ELSEVIER

Int. J. Production Economics 73 (2001) 175–188

international journal of
production
economics

www.elsevier.com/locate/dsw

A hybrid genetic algorithm for the vehicle scheduling problem with due times and time deadlines☆

Yang-Byung Park*

Industrial Engineering Division, College of Mechanical and Industrial System Engineering, Kyung Hee University, 1 Seocheon, Kihung, Yongin-Si, Kyunggi-Do 449-701, South Korea

Received 2 May 2000; accepted 9 December 2000

Abstract

In this paper, I propose a hybrid genetic algorithm (HGAV) incorporating a greedy interchange local optimization algorithm for the vehicle scheduling problem with service due times and time deadlines where three conflicting objectives of the minimization of total vehicle travel time, total weighted tardiness, and fleet size are explicitly treated. The vehicles are allowed to visit the nodes exceeding their service due times with a penalty, but within their latest allowable times. The HGAV employs a mixed farming and migration strategy along with a variant of partially mapped crossover and several mutation operators newly developed while maintaining the solution feasibility during the whole evolutionary process. The HGAV is extensively evaluated on the various types of test problems, including a sensitivity analysis with varied genetic parameters on the weighted sum of three objectives of a solution. The results show that the HGAV always attains better solutions than the BC-saving algorithm, but with a much longer computation time.

Keywords: Vehicle scheduling problem; Service due times; Service time deadlines; Hybrid genetic algorithm; Sensitivity analysis

1. Introduction

The vehicle scheduling problem involves determining minimum cost vehicle schedules for a fleet of vehicles originating and terminating from a central depot. The vehicles service a set of nodes with vehicle capacity and travel time constraints. All nodes must be assigned to vehicles such that each node is serviced exactly once and each vehicle cannot violate the given constraints. The vehicle scheduling and routing problem has been extensively studied. Bodin et al. [1], Golden and Assad [2], Soloimon [3], Laporte [4], and Anily [5] provide a comprehensive survey of vehicle scheduling and routing problem and several variations.

During the past few years, several studies have reported their experiences about applying genetic algorithms to the vehicle scheduling and routing problem with time and capacity constraints. The obvious advantages of genetic algorithms include ease of implementation, emphasis on global as well as local search, use of randomization in search process, and interactive feature.

☆This work was supported by Grant No. 981-1014-075-1 from the Basic Research Program of the KOSEF.

* Corresponding author. Tel.: + 82-31-201-2553; fax: + 82-31-203-4004.

E-mail address: ybpark@nms.kyunghee.ac.kr (Y.-B. Park).

0925-5273/01/$ - see front matter
PII: S0925-5273(00)00174-2

Gabbert et al. [6] presented a genetic algorithm approach to learning low-cost routes and schedules for a large rail freight transportation network. Thangiah et al. [7] developed a kind of cluster-first route-second heuristic for the vehicle routing problem with time deadlines. They used a genetic algorithm to cluster nodes. Blanton and Wainwright [8] developed two new crossover operators to apply an order-based genetic algorithm to the vehicle routing problem with time windows. Thangiah [9] used a genetic algorithm to search for the attributes of a set of circles, corresponding to the number of vehicles, for clustering nodes in the vehicle routing problem with time windows.

Cheng and Gen [10] proposed a hybrid genetic algorithm to solve the fuzzy vehicle routing and scheduling problem where fuzzy due time is used to represent the grade of satisfaction for the service time at each node. They designed an insertion heuristic-based crossover operator to find out improved offspring. Potvin et al. [11] used a genetic algorithm to find good parameter settings in the route construction phase of a parallel insertion heuristic for the vehicle routing problem with time windows. Malmborg [12] applied a genetic algorithm to a service-level-based vehicle routing problem where vehicles serve the material flow requirements between a collection of work centers over a fixed operating period and the objective is to minimize the delay associated with material transfer. Chen et al. [13] modeled the rolling batch problem in iron and steel industry as the vehicle routing problem with time windows and solved it using genetic algorithm.

Genetic algorithms have proven to be a versatile and effective approach for solving the vehicle routing and scheduling problem. Nevertheless, there are many situations in which the simple genetic algorithm does not perform well particularly due to a premature convergence to a local optimization. One of the most common approaches to resolve the premature convergence is to employ a hybrid genetic algorithm. In the hybrid approaches, genetic algorithms are used to perform global exploration among a population while heuristic methods are used to perform local exploitation around chromosomes. Because of the complementary properties of genetic algorithms and conventional heuristics, the hybrid approach often outperforms either method operating alone and so many applications are strongly in favor of the hybrid approach [14,15].

In many vehicle scheduling, the vehicles are allowed to visit the nodes exceeding the given service due times with a penalty, but within time deadlines (or latest allowable service times), and the vehicles that arrive at the nodes earlier than due times serve them immediately. Hence, optimization of vehicle scheduling involves not only total vehicle travel time and the number of vehicles required, but total weighted tardiness occurring when the vehicles arrive at the nodes after due times. This type of vehicle scheduling problem is termed a vehicle scheduling problem with due times and time deadlines, or VSPDT. The VSPDT arises in a wide array of practical decision making problems. Instances of the VSPDT occur in retail distribution, school bus routing, industrial refuse collection, after-service (A/S) visit, mail delivery, dial-a-ride service, freight operations, etc. Efficient scheduling of vehicles can save a significant amount of logistics cost and improve the customer service level.

