Journal of the Society of Korea Industrial and Systems Engineering
Vol, 31, No. 2, pp.132－140, June 2008.

시간 제약을 갖는 차량 라우팅 문제에서 염색체 해석에 기초한 유전자 알고리듬과 부분 최적화 알고리듬

임동순 · 오현승[†]

한남대학교 공과대학 산업경영공학과

An Interpretation-based Genetic Algorithm and a Post Local Search Method for Vehicle Routing Problems with Time Windows

Dong-Soon Yim · Hyun-Seung Oh[†]

Department of Industrial and Management Engineering, Hannam University

본 논문은 시간 제약을 갖는 차량 라우팅 문제를 해결하기 위해 유전자 알고리듬과 부분 최적화 알고리듬을 적용한 방법을 소개한다. 유전자 알고리듬에서의 염색체는 노드를 나타내는 정수의 순열로 표현되어 직접적인 해를 나타내지 않지만, 경험적 방법에 의한 해석을 통해 유효한 해로 변형되도록 하였다. 유전자 알고리듬에 의해 생성된 주어진 수의 우수한 해들에는 세 부분 최적화 방법이 순차적으로 적용되어 보다 좋은 해를 생성하도록 하였다. 부분 최적화 방법들에 의한 해는 다시 유전자 알고리듬의 해로 바뀌지 않도록 하여 두 알고리듬은 느슨하게 연결되도록 하였다. 솔로몬의 데이터를 이용한 실험에서 본 연구에서의 방법이 모든 문제에 대해 우수한 해를 생성함을 나타내었다. 특히, 지금까지 알려진 가장 우수한 경험적 방법에 비교될 만한 결과를 가져옴을 보였다.

Keywords : Vehicle Routing Problem with Time Window(VRPTW), Genetic Algorithm(GA), Heuristic Algorithm

1. Introduction

The vehicle routing problem (VRP) has a long history in the field of operations research. Since Clark and Wright [5] proposed a "savings algorithm solution method, a number of methods have been developed to tackle the problem. Vehicle routing problem with time windows (VRPTW) as well as VRP are known as being NP-complete, i.e., an optimal solution of the problem cannot be obtained in a reasonable time. Although there have been significant works [9] toward for exact algorithms, almost all methods have been based on heuristics to solve real life problems. During the past decades, solution methods using meta-heuristics have received much attention. Tabu search (TS) [15], simulated annealing (SA) [12], and genetic algorithm (GA) [4] have been applied to VRP. While TS and SA have been successfully applied to VRP and VRPTW, GA has been considered inferior to other meta-heuristics. The main disadvantage of GA in VRP and VRPTW is its poor capability for chromosome representation of a solution. Path representation with delimiters and vehicle number representation will be natural since these explicit representations are directly decoded to a solution. A chromosome exactly represents the routes and sequence of node numbers in a

논문접수일 : 2008년 05월 09일　　논문심사일 : 2008년 06월 08일　　게재확정일 2008년 06월 08일
† 교신저자 dsyim@mail.hannam.ac.kr

route. However, maintaining the valid chromosomes which represent feasible solutions for satisfying vehicle capacity and time window constraints is not an easy task. To maintain feasibility in these chromosomes, a special procedure in constructing initial populations and genetic operators is required. These limitations hinder GA from taking full advantage of intrinsic GA features such as randomness and diversity. It is not surprising that recent papers show that GA can compete with other meta-heuristics. Its chromosome representations and decoding procedures are different from those of standard GA. Instead of explicit representation, it adopts implicit representations which are then interpreted to create feasible solutions through heuristics or specified optimization methods.

In Prins [14], a chromosome is represented by a sequence of node numbers to be visited. Since there is no delimiter in the chromosome representation, the shortest path method is used to interpret for each route of vehicles. In constructing a network to be solved by the shortest path method, constraints on vehicle capacity are considered. Therefore, their interpretation can be categorized as a route-first, cluster-second method. In Tan et al. [17], a chromosome represents a node sequence to be inserted into each route. Node numbers in a chromosome are selected one by one, and subsequently attached (rather than inserted) to each route. If no available route is found, a new route is created. In these GAs, standard crossover operators and mutation operators for permutation-based chromosome representation can be applied.

By recognizing that proper explicit chromosome representation is difficult, some authors have used the concept of solution individual instead of the chromosome in GA. According to Berger and Barkaoui [2, 3], a solution individual is a solution itself consisting of routes and a customer sequence in each route. To construct an initial population, a sequential insertion heuristic is applied. Customers are inserted at random in arbitrarily chosen insertion positions within routes. Then, re-initialization using another insertion procedure proposed by Liu and Shen [10] is used to improve the initial solution. Since no chromosome representations are employed, standard crossover and mutation operators do not apply. Instead, special heuristics are devised to create offspring and mutants.

