7	1018.09	5	**928.51**	5.29	**943.81**	104
RC104	11	1202.12	10	**1136.81**	10.56	**1154.26**	57	RC204	4	865.51	4	**786.38**	4.05	**799.19**	81
RC105	16	1585.34	16	**1518.58**	15.96	**1540.66**	42	RC205	9	1225.69	7	**1157.55**	7.80	**1164.43**	69
RC106	12	1449.30	13	**1381.23**	13.39	**1397.45**	33	RC206	5	1122.23	7	**1054.61**	6.39	**1067.49**	64
RC107	11	1303.36	12	**1212.83**	12.03	**1227.81**	32	RC207	6	1047.86	6	**966.08**	6.07	**975.24**	76
RC108	11	1197.13	11	**1117.53**	11.00	**1135.81**	32	RC208	4	854.75	4	**779.31**	4.98	**791.35**	68

the near-optimality.

We also tested the usefulness of parents' decision variables during crossover. In terms of solution qualities, the use of parents' decision variables did not give notable improvement; but, it shortened the running time.

We used the synergy of three local optimization heuristics. Some stronger local optimization heuristic may further improve the hybrid genetic algorithm. This part is left for future study.

Acknowledgements

This work was partly supported by KOSEF through Statistical Research Center for Complex Systems at Seoul National University and Brain Korea 21 Project. The RIACT at Seoul National University provided research facilities for this study.

References

[1] C. A. Anderson, K. F. Jones, and J. Ryan. A two-dimensional genetic algorithm for the Ising problem. *Complex Systems*, 5:327–333, 1991.

[2] J. Blanton and R. Wainwright. Multiple vehicle routing with time and capacity constraints using genetic algorithms. In *Fifth International Conference on Genetic Algorithms*, pages 452–459, 1993.

[3] T. N. Bui and B. R. Moon. On multi-dimensional encoding/crossover. In *Sixth International Conference on Genetic Algorithms*, pages 49–56, 1995.

[4] W. Chiang and R. Russell. Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Annals of Operations Research*, 63:3–27, 1996.

[5] W. Chiang and R. Russell. A reactive tabu search metaheuristics for the vehicle routing problem with time windows. *INFORMS Journal on Computing*, 9:417–430, 1997.

[6] N. Christofides. Vehicle routing. In E. Lawler, J. Lenstra, A. Rinnooy Kan, and D. Shmoys, editors, *The Traveling Salesman Problem*, pages 431–448. John Wiley & Sons, 1985.

[7] G. Clarke and J. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12:568–582, 1964.

[8] J. P. Cohoon and W. Paris. Genetic placement. *IEEE Trans. on Computer-Aided Design*, CAD-6(6):956–964, 1987.

[9] J. Cordeau, G. Desaulniers, J. Desrosiers, M. M. Solomon, and F. Soumis. The VRP with time windows. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, pages 157–193. SIAM Monographs on Discrete Mathematics and Applications, 2002.

[10] M. Desrochers, J. Desrosiers, and M.M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40:342–354, 1992.

[11] M. Fisher and R. Jaikumar. A generalized assignment heuristic for vehicle routing. *Networks*, 11:109–124, 1981.

[12] M. Fisher, K. Jornsten, and O. Madsen. Vehicle routing with time windows: Two optimization algorithms. *Operations Research*, 45:488–492, 1995.

[13] B. Gillet and L. Miller. A heuristic algorithm for the vehcile distpatch problem. *Operations Research*, 22:340–349, 1974.

[14] B. Golden and W. Stewart. Empirical analysis of heuristics. In E. Lawler, J. Lenstra, A. Rinnooy Kan, and D. Shmoys, editors, *The Traveling Salesman Problem*, pages 207–249. John Wiley & Sons, 1985.

[15] J. Homberger and H. Gehring. Two evolutionary metaheuristics for the vehicle routing problem with time windows. *INFOR*, 37:297–318, 1999.