In this paper, I propose a hybrid genetic algorithm (HGAV) incorporating a greedy interchange local optimization procedure to solve the VSPDT where three conflicting objectives of the minimization of total travel time, total weighted tardiness, and fleet size are explicitly considered. The hybrid genetic algorithm developed is extensively evaluated on various types of test problems, including a sensitivity analysis with varied genetic parameters on the weighted sum of three objectives.

2. HGAV

2.1. Procedure of HGAV

The HGAV guarantees the generation of feasible solutions in its whole evolutionary process by using the special solution coding and applying the repair process to correct any infeasible solutions generated. The HGAV also employs an elitist strategy to report the best solution found during the evolutionary process. The HGAV adopts a mixed farming and migration strategy [16] that is being implemented in designing a "virtual parallel computer",

in order to avoid a premature convergence to a local optimum and, at the same time, to maintain a high survival probability for the chromosomes with dominant elements (or genes). The basic idea of this strategy is to divide initial population into several sub-populations and apply a genetic algorithm to each of them while performing a local optimization periodically to some best chromosomes collected from them.

The procedure of HGAV is described as follows:

Step 1: Divide the initial population into l sub-populations with the same number of chromosomes each.

Step 2: Apply the crossover and mutation to the chromosomes in each sub-population, and then calculate their fitness values. If the current generation number is a multiple of m, then move to Step 3. Otherwise, move to Step 4. m is an integer value representing an interval of local optimization to be performed in Step 3.

Step 3: Select the best n different chromosomes from each sub-population and apply a local optimization technique to all of them (a total of $l \cdot n$ chromosomes). Then, substitute the current best n different chromosomes in each sub-population by the best n different chromosomes obtained by applying a local optimization technique.

Step 4: If a termination condition is satisfied, then report the best chromosome found so far and stop. Otherwise, apply a roulette wheel selection method [17] to each sub-population and return to Step 2.

2.2. HGAV Parameters

2.2.1. Solution coding

A solution is coded through a diploid structure (or two strings). The first string includes all nodes in any order except depot. The second string has the same number of fields as the first one and it denotes the vehicle numbers assigned to the nodes at the corresponding fields in the first string. The example below represents three vehicles whose routes are 0-3-9-1-10-7-0, 0-2-4-5-6-0 and 0-8-0, where 0 denotes a central depot.

A_1	3	9	2	1	4	10	8	5	6	7
A_2	1	1	2	1	2	1	3	2	2	1

2.2.2. Initialization

Because of the existence of time and capacity constraints, it is impossible to use a simple random procedure to generate the initial population. In the proposed initial procedure, the first string of a chromosome is constructed by generating node numbers randomly and the second string is constructed by assigning vehicle numbers to the corresponding nodes sequentially from the leftmost field. Vehicle number 1 is always assigned to the first node in the first string.

To determine the vehicle number for a node, a feasible insertion position for the node is searched on the partial routes starting from vehicle number 1 in ascending order. It is always required to examine the route feasibility whenever a node insertion is considered. If the first feasible insertion position for the node is found, then the node is moved to the field of its insertion position in the first string and the corresponding vehicle number is assigned to the same field in the second string. Then, the node and vehicle numbers after the insertion position are moved backward by one field in each string. If no feasible insertion position for the node is found, a new vehicle number is assigned to it at the current field.

2.2.3. Fitness evaluation

Fitness is calculated based on the three objective functions. The weighted sum of three objectives for a chromosome is the combination of the normalized total travel time, the normalized total weighted tardiness, and the normalized fleet size. The weighted sum of three objectives is computed by

$$f(S_k) = \left(w_1 \frac{t_k}{t^0_{\max}} + w_2 \frac{d_k}{d^0_{\max}} + w_3 \frac{v_k}{v^0_{\max}} \right) 100,$$

$$\sum_{i=1}^{3} w_i = 1 \text{ and } w_i \geqslant 0,\ i = 1, 2, 3, \qquad (1)$$

where w_1 is the weight of the minimization of the total travel time, w_2 the weight of the minimization of total weighted tardiness, w_3 the weight of the minimization of the fleet size, $t^0_{\max}$ the maximum total travel time among the initial chromosomes, $d^0_{\max}$ the maximum total weighted tardiness among the initial chromosomes, $v^0_{\max}$ the maximum fleet size among initial chromosomes, t_k the total travel

time of chromosome k, d_k the total weighted tardiness of chromosome k, and v_k the fleet size of chromosome k.

Based on the weighted sum of three objectives, the relative fitness of a chromosome used in the selection process (Step 4 of HGAV) for reproduction is calculated by

$$r(S_k) = \frac{f(S^k_{\max}) - f(S_k)}{f(S^k_{\max}) - f(S^k_{\min})}, \tag{2}$$

where $f(S^k_{\max})$ is the maximum weighted sum of objectives among chromosomes in sub-population of chromosome k, and $f(S^k_{\min})$ is the minimum weighted sum of objectives among chromosomes in sub-population of chromosome k. Eq. (2) expresses the relative difference among chromosomes in sub-population better than (1), particularly when the difference between $f(S^k_{\max})$ and $f(S^k_{\min})$ is small.