Even though any of the GA methods, whether interpretation-based or solution individual-based, can improve the performance, the solution quality of GA alone is not satisfactory. GA appears to have the advantage of searching for a diverse solution space with proper genetic operators. However, it is difficult to reach a near optimal solution. To improve the solution quality, many studies incorporate local optimal procedure into GA. Several local search methods are used as mutation operators in the studies [2, 3, 13]. A local search including a λ-interchange method [13] is applied to the solution generated by the GA process in the studies [1, 17]. It should be noted that these methods result in a tightly coupled hybrid GA in that improved solutions by local search heuristics are encoded back to the chromosome format, which subsequently replaces the original one.

The main purpose of this paper is to illustrate the application of GA to VRPTW. To be effective and efficient, a standard GA must be modified and hybridized as previously stated. However, a standard representation of chromosomes is necessary in order to take advantage of GA properties. Since standard representation does not express a solution explicitly, an interpretation method will be required. Furthermore, the combination of local optimization techniques with GA is carefully considered to enhance the solution quality. This paper aims at developing an efficient and reliable solution method toward this aim.

2. Problem statement and notations

The VRPTW dealt in this paper is defined on the network $G(V, A)$ where $V=\{0, 1, 2, \cdots, n\}$ is a set of nodes, and $A=\{(i, j) : i \neq j\}$ is a set of arcs. The node 0 is the depot, and node $i(>0)$ is a customer. Every node i has the attributes of demand $d(i)$, time window $(e(i), l(i))$, and service duration $s(i)$. The arc (i, j) connecting from node i to node j has an attribute of travel time $t(i, j)$. It is assumed that an unlimited number of vehicles with the same capacity Q is available. The objective of the problem is to find the routes for vehicles in which total travel time is minimized.

Let a solution R for the problem consist of m routes. i.e., $R=\{r_1, r_2, \cdots, r_m\}$, where the route is a sequence of customers to be visited by a vehicle, i.e., $r_k=(v_{k,1}, v_{k,2}, \cdots, v_{k,n_k})$, $v_{k,j} \in V$. The total travel time is computed by considering the fact that a complete route starts and ends at the depot.

$$z(R)=\sum_{k=1}^{m}\left(t(0,v_{k,1})+\sum_{j=1}^{n_k-1}t(v_{k,j},v_{k,j+1})+t(v_{k,n_k},0)\right) \quad (1)$$

A solution is feasible if it satisfies the following constraints :

C1) Each customer is serviced at the same time.

C2) The total demand for all customers in a route does not exceed Q.

C3) The vehicle's arrival time at node i does not exceed $l(i)$.

The total demand of a route $r_k=(v_{k,1},v_{k,2},\cdots,v_{k,n_k})$ specified in the constraint C2 is computed simply by summing the demands of all customers in the route.

$$D(r_k)=\sum_{p=1}^{n_k}d(v_{k,p}) \quad (2)$$

Constraint C3 requires additional computations to determine the vehicle's arrival time at each node. Let $a(i)$ and $b(i)$ be the vehicle's arrival time and the vehicle's departure time at node i, respectively. From a route $r_k=(v_{k,1},v_{k,2},\cdots,v_{k,n_k})$ in the solution, the following relations hold under the assumption that the vehicle departs the depot at time 0.

$$a(v_{k,j})=\begin{cases}b(0)+t(0,v_{k,j}), & j=1\\ b(v_{k,j-1})+t(v_{k,j-1},v_{k,j}), & 2\le j\le n_k\end{cases} \quad (3)$$

$$a(0)=b(v_{k,n_k})+t(v_{k,n_k},0) \quad (4)$$

$$b(0)=0 \quad (5)$$

$$b(v_{k,j})=a(v_{k,j})+\max\{0,e(v_{k,j})-a(v_{k,j})\}+s(v_{k,j}),\ 1\le j\le n_k \quad (6)$$

The second term in the right hand side of equation (6) is the vehicle's wait time at $v_{k,j}$. When a vehicle arrives at the customer earlier than $e(v_{k,j})$, it should wait for $e(v_{k,j})-a(v_{k,j})$. By solving the above equations (3)~(6), the arrival time of every node is obtained.