Table 3: Comparison with Optimal Solutions

Instance	Optimum #V	Optimum TD	VGA1-best #V	VGA1-best TD	Instance	Optimum #V	Optimum TD	VGA1-best #V	VGA1-best TD	Instance	Optimum #V	Optimum TD	VGA1-best #V	VGA1-best TD
C101	10	827.3	10	**827.3**	R101	20	1637.7	20	**1637.7**	RC101	15	1619.8	16	1637.8
C102	10	827.3	10	**827.3**	R102	18	1466.6	18	**1466.6**	RC102	14	1457.4	14	1473.5
C103	10	826.3	10	**826.3**	R103	14	1208.7	14	**1208.7**	RC103	11	1258.0	11	1273.4
C104	10	822.9	10	**822.9**	R104	-	-	11	971.5	RC104	-	-	10	1132.8
C105	10	827.3	10	**827.3**	R105	15	1355.3	15	**1355.3**	RC105	15	1513.7	15	**1513.7**
C106	10	827.3	10	**827.3**	R106	13	1234.6	13	**1234.6**	RC106	-	-	13	1373.9
C107	10	827.3	10	**827.3**	R107	11	1064.6	11	**1064.6**	RC107	-	-	12	1209.3
C108	10	827.3	10	**827.3**	R108	-	-	10	935.1	RC108	-	-	11	1114.2
C109	10	827.3	10	**827.3**	R109	13	1146.9	13	**1146.9**					
					R110	12	1068.0	12	**1068.0**					
					R111	12	1048.7	12	1049.6					
					R112	-	-	10	948.6					
C201	3	589.1	3	**589.1**	R201	8	1143.2	8	**1143.2**	RC201	9	1261.8	9	1262.4
C202	3	589.1	3	**589.1**	R202	-	-	8	1029.6	RC202	-	-	8	1092.3
C203	3	588.7	3	**588.7**	R203	-	-	6	870.8	RC203	-	-	5	925.5
C204	-	-	3	588.1	R204	-	-	4	731.8	RC204	-	-	4	783.5
C205	3	586.4	3	**586.4**	R205	-	-	5	951.3	RC205	-	-	7	1154.0
C206	3	586.0	3	**586.0**	R206	-	-	5	875.9	RC206	-	-	7	1051.1
C207	3	585.8	3	**585.8**	R207	-	-	4	797.1	RC207	-	-	6	962.9
C208	3	585.8	3	**585.8**	R208	-	-	4	701.4	RC208	-	-	5	779.6
					R209	-	-	5	854.8					
					R210	-	-	6	901.8					
					R211	-	-	4	746.7					

Table 4: Results of VGA1 and VGA2

Group	VGA1 Best #V	VGA1 Best TD	VGA1 Average #V	VGA1 Average TD	VGA1 t	VGA2 Best #V	VGA2 Best TD	VGA2 Average #V	VGA2 Average TD	VGA2 t
C1	10.00	**828.38**	10.00	828.40	12	10.00	**828.38**	10.00	**828.38**	13
C2	3.00	**589.86**	3.00	**589.93**	18	3.00	**589.86**	3.00	589.95	21
R1	13.25	**1179.95**	13.73	1190.69	50	13.25	1180.20	13.70	**1189.28**	69
R2	5.36	**878.41**	5.61	**885.99**	87	5.27	878.46	5.64	886.60	109
RC1	13.00	1343.64	13.27	1363.61	40	12.88	**1340.53**	13.34	**1362.38**	54
RC2	6.25	**1004.20**	6.43	1014.06	73	6.38	1004.57	6.44	**1013.72**	88

[16] S. Jung and B. R. Moon. Toward minimal restriction of genetic encoding and crossovers for the 2D Euclidean TSP. *IEEE Trans. on Evolutionary Computation* accepted with minor revision.

[17] S. Jung and B. R. Moon. The natural crossover for the 2D Euclidean TSP. In *Genetic and Evolutionary Computation Conference*, pages 1003–1010, 2000.

[18] G. Kindervater and M. Savelsbergh. Vehicle routing: Handling edge exchanges. In E. Aarts and J. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 337–360. John-Wiley and Sons, Ltd., 1997.

[19] N. Kohl, J. Desrosiers, O. B. G. Madsen, M. M. Solomon, and F. Soumis. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33:101–116, 1999.

[20] G. Kontoravdis and J. Bard. A GRASP for the vehicle routing problem with time windows. *ORSA Journal on Computing*, 7:10–23, 1995.

[21] J. Larsen. *Parallelization of the Vehicle Routing Problem with Time Windows*. PhD thesis, Technical University of Denmark, 1999.

[22] I. Or. *Traveling Salesman-Type Combinatorial Problems and Their Relation to the Logistics of Regional Blood Banking*. PhD thesis, Northwestern University, 1976.

[23] J. Potvin and S. Bengio. The vehicle routing problem with time windows — part II: Genetic search. *INFORMS Journal on Computing*, 8:165–172, 1996.

[24] M. W. P. Savelsbergh. *Computer Aided Routing*. PhD thesis, Centrum voor Wiskunde en Informatica, 1988.

[25] M. W. P. Savelsbergh. The vehicle routing problem with time windows: Minimizing route duration. *ORSA Journal on Computing*, 4:146–154, 1992.

[26] M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987.

[27] M. Solomon and J. Desrosiers. Time window constrained routing and scheduling problems. *Transportation Science*, 22(1):1–13, 1988.

[28] K. Tan, T. Lee, K. Ou, and L. Lee. A messy genetic algorithm for the vehicle routing problem with time window constraints. In *IEEE Congress on Evolutionary Computation*, pages 679–686, 2001.

[29] S. Thangiah. An adaptive method using a geometrical shape for vehicle routing problems with time windows. In *International Conference on Genetic Algorithms*, pages 536–543, 1995.

[30] S. Thangiah. Vehicle routing with time windows using genetic algorithms. In Lance Chambers, editor, *Application Handbook of Genetic Algorithms: New Frontiers*, pages 253–277. CRC Press, 1995.

[31] S. Thangiah, K. Nygard, and P. Juell. GIDEON: A genetic algorithm system for vehicle routing with time windows. In *Seventh Conference on Artificial Intelligence Applications*, pages 322–328, 1991.

[32] P. Thompson and H. Psaraftis. Cyclic transfer algorithms for multi-vehicle routing and scheduling problems. *Operations Research*, 41(5):935–946, 1993.