2.2.4. Local optimization

For a local optimization, a greedy interchange algorithm introduced by Dror and Levy [18] is applied with a minor modification. The local optimization procedure consists of two phases. In the first phase, a two-node and a two-arc interchange are applied to each route in a chromosome. Whenever an interchange is considered, the route feasibility is tested. The weighted saving from either the two-node or two-arc interchange is given by

$$\text{sav} = w_1 \frac{\nabla t}{t^0_{\max}} + w_2 \frac{\nabla d}{d^0_{\max}}, \tag{3}$$

where ∇t is the saving in travel time from two-node (arc) interchange, and ∇d the saving in weighted tardiness from the two-node (arc) interchange.

Lemma 1. *When an interchange of two arcs* (i, j) *and* (k, l) *where* $j < k$ *is attempted, all nodes following node* l *on the route after interchange are served before time deadlines if the new arrival times to the nodes between* i *and* l *on the route before interchange do not violate time deadlines and the new arrival time to* l *is decreased.*

In the second phase, a one-node and a two-node interchange are applied to all the pairs of routes in a chromosome. The candidate nodes for interchange must satisfy the individual route constraints as well. The weighted saving from one-node interchange is computed by adding $w_3 \nabla v / v^0_{\max}$ to Eq. (3) where ∇v denotes the saving in fleet size from one-node interchange.

Lemma 2. *When a one-node interchange between a route of* i *and* j, *where* $i \ll j$, *and a route of* k, r, *and* l, *where* $k \ll r \ll l$, *is attempted with node* r:

(i) *all the nodes following node* j *on the route after interchange are served before time deadlines if* $t_{ir} + t_r - t_{ij} \leqslant 0$, *where* t_{ir} *represents a travel time from* i *to* r *and*

(ii) *the new arrival times to the nodes following* j *after interchange are given by* $a'_u = a_u + \delta$, *where* a_u *denotes the arrival time to node* u *before interchange and* $\delta = t_{ir} + t_r - t_{ij}$.

2.2.5. Crossover

A variant of the partially mapped crossover (PMX) [19] is used. The variant is required to consider the multiple vehicles and to include the repair process for infeasible solutions. The crossover is initiated by choosing two random cut points of parents and swaps the segments before the first cut points of parents with the nodes and corresponding vehicle numbers at the same time. Next, it copies the segments between two cut points of each parent into each offspring. Finally, it swaps the segments after the second cut points of parents.

When there is a conflict for node number in filling the offspring after the first cut point, the node and vehicle number under consideration are replaced by other ones based on the predefined mappings. When there is infeasible node in filling the offspring after the first cut point, its vehicle is replaced by the first feasible one while considering the vehicles in ascending order.

2.2.6. Mutation

Since a chromosome represents multiple vehicle routes simultaneously, normal mutation may cause random breaking, sub-tours, unnecessary increase of fleet size, and repetitions of nodes, resulting in loss of the optimality information gained by genetic algorithm over the previous generations. Thus, I propose a set of mutation operators that

overcomes the defects of normal mutation in genetic algorithm. The mutation operators utilize the route information of a chromosome and they are as follows.

(i) *Merge mutation*: Select one node randomly in the route whose number of nodes is less than three and change its vehicle number to one of the other vehicle numbers if the change does not violate the constraints. If it is not possible to change the vehicle for any selected node due to the constraint violations, give up a merge mutation. This mutation helps to reduce the fleet size of a solution.

(ii) *Time mutation*: Select the arrival node of the longest path on the route with the longest travel time and change its vehicle number to the one with the shortest travel time if the change does not violate the constraints. If the change violates the constraints, attempt repeatedly to change to the vehicle number with the next shortest travel time. If it is not possible to change the vehicle for the selected node, give up a time mutation. This mutation helps to reduce the total travel time of a solution.

(iii) *Tardy mutation*: Select a node whose weighted tardiness is the largest on the route with the largest total weighted tardiness and change its vehicle number to the one with the smallest total weighted tardiness if the change does not violate the constraints. If the change violates the constraints, try repeatedly to change to the vehicle number with the next smallest total weighted tardiness. If it is not possible to change the vehicle for the selected node, give up a tardy mutation. This mutation helps to reduce the total weighted tardiness of a solution.

(iv) *Random mutation*: Select a node randomly in a chromosome and change its vehicle number to one of the other vehicle numbers if the change does not violate the constraints. If it is not possible to change the vehicle for the selected node to any other one, assign a new vehicle number to the node.

(v) *Exchange mutation*: Select randomly two nodes in a chromosome and exchange mutually the positions of these nodes together with the corresponding vehicle numbers if the exchange does not violate the constraints. If it is not possible to exchange them, attempt repeatedly an exchange with another two randomly selected nodes. If it is not possible to exchange any two nodes, give up an exchange mutation.

The first three mutations try to find, from a given solution, a new solution improved on one of three objectives. These mutations integrate the gradient descent method implicitly into genetic algorithm, and thus the fitness of a chromosome may be improved by implementing one of the three mutations selectively based on the route characteristics of a chromosome to mutate. The last two mutations increase the population diversity in the genetic search.