To explain the algorithms developed in this paper, two special operators applied in the solution will be defined. Deleting the customer v from a solution R is defined as :

$$R-v=\{r_1,\cdots,r_{k-1},r_k',r_{k+1},\cdots,r_m\}, \quad (7)$$
$$\text{where } v_{k,p}=v \text{ and } r_k'=(v_{k,1},\cdots,v_{k,p-1},v_{k,p+1},\cdots,v_{k,n_k}).$$

Also, adding a customer v into p-th position of the k-th route of R is defined as :

$$[R+v]_{k,p}=\{r_1,\cdots,r_{k-1},r_k',r_{k+1},\cdots,r_m\}, \quad (8)$$
$$\text{where } r_k'=(v_{k,1},\cdots,v_{k,p},v,v_{k,p+1},\cdots,v_{k,n_k}).$$

3. Genetic algorithm based on heuristic interpretation

3.1 Representation and interpretation

The chromosome in this GA is represented as an integer string of length n : $(i_1,i_2,\cdots,i_n)$. Each number in the chromosome is a customer node. The sequence of integer numbers is not directly related to the path in routes. Instead, a heuristic-based interpretation transforms a sequence into a feasible solution.

```
Algorithm : GAInterpretation($(i_1,i_2,\cdots,i_n)$, m)
0  for each route $r_k$, $k=1,\cdots,m$
0.1  $r_k=(i_k)$
   end for
1  for each customer $i_j$, $j=m+1,\cdots,n$
1.1  for each route $r_k$, $k=1,\cdots,m$
1.1.1  for each position $p$ in $r_k$, $p=1,\cdots,n_k+1$
1.1.1.1 if $[R+i_j]_{k,p}$ violates the constraints C2 and C3,
          $T_{kp}=\infty$
1.1.1.2 else if $p$ is between two customers $a$ and $b$, compute
          $T_{kp}=t(a,i_j)+t(i_j,b)$
1.1.1.3  else if $p$ is the first position in the route followed by
          customer $b$, compute $T_{kp}=2\times t(i_j,b)$
1.1.1.4 else if $p$ is the last position in the route following customer
          $a$, compute $T_{kp}=2\times t(a,i_j)$
               end if
               end if
               end if
            end if
         end for
      end for
1.2  if $T_{kp}=\infty$ for all $k$ and $p$, then $m=m+1$  $r_m=(i_j)$
1.3  else $R=[R+i_j]_{k^*,p^*}$ where $T_{k^*p^*}=Min\{T_{kp},\text{for all }k\text{ and }p\}$
      end if
   end for
```

The basic process in GAInterpretation is similar to the insertion heuristic (Solomon, 1987) with a minimum total time traveled. In step 0, m routes are initialized with the first m customers in the chromosome. In step 1, the remaining customers in the chromosome are inserted into the current routes one by one. The route and the position in the route to insert are determined by examining all the possible alternatives. Among them, one with the minimum value of T_{kp}, computed in steps 1.1.1.2, 1.1.1.3, and 1.1.1.4, is selected. Even though the computed value is not directly related to the total time traveled, it helps to construct a good cluster for each route. When a position and route for a customer to be inserted are unavailable, a new route is created (step 1.2). The complexity of the interpretation method is $O(n^2)$.

In this interpretation algorithm, the initial value of m and the first m customers play an important role in constructing a solution. In selecting a route in which to insert each of the remaining $n-m$ customers, the route which contains adjacent customers is preferred according to the insertion criteria specified in step 1.3. Therefore, the solution generated by the algorithm is sensitive to the first customer assigned to each route. When an appropriate number of initial customers is assigned to each route, a good solution showing the adjacent customers that are clustered in each route will be generated. This also implies that the appropriate value of m should be given. If m is given with too large a value, the solution will include extra routes with too few customers. This inhibits a good solution. Conversely, if m is too small, the diversity of the solution may not be guaranteed. There will be high possibility that different chromosomes will be interpreted to the same solution.

3.2 Initial Population

The initial population is created on a random basis, i.e., the position for every integer number in a chromosome is selected randomly.

3.3 Selection

In each generation of the GA process, a number of excellent chromosomes are selected for new population. The selection is based on a roulette-wheel method. According to the fitness value of a chromosome, a good chromosome has a high probability of being selected. The fitness value of a chromosome is the total distance traveled of an interpreted solution as defined in equation (1).

3.4 Operators

Since the chromosome representation is the same as the path representation in TSP (Traveling Salesman Problem), standard genetic operators can be applicable. For the TSP, PMX [7], OX [6], and CX [11] crossover operators are commonly used. Among them, PMX was considered in this paper. In implementing PMX, each chromosome in the population is selected for crossover with a probability P_c. Then, selected chromosomes are mated randomly. For each pair of coupled chromosomes, the PMX operator is applied.