In order to maximize the effectiveness of mutation, a simple mutation selection procedure by which the above various mutations are integrated is developed. In the selection procedure, the order of considering the mutations is varied on the basis of the importance of three objectives. As an example, the following selection procedure is based on the assumption that the importance order of three objectives is the minimization of fleet size, total travel time, and total weighted tardiness. The multiple of minimal values in Steps 2 and 3 can be adjusted.

Step 1: Select a merge mutation if its application is possible. Otherwise, move to Step 2.

Step 2: Select a time mutation if the maximal travel time of routes is 1.3 times longer than their minimal value and its application is possible. Otherwise, move to Step 3.

Step 3: Select a tardy mutation if the maximal total weighted tardiness of routes is 1.3 times larger than their minimal value and its application is possible. Otherwise, move to Step 4.

Step 4: Select a random mutation and an exchange mutation alternately.

3. Example

In this section, I present result of solving an example of VSPDT by HGAV. The data used for the example are based on Solomon's R101 problem [3]. The data for the node coordinates and demands are the same as those used in R101 problem. The due times of nodes are deemed as the lower limits of time windows used in R101 problem and the time deadlines are determined by adding 10 time units to the respective due times. The important characteristics of the example named MR101 are summarized in Table 1. The test

problems of VSPDT are distinguished by affixing an 'M' before the original names of Solomon's problems. The actual data of the example are available from the author.

The weights of the three objectives are set to 0.6 for total travel time, 0.3 for total weighted tardiness, and 0.1 for fleet size. In applying the HGAV, the population size is set to 90, the number of sub-populations to 3, the crossover rate to 0.4, the mutation rate to 0.4, the interval of local optimization to 5, and the number of chromosomes for local optimization in sub-population to 3.

Fig. 1 shows graphically three objective values and their weighted sum of the best chromosomes obtained after every 100 generations up to 1000 generations during the HGAV run in the example, including the results obtained after the first generation. The results show that the solution is improved gradually until about 600 generations, but after then no significant improvement is observed. The best solution obtained after 1000 generations has a total travel time of 1586.4, total weighted tardiness of 11.7, fleet size of 21, and weighted sum of three objectives of 37.4. The vehicle routes of the best solution are depicted in Fig. 2.

4. Computational study

In order to evaluate the performance of the HGAV, the computational study is performed in three aspects: first, a comparison of the HGAV with modified HGAV that allows infeasible solutions by associating a penalty with all constraint violations; second, a sensitivity analysis with varied genetic parameters; third, a comparison of the HGAV with BC-saving algorithm [20]. The HGAV and BC-saving algorithm were programmed in Visual Basic 5.0 and all computational experiments were conducted on an IBM-compatible PC 586 (32 M RAM, Intel MMX 233).

4.1. Comparison of HGAV with modified HGAV

The HGAV applies special representation mappings and repair processes to correct any infeasible solutions generated, and so always guarantees the generation of feasible solutions. The HGAV can be modified to allow infeasible solutions by associating the penalties with all constraint violations and including them in the fitness function of a solution. The performances of these two different approaches for handling the given constraints are compared with each other. In the modified HGAV, the weighted sum of three objectives of a chromosome is given by

$$f(S_k) = \left(w_1 \frac{t_k}{t^0_{\max}} + w_2 \frac{d_k}{d^0_{\max}} + w_3 \frac{v_k}{v^0_{\max}} + \rho_1 l_k + \rho_2 r_k + \rho_3 q_k \right) \times 100, \quad (4)$$

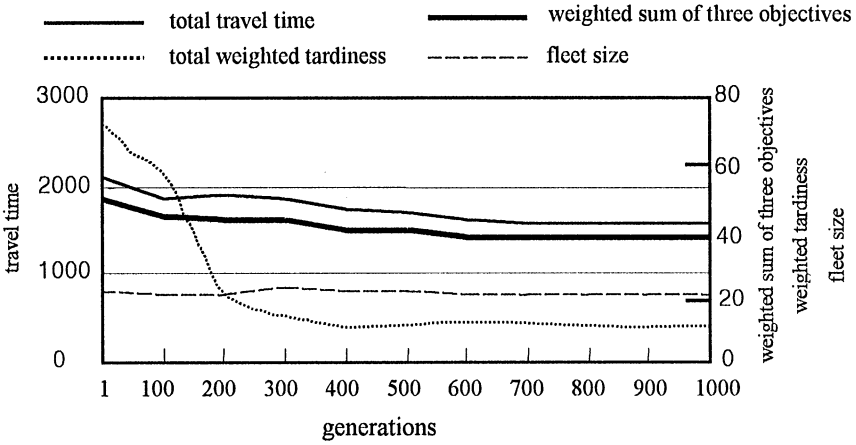

Fig. 1. Variation of three objective values and their weighted sum over 1000 generations in example.

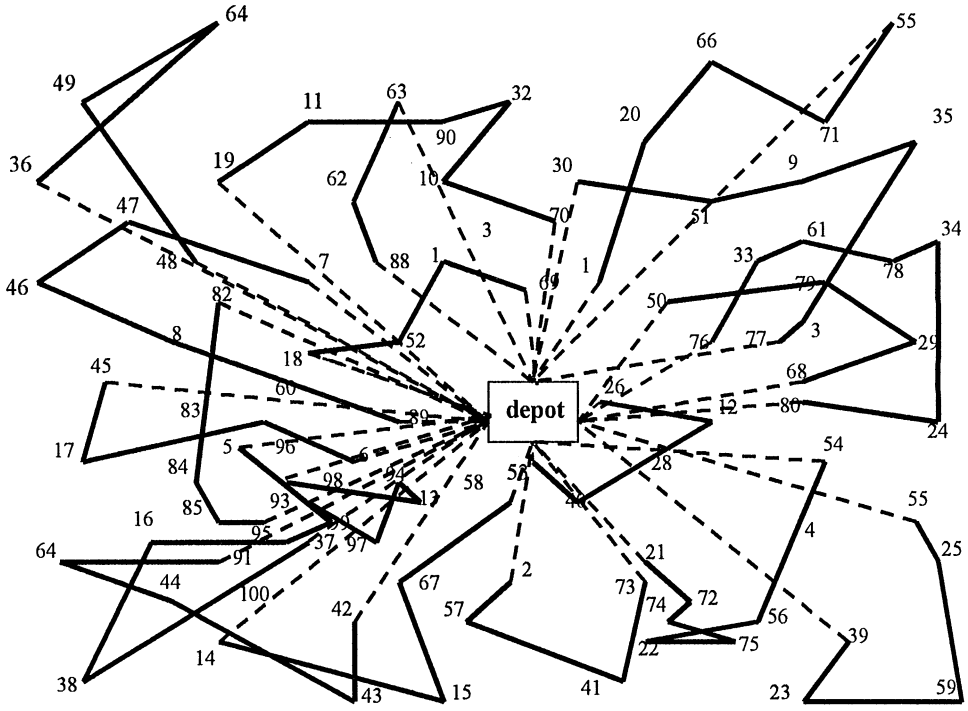

Fig. 2. Solution of vehicle routes for example after 1000 generations.

where ρ_1 is the penalty coefficient for the constraint of time deadline, ρ_2 the penalty coefficient for the constraint of vehicle return time to depot, ρ_3 the penalty coefficient for the constraint of vehicle capacity, l_k the number of nodes whose time deadlines are violated in chromosome k, r_k the number of routes that violate the vehicle return time to depot in chromosome k, q_k the number of routes that violate the vehicle capacity in chromosome k.

For experiments, two problems are constructed on the basis of the data used in Solomon's R101 and RC201 problems [3]. The data for the due times and time deadlines of nodes are determined by the same way as in the example in Section 3. The characteristics of two test problems named MR101 and MRC201 are summarized in Table 1. In applying the HGAV and modified HGAV, the population size is set to 90, the crossover and mutation rates are set to 0.4 each, and w_1, w_2 and w_3 are arbitrarily assumed to be 0.8, 0.1 and 0.1 each. In the modified HGAV, ρ_1, ρ_2 and ρ_3 are equally set to 1.

Fig. 3 shows the comparison of the HGAV with modified HGAV with respect to the solution quality and speed of searching for the best solution in two test problems. It is seen from Fig. 3 that both algorithms improve the solutions on both problems as the evolutionary process is continued, except between generations 100 and 200 when the modified HGAV is applied to MRC201 problem. However, the HGAV quickly finds a very good solution right after the first round of evolution and converges to the final solution much faster than the modified HGAV on both problems. This outcome seems to be due to the fact that the modified HGAV concentrates on improving infeasible solutions being continuously generated rather than on moving toward an optimal feasible solution during the evolutionary process.

4.2. *Sensitivity analysis with varied genetic parameters of HGAV*

It is required to determine the values for genetic parameters in applying the HGAV. The important

Table 1
Summary of characteristics of 20 test problems

Set	Problem	Vehicle capacity	Vehicle return time	Time deadline	Service time
MR1	MR101	200	230	Due time + 10	10
	MR102	200	230	Due time + 10	10
	MR103	200	230	Due time + 10	10
	MR105	200	230	Due time + 30	10
	MR106	200	230	Due time + 30	10
MR2	MR201	1000	1000	Due time + 116	10
	MR202	1000	1000	Due time + 57	10
	MR203	1000	1000	Due time + 103	10
	MR205	1000	1000	Due time + 240	10
	MR206	1000	1000	Due time + 240	10
MRC1	MRC101	200	240	Due time + 30	10
	MRC102	200	240	Due time + 30	10
	MRC103	200	240	Due time + 30	10
	MRC105	200	240	Due time + 54	10
	MRC106	200	240	Due time + 60	10
MRC2	MRC201	1000	960	Due time + 120	10
	MRC202	1000	960	Due time + 120	10
	MRC203	1000	960	Due time + 240	10
	MRC205	1000	960	Due time + 223	10
	MRC206	1000	960	Due time + 240	10