In addition to crossover, an inversion operator was considered. A chromosome in the population undergoes inversion with the probability P_i. Then, two nodes in the considered chromosome are selected randomly the positions of both nodes are then switched.

4. Local search methods

To improve the solution of GA, three improving heuristics were implemented : 1-relocate, 2-relocate, and 2-opt. 1-relocate simply relocates a customer to a certain position in a route. The new position may be another position in the same route or a position in another route. 2-relocate is similar to λ-interchange [13], where λ is one. For every pair of two nodes in different routes, these nodes are relocated to the other route rather than interchanged. 2-opt is a well known heuristic in TSP.

4.1 Local search based on 1-relocate

```
Algorithm : 1RLocalSearch ( $R$ )
0     flag_1R = true task_1R = false
1     while (flag_1R = true) do
1.1     flag_1R= false
1.2     for each customer $i$, $i = 1, 2, \cdots, n$
1.2.1     $R^{(1)} = R - i$
1.2.2     find $k*$ and $p*$ such that
              $z([R^{(1)}+i]_{k*,p*}) = \min\{z([R^{(1)}+i]_{k,p})$, for all $k$ and $p\}$
1.2.3     if $z([R^{(1)}+i]_{k*,p*}) < z(R)$, then $R = [R^{(1)}+i]_{k*,p*}$
            flag_1R = true; task_1R = true
          end if
```

```
    end for
  end while
2 return task_1R
```

The algorithm 1RLocalSearch is based on the 1-relocate heuristic, and improves the solution R. In step 1.2, every customer undergoes relocation. The best position and route to be relocated is determined in steps 1.2.1 and 1.2.2. Then, the relocation is performed only if solution R is improved. Step 1.2 is repeated when any customer has been relocated. If no customer has been relocated, the algorithm returns false (step 2). Given that the complexity of step 1.2.2 is $O(n)$, the time complexity of step 1.2 is $O(n^2)$.

4.2 Local search based on 2-relocate

Algorithm : *2RLocalSearch* (R)

```
0 flag_2R = true; task_2R = false
1 while (flag_2R = true) do
1.1   flag_2R = false
1.2   for each customer $i$, $i = 1, 2, \cdots, n-1$
1.2.1     foreach customer $j$, $j = i+1, i+2, \cdots, n$
1.2.1.1      $k1$ is index of route in which customer $i$is included
1.2.1.2      $k2$ is index of route in which customer $j$is included
1.2.1.3      if $k1 = k2$, continue
             end if
1.2.1.4      $R^{(1)} = R - i$
1.2.1.5      $R^{(2)} = R^{(1)} - j$
1.2.1.6      $R^{(3)} = [R^{(2)} + i]_{k2, p^*}$ where
               $z([R^{(2)} + v]_{k2, p^*}) = \min\{z([R^{(2)} + v]_{k2, p})$, for all $p\}$
1.2.1.7      $R^{(4)} = [R^{(3)} + j]_{k1, p^*}$ where
               $z([R^{(3)} + v]_{k1, p^*}) = \min\{z([R^{(3)} + v]_{k1, p})$, for all $p\}$
1.2.1.8      if $z(R^{(4)}) < z(R)$, then $R = R^{(4)}$ flag_2R = true
                  task_2R = true
             end if
        end for
    end for
  end while
2 return task_2R
```

The algorithm 2RLocalSearch relocates two customers at the same time. For every pair of customers in different routes, the relocation of these customers to the other routes is considered. Through steps 1.2.1.4 to 1.2.1.7, the total time traveled is computed as a result of relocations. At first, two customers are removed from the current solution. Then each customer is inserted to the best position in another route. If the relocations decrease the total time traveled, a new solution is generated. The complexity of step 1.2 is $O(n^3)$. Step 1.2 is repeated while the relocation succeeds. If no relocation is accomplished, false is returned in step 2.

4.3 Local search based on 2-opt

Algorithm : *2OPTLocalSearch* (R)

```
0 flag_opt = true; task_opt = false
1 while (flag_opt = true) do
1.1   flag_opt = false
1.2   for each route $r_k$, $k = 1, \cdots, m$
1.2.1     perform 2-opt procedure
1.2.2     If reconstruction of the route occurs, flag_opt = true;
                                  task_opt = true
          end if
      end for
  end while
2 return task_opt
```

In 2-opt procedure, alternatives are considered by deleting two edges and creating two new edges in a route. Among these alternatives, a new route giving the best improvement is selected. This process is applied for all routes of a solution, then repeated as long as improvement occurs. The complexity of 2-opt is known as $O(n^2)$.

4.4 Sequential execution of local search methods

The local search methods described so far can be sequentially executed in order to gain greater improvement.