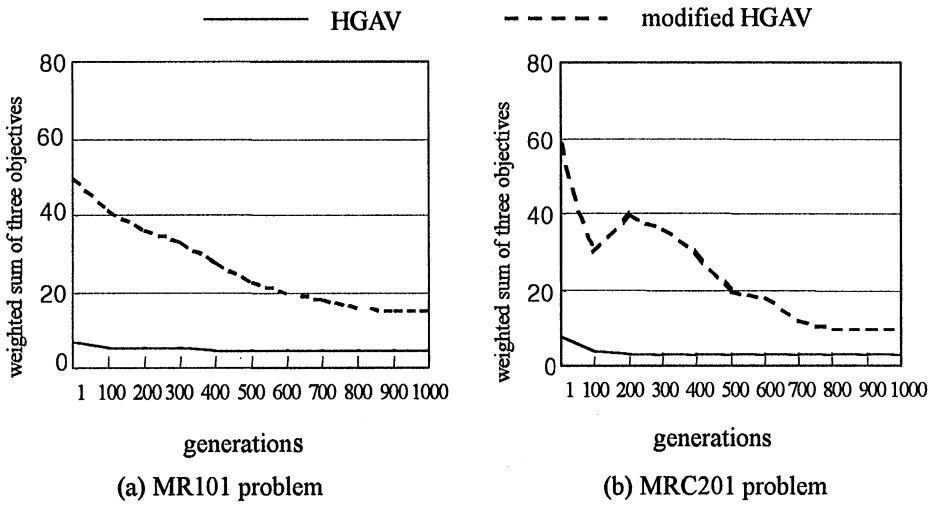

Fig. 3. Comparison of HGAV with modified HGAV in MR101 and MRC201 problems.

genetic parameters include the number of generations (GEN), initial population size (pop_size), number of sub-populations (l), crossover rate (p_c), mutation rate (p_m), number of chromosomes for local optimization in each sub-population (n), and local optimization interval (m). It is known that the genetic parameters affect significantly the performance of genetic algorithm such as the solution quality and search time for a solution. However, there exists no analytical method generally

applicable to find the best values for the parameters because they are heavily dependent upon the problems to be solved.

In this section, I present the sensitivity analysis with varied genetic parameters of the HGAV on the weighted sum of three objectives. It should be noted that the experimental results are not conclusive, because each sensitivity analysis is performed with one varied parameter while the remaining parameters are fixed to their basic settings and so the interaction effects among the parameters are ignored. A total of 4 sensitivity analyses are performed. Their outcomes will be used in determining the best parameter values for the HGAV in Section 4.3. For experiments, MR101 problem is employed. The basic setting of parameters is GEN = 1000, pop_size = 90, $l = 3$, $p_c = 0.4$, $p_m = 0.4$, $n = 3$, and $m = 5$. The basic setting of weights of three objectives is arbitrarily assumed as $w_1 = 0.6$, $w_2 = 0.3$, and $w_3 = 0.1$.

Before discussing the sensitivity analysis on MR101 problem, it is noted that very similar results were obtained from the sensitivity analysis with varied genetic parameters for the various settings of three objectives' weights in several problems listed in Table 1. That is, the results of the sensitivity analysis were almost the same for the problems and settings of three objectives' weights.

4.2.1. Variation of population size

Based on the above settings, the population size is varied from 30 to 390 by 60. Fig. 4 shows graphically the results for each setting. The results show that when the population size is larger than 90, its value has no significant influence on the performance of HGAV.

4.2.2. Variation of local optimization interval

Based on the above basic settings, the local optimization interval is varied from 10 to 100 by 10 including an interval of 1. Fig. 5 shows graphically the results for each setting. It is seen from the figure that the solution is improved as the local optimization interval is reduced. However, the computation time is increased to about 1.7–2.5 times as the interval is reduced to half. This means that there exists a trade-off between the solution quality and the computation time.

4.2.3. Variations of crossover and mutation rate

Based on the above settings, each of the crossover and mutation rates is varied from 0.0 to 1.0 by 0.2 while the other is fixed to its basic setting. Fig. 6 shows graphically the results for each setting. The results show that neither the varied crossover nor the mutation rate has a significant influence on the performance of HGAV, though crossover and mutation with a probability of 0.4 yield marginally the best solution.

4.3. Comparison of HGAV with BC-saving algorithm

The proposed HGAV is compared with the BC-saving algorithm [20] that is a heuristic developed

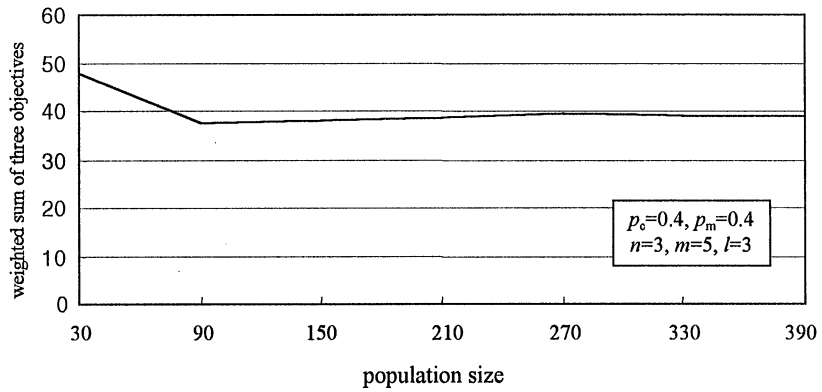

Fig. 4. Results with varied population size.

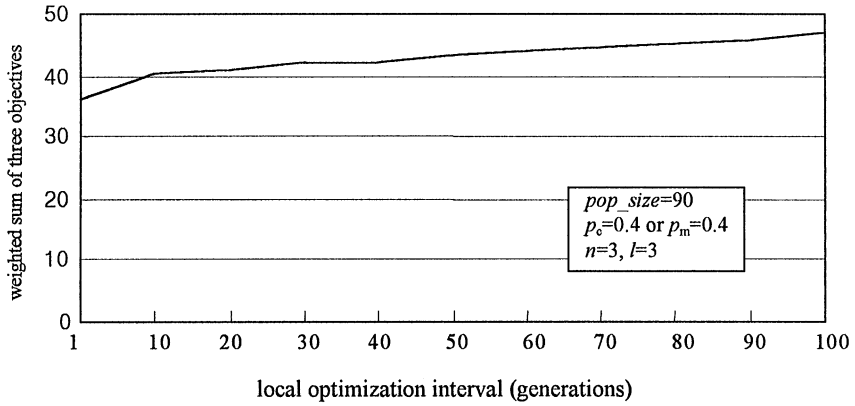

Fig. 5. Results with varied local optimization interval.

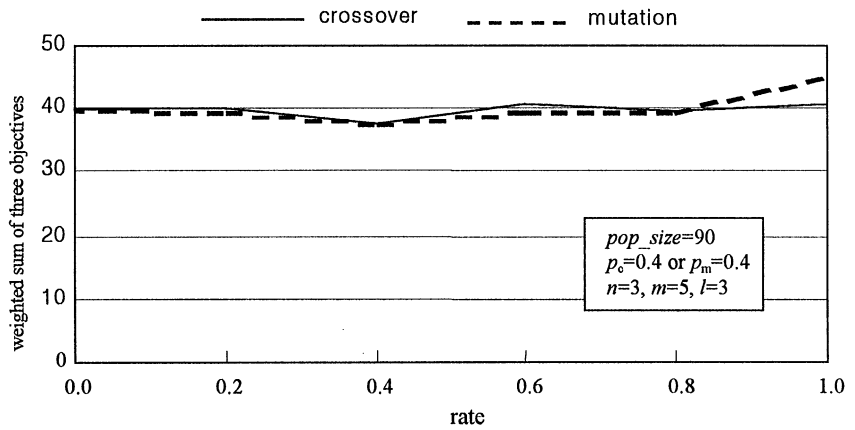

Fig. 6. Results with varied crossover and mutation rates.

for solving the bicriteria vehicle scheduling problem with time- and area-dependent travel speeds and is proven to be excellent. The BC-saving algorithm constructs the routes in parallel by combining two partially formed routes whose total saving is the largest while keeping the constraints given. The total saving is the weighted sum of the savings in vehicle travel time (ST_{ij}) and weighted tardiness (SD_{ij}) that are expected by connecting the last node i to visit of one route to the first node j to visit of the other route. In applying the BC-saving algorithm to solve the VSPDT, I used the following saving equation, assuming that vehicle travel speed is always constant. The third term in the equation represents the saving in fleet size.

$$BC_{ij} = w_1 \frac{ST_{ij}}{t^0_{\max}} + w_2 \frac{SD_{ij}}{d^0_{\max}} + w_3 \frac{1}{v^0_{\max}}. \quad (5)$$

The procedural steps of the BC-saving algorithm are briefly summarized as follows:

Step 1: Construct $N-1$ distinct routes in which each customer is served by a dedicated vehicle. Compute the vehicle travel time and total weighted tardiness for each route.

Step 2: Select two partially formed routes and perform the feasibility test for their combined route with respect to the restrictions for vehicle

and delivery time. Repeat the process of selection and feasibility test for all the partially formed routes, and compute BC_{ij} of all the feasible pairs of nodes for combination. If there exist no feasible pairs of nodes to link, go to Step 4.

Step 3: Combine the two routes with the largest value of BC_{ij} into a single route, and go to Step 2.

Step 4: Compute the improvement in the weighted sum of three objectives, Eq. (1) in Section 2.2, by interchanging adjacent pairs of nodes in a route while keeping the restrictions. If there exist no feasible adjacent pairs of nodes to interchange or no improvement is expected from the feasible interchanges, then go to Step 5. Otherwise, construct a new route by switching the positions of the adjacent pair that shows the largest improvement. Repeat the interchange process to the newly constructed route until no more improvement is expected.

Step 5: Apply Step 4 to the rest of the routes and stop. The vehicles are scheduled on the basis of the final vehicle routes obtained.

For comparison test, five problems are selected from each of Solomon's problem sets R1, R2, RC1, and RC2 [3]. A total of 20 test problems with a single depot and 100 nodes are constructed on the basis of these problems. The geographical data are randomly generated by a random uniform distribution in the problem sets R1 and R2, and semi-clustered by a mix of randomly generated data and clusters in the problem sets RC1 and RC2. The sets R1 and RC1 have a short scheduling horizon; in contrast, the sets R2 and RC2 have a long scheduling horizon. In test problems, data for node coordinates and demands are the same as the one used in Solomon's problems. The due times of nodes are deemed as the lower limits of the time windows used in Solomon's problems and the time deadlines are determined by adding a numerical value to the respective due times. The numerical value is approximately the average travel time between nodes in each problem. The actual data for test problems are available from the author. Some important characteristics of test problems are summarized in Table 1.