Algorithm : *SequentialLocalSearch* (R)

```
1 flag = flag_1R = falg_2R = flag_opt = true
2 while (flag = true) do
2.1   flag_1R = 1RLocalSearch $(R)$
2.2   if(flag_opt = false) and (flag_1R = false), flag = false; return
      end if
2.3   flag_2R = 2RLocalSearch $(R)$
2.4   if(flag_1R = false) and (flag_2R = false), flag = false; return
      end if
2.5   flag_opt = 2OPTLocalSearch $(R)$
2.6   if(flag_2R = false) and (flag_opt = false), flag = false; return
      end if
  end while
```

In steps 2.1 through 2.5 of the above algorithm, the three local search methods are sequentially executed. These steps are repeated until there is no more improvement in the solution. The algorithm takes a solution from GA as an input, then produces a local optimal solution. To achieve significant improvement, more than one local optimal solution might be obtained with a number of initial solutions generated from GA. Then, the best solution is selected from among the obtained local optimal solutions. However, a large amount of computation time will be required because the number of local optimal solutions will increase. It is impossible to foresee how many iterations of each local search method will be executed to obtain a local optimal solution. If these local search methods are executed with many iterations, the large computation time will be a barrier in spite of the improvement in solution quality. Therefore, the number of local optimal solutions should be determined by considering the trade-off between computation time and solution quality.

According to the type of chromosome representation mechanism, an improved solution by local search methods may be encoded back to the chromosome format. Since the chromosome in our GA is represented implicitly and interpreted by a heuristic, however, an improved solution cannot easily be encoded back. In addition, the algorithm SequentialLocalSearch is executed after the whole GA process is complete. Thus, the GA and local search methods are loosely coupled such that the chromosome structure is not influenced by a local search.

5. Computational results

The GA and local search methods described in this paper were coded with JAVA programming language. Then, a number of experiments with Solomon's instances [15] were conducted using a Pentium-4 3.2 GHz PC. The Solomon's data includes six sets : R1, R2, C1, C2, RC1, and RC2. Each of 56 problems has 100 customers. The problem sets R1, C1, and RC1 are designed to have a narrow scheduling horizon. Hence, short routes in which only a few customers are served by the same vehicle will be generated. Conversely, problem sets R2, C2, and RC2 have large scheduling horizon, and more customers can be served by the same vehicle. The problems belonging to the C categories are clustered data; customers are clustered together geographically or in terms of time windows. The problems from the R categories are uniformly distributed data and those from the RC categories are hybrid problems having features of both C and R categories.

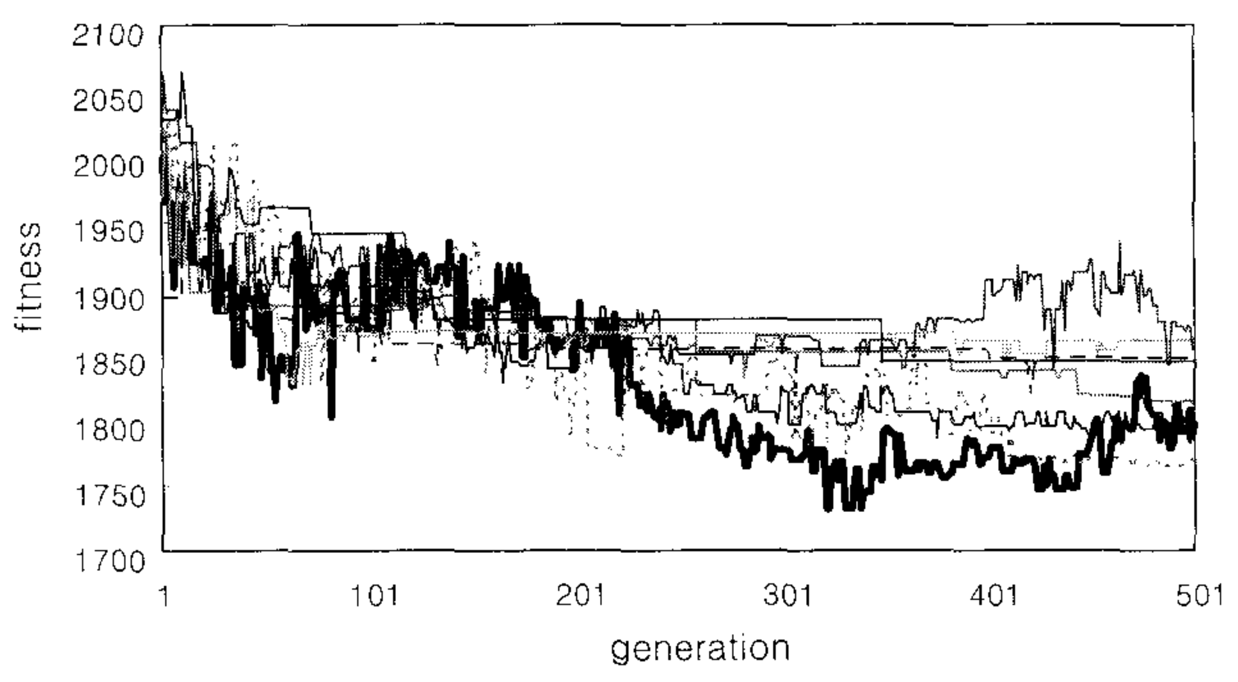
5.1 GA parameters

In order to determine the appropriate values of GA parameters, several alternatives for parameters were examined with Solomon's data sets. Judgments concerning GA parameters were based on the following criteria :

1. Diverse and high-quality solutions are produced within total generations
2. The computation time is as small as possible

<Figure 1> shows an instance of results accomplished with a problem of Solomon's data sets. Several trends of GA's best solutions produced in each generation are shown. Each trend was generated with specific values of GA parameters. From the figure, it is inferred that the trend with the bold line generates more promising solutions than the others. Some other trends demonstrate premature convergence toward bad solutions or fluctuation in the range of bad solutions. Therefore, GA parameter values used in the bold line trend will be preferred. <Table 1> shows parameter values determined from this kind of selection process performed with Solomon's data sets.

As stated in Section 3.1, the interpretation algorithm GAInterpretation requires the initial number of routes,



<Figure 1> Trends of best solutions

<Table 1> GA parameters

Parameter	Value
Population size	100
No. of generations	500
Crossover probability	0.4
Inversion probability	0.1

m. In this experiment, this initial value was given as the minimum number of routes, and calculated by considering all the customer demands and vehicles' capacity, i.e., $m = \sum_{i=1}^{n} D(i) / Q$.

5.2 Results of Solomon's data set

The performance of the proposed method was evaluated with 56 problems in Solomon's data sets. For GA, the parameter values were set as specified in Table 1. In addition, the number of GA solutions was set to 100 by considering solution quality and computation time. Among all the solutions generated in GA, therefore, the 100 best solutions were stored. After GA, a sequential local search method (1R, 2R, 2OPT) was applied to every 100 solutions. Finally, the best solution among the 100 local optimal solutions was selected as a final solution. To provide reliable results, 5 runs for each problem were executed.

To investigate the performance of each local search algorithm and the effect of sequential execution of several search methods, extensive experiments were performed. Among three local search algorithms, 1R shows the best performance for all categories. Improvement over initial solutions lies between 30% (R1 problems) and 50% (C2 problems) has been achieved by 1R. Improvement ratio of 2R is lower than 1R in spite of more computation time. 2OPT shows the worst performance : negligible improvement with small computation time. When two local search algorithms are executed sequentially, the combination of 1R and 2R shows the best performance. Finally, the combination of three local search algorithms shows slightly better improvement than (1R, 2R). For instance, improvement ratio for R2 category increased by 1% compared with (1R, 2R).

The computational results from <Table 2> to <Table 7> show average fitness value and average CPU time, as well as the minimum fitness value for each problem. Also, the average improvement ratio (Imp) compared with the best known heuristic is included.

<Table 2> Result of R1 problem sets

Problem	Best[1]	GA with post-local search		
		Ave[2](Imp[3])	CPU[4]	Min[5]
R101	1645.79	1666.32(-1.25%)	156	1653.98
R102	1486.12	1500.89(-0.99%)	182	1487.02
R103	1292.68	**1249.25(3.36%)**	191	**1235.74**
R104	1007.24	1026.75(-1.94%)	210	1016.57
R105	1377.11	1415.55(-2.79%)	177	1392.15
R106	1251.98	1294.59(-3.40%)	200	1267.54
R107	1104.66	1115.57(-0.99%)	205	**1103.76**
R108	960.88	978.16(-1.80%)	225	**959.86**
R109	1194.73	1208.09(-1.12%)	196	**1174.21**
R110	1118.59	1143.27(-2.21%)	212	**1115.84**
R111	1096.72	1111.03(-1.30%)	217	1096.99
R112	982.14	1006.68(-2.50%)	230	996.91
Ave		(-1.41%)	200	

주) [1] Best : fitness value of best known heuristic.
[2] Ave : average fitness value.
[3] Imp = (Best - Ave)/Best.
[4] CPU : average CPU time in seconds.
[5] Min : minimum fitness value.

<Table 3> Result of R2 problem sets

Problem	Best	GA with post-local search		
		Ave(Imp)	CPU	Min
R201	1252.37	1262.39(-0.80%)	347	**1243.60**
R202	1191.70	**1104.49(7.32%)**	382	**1075.11**
R203	939.54	957.83(-1.95%)	470	944.01
R204	825.52	**776.86(5.89%)**	505	**766.30**
R205	994.42	1056.69(-6.26%)	402	1021.93
R206	906.14	972.15(-7.29%)	451	924.80
R207	893.33	**862.98(3.40%)**	514	**836.35**
R208	726.75	741.03(-1.97%)	506	**719.57**
R209	909.16	918.57(-1.04%)	442	**892.47**
R210	939.34	952.27(-1.38%)	447	**937.10**
R211	829.71	835.92(-0.75%)	491	**819.33**
Ave		(-0.44%)	451	

<Table 4> Result of C1 problem sets

Problem	Best	GA with post-local search		
		Ave(Imp)	CPU	Min
C101	828.94	828.94(-0.00%)	140	828.94
C102	828.94	828.94(0.00%)	150	828.94
C103	828.06	847.40(-2.34%)	160	828.06
C104	824.78	840.58(-1.92%)	177	824.78
C105	828.94	828.94(0.00%)	140	828.94
C106	828.94	828.94(0.00%)	140	828.94
C107	828.94	828.94(0.00%)	139	828.94
C108	828.94	835.03(-0.73%)	158	828.94
C109	828.94	828.94(0.00%)	167	828.94
Ave		(-0.55%)	153	

<Table 5> Result of C2 problem sets

Problem	Best	GA with post-local search		
		Ave(Imp)	CPU	Min
C201	591.56	591.56(0.00%)	292	591.56
C202	591.56	591.56(0.00%)	336	591.56
C203	591.17	598.47(-1.23%)	377	591.17
C204	590.6	606.70(-2.73%)	483	597.46
C205	588.88	588.88(0.00%)	335	588.88
C206	588.49	588.49(0.00%)	361	588.49
C207	588.29	588.29(0.00%)	371	588.29
C208	588.32	588.32(0.00%)	372	588.32
Ave		(-0.50%)	366	

<Table 6> Result of RC1 problem sets

Problem	Best	GA with post-local search		
		Ave(Imp)	CPU	Min
RC101	1696.94	**1686.99(0.59%)**	166	**1655.88**
RC102	1554.75	**1516.41(0.47%)**	178	**1494.81**
RC103	1261.67	1321.95(-4.78%)	186	1281.92
RC104	1135.48	1174.33(-3.42%)	187	1138.70
RC105	1629.44	**1592.30(2.28%)**	177	**1573.99**
RC106	1424.73	1431.84(-0.50%)	181	**1423.37**
RC107	1230.48	1271.78(-3.36%)	189	1250.51
RC108	1139.82	1164.48(-2.16%)	198	**1135.55**
Ave		(-1.11%)	183	

<Table 7> Result of RC2 problem sets

Problem	Best	GA with post-local search		
		Ave(Imp)	CPU	Min
RC201	1406.91	1454.80(-3.40%)	345	**1403.00**
RC202	1367.09	**1230.01(10.03%)**	390	**1199.46**
RC203	1049.62	**1018.57(2.96%)**	431	**993.66**
RC204	798.41	826.05(-3.50%)	489	806.07
RC205	1297.19	1303.05(-0.45%)	338	**1281.23**
RC206	1146.32	1176.20(-2.61%)	441	**1141.57**
RC207	1061.14	**1040.53(1.94%)**	435	**1004.13**
RC208	828.14	912.33(-10.17%)	483	889.48
Ave		(-0.65%)	419	

The results show that the average improvement ratio over known best solutions by heuristics for each category ranges in (-1.41%, -0.44%). Since the improvement ratio in all categories is less than zero, the proposed method appears slightly inferior to the best known heuristics. However, the average fitness values for each problem show that the proposed method generated better solutions than the best-known heuristics in 10 problems out of 56 (1, 3, 3, and 3 problems of R1, R2, RC1, and RC2 respectively). For 12 problems from the C1 and C2 categories, our method generates the same results as the best-known heuristics. Also, 24 new best solutions were produced during 5 runs for each problem. These results demonstrate that the proposed method can compete with the best-known heuristics.

The computation time for each problem set shows different patterns. While the average computation time for the short route problem sets, R1, C1, and RC1 is 200 sec, 153 sec, and 183 sec respectively, the long route problems, R2, C2, and RC2 require 451 sec, 366 sec, and 419 sec. The computation time somewhat depends on the number of vehicles determined in a solution method. As the number of vehicles decreases, i.e., the number of customers in a route increases, the computation time increases mainly due to the greater amount of computation time by a local search. Considering that long a route requires a lot of time to examine time window constraints, it is a natural result in the case of VRPTW.

5. Conclusion

Considering that transportation and distribution networks nowadays require more cost effective solutions ever before, the vehicle routing problem has been recognized as one of the most important problems to be solved. Though many heuristic-based solution methods have been proposed to solve practical problems in a reasonable amount of time, there still remains a difficulty in providing quality solutions.

In this paper, the combination of a genetic algorithm and a post-local search method for VRPTW was discussed. A chromosome in this GA is interpreted to a feasible solution using a heuristic method. To provide better solutions, three local search methods are sequentially applied to a number of GA solutions. The computation results with Solomon's data sets showed that the proposed method generates excellent solutions within 4 minutes in short route problems and 8 minutes in long route problems. Compared with the best-known solutions by heuristics, the average performance of our method was equal or superior in 22 out of 56 problems. Additionally, it produced 24 new best solutions from 5 runs for each problem.

References

[1] Baker, B. M. and Ayechew, M. A.; "A genetic algorithm for the vehicle routing problem," *Computers and Operations Research*, 30 : 787-800, 2003.

[2] Berger, J. and Barkaoui, M. A.; "parallel hybrid genetic algorithm for the vehicle routing problem with time window," *Computers and Operations Research*, 31 : 2037-2053, 2004.

[3] Berger, J. and Barkaoui, M. A.; "new hybrid genetic algorithm for the capacitated vehicle routing problem," *Journal of Operations Research Society*, 54 : 1254-1262, 2004.

[4] Blanton, J. L. and Wainwright, R. L.; "Multiple vehicle routing with time and capacity constraints using genetic algorithms," *Proceedings of the fifth International Conference on Genetic Algorithms*, Morgan Kaufmann : San Francisco, 452-459, 1993.

[5] Clark, G. and Wright, J. W.; "Scheduling of vehicles from a central depot to a number of delivery points," *Operations Research*, 12 : 568-581, 1964.

[6] Davis, L.; "Applying Adaptive Algorithms to epistatic domains," *Proceedings of the International Joint Conference on Artificial Intelligence*, 162-164, 1985.

[7] Goldberg, D. E. and Lingle, R. Alleles; "Loci, and the TSP," *Proceedings of the First International Conference on Genetic Algorithms*, Alwrence ErlBaum Associates, Hillsdale, NJ, 154-159, 1985.

[8] Jones, D. R. and Beltramo, M. A.; "Solving partitioning problems with genetic algorithms," *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, Los Altos, CA, 442-449, 1991.

[9] Kallehauge, Brian; "Formulations and exact algorithms for the vehicle routing problem with time windows," *Computers and Operations Research*, 35(7) : 2307-2330, 2008.

[10] Liu, F. H. and Shen, S. Y.; "A route-neighborhood-based metaheuristic for vehicle routing problem with time windows," *European Journal of Operations Research*, 118 : 485-504, 1999.

[11] Oliver, I. M., Smith, D. J., and Holland, J. R. C.; "A study of permutation crossover operators on the Traveling Salesman Problem," *Proceedings of the First International Conference on Genetic Algorithms*, Alwrence Erl-Baum Associates, Hillsdale, NJ, 224-230, 1985.

[12] Osman, I. H.; "Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem," *Operations Research*, 41 : 421-451, 1993.

[13] Osman, I. H. and Christofides, N.; "Capacitated clustering problem by hybrid simulated annealing and tabu search," *International Transaction in Operations Research*, 1(3) : 317-336, 1994.

[14] Prins, C. A.; "simple and effective evolutionary algorithm for the vehicle routing problem," *Computers and Operations Research*, 31 : 1985-2002, 2004.

[15] Rochat, Y. and Taillard, E.; "Probabilistic diversification and intensification in local search for vehicle routing," *Journal of Heuristics*, 1 : 147-167, 1995.

[16] Solomon, M. M.; "Algorithm for the vehicle routing and scheduling problems with time window constraints," *Operations Research*, 35 : 254-265, 1987.

[17] Tan, K. C., Lee, L. H., and Ou, K.; "Artificial intelligence heuristics in solving vehicle routing problems with time window constraints," *Engineering Applications of Artificial Intelligence*, 14 : 825-837, 2001.

[18] Tangiah, S. R, Vinayagamoorthy, and Gubbi, A.; "Vehicle routing with time deadlines using genetic and local algorithms," *Proceedings of the Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann, NY, 506-513, 1993.