In applying the HGAV to test problems, I set GEN = 1000, pop_size = 90, $p_c = 0.4$, $p_m = 0.4$, $n = 3$ and $m = 5$, based on the experimental results obtained in Section 4.2. Three settings for the weights of three objectives with the different first-priority objectives are considered in all test problems to investigate how they impact on the performance of HGAV. The computational results are given in Table 2. The reduction rate is computed by

(weighted sum of three objectives of the solution obtained by BC-saving algorithm − weighted sum of three objectives of the solution obtained by HGAV) × 100/weighted sum of three objectives of the solution obtained by BC-saving algorithm.

Thus, the positive rate denotes the percentage improvement of solution by the HGAV over the BC-saving algorithm. The average reduction rates for four problem sets and three settings of the objective weights are summarized in Table 3.

The computational results are very impressive. It is known from Tables 2 and 3 that the HGAV is clearly superior to the BC-saving algorithm in all test problems regardless of the problem types and objective weights, showing shorter total travel time, less total weighted tardiness, and smaller fleet size. The overall average of reduction rate is computed as about 8.0%. In particular, it appears that the HGAV performs very well in the problem sets MR2 and MRC2 with the weight setting of (0.3, 0.1, 0.6). This outcome may be explained by the fact that the sets MR2 and MRC2 have the characteristic of requiring smaller fleet size than the other two sets, so the reduction in fleet size with a high weight of 0.6 leads to a significant decrease in the weighted sum of objectives. From Table 2, it is seen that three objectives are traded off well in the HGAV according to the changes of their weights. For instance, when the weight setting is (0.3, 0.6, 0.1), the total weighted tardiness is by far the least among three weight settings, owing to the increase in total travel time and fleet size.

In terms of computation time, I did not perform a precise and extensive comparison of two algorithms on the test problems. Based on the rough measurement, the HGAV and BC-saving algorithms require 6–30 CPU minutes and 10–20 CPU seconds, respectively, depending on the problems to be solved. The HGAV requires much longer CPU time to obtain a high-quality solution than the BC-saving algorithm.

Table 2
Comparison of HGAV with BC-saving algorithm[a]

Problem		(0.6, 0.3, 0.1)[b]			(0.3, 0.6, 0.1)			(0.3, 0.1, 0.6)		
		HGAV	BC-saving	RR (%)	HGAV	BC-saving	RR (%)	HGAV	BC-saving	RR (%)
		ST VT	ST VT		ST VT	ST VT		ST VT	ST VT	
MR1	MR101	37.4 (1586, 12, 21)[c]	39.0 (1642, 19, 23)	5.2	22.8 (1634, 3, 21)	24.6 (1695, 5, 24)	7.3	56.7 (1580, 40, 20)	61.3 (1622, 45, 22)	7.5
	MR102	42.0 (1527, 906, 19)	44.7 (1578, 1025, 21)	6.1	33.5 (1712, 599, 24)	35.3 (1758, 646, 26)	5.6	56.6 (1503, 1089, 18)	61.2 (1515, 1118, 20)	7.6
	MR103	43.5 (1420, 4480, 18)	46.0 (1478, 4956, 19)	5.5	39.5 (1659, 3557, 21)	42.1 (1669, 3942, 23)	6.1	59.7 (1399, 6020, 15)	65.8 (1406, 6497, 17)	9.3
	MR105	37.7 (1511, 145, 19)	39.9 (1553, 189, 21)	5.4	25.2 (1726, 18, 22)	27.7 (1790, 31, 26)	8.9	60.0 (1598, 250, 19)	66.6 (1550, 282, 22)	9.9
	MR106	39.8 (1414, 1911, 17)	42.9 (1471, 2326, 19)	7.3	32.6 (1667, 1302, 22)	34.8 (1671, 1542, 24)	6.4	56.0 (1401, 2597, 16)	61.3 (1415, 2785, 18)	8.7
MR2	MR201	27.1 (978, 1158, 8)	29.2 (1023, 1319, 9)	7.2	19.6 (1027, 554, 9)	20.3 (1088, 582, 9)	3.4	40.1 (960, 1750, 6)	50.2 (969, 1769, 8)	20.2
	MR202	27.2 (912, 4633, 7)	29.3 (951, 5007, 8)	7.1	15.7 (658, 3011, 5)	16.6 (894, 6260, 7)	5.6	42.6 (889, 5897, 5)	56.2 (910, 6349, 7)	24.2
	MR203	28.5 (929, 11458, 6)	31.2 (973, 12919, 7)	8.5	26.2 (1058, 9674, 7)	28.0 (1077, 10063, 8)	6.3	46.5 (870, 10112, 5)	55.1 (909, 15169, 6)	15.5
	MR205	27.1 (941, 2122, 7)	29.0 (979, 2228, 8)	6.2	21.8 (1014, 1075, 9)	23.1 (1058, 1277, 10)	7.3	37.7 (894, 7985, 4)	44.4 (911, 7670, 5)	15.2
	MR206	27.2 (874, 5721, 7)	28.5 (929, 6373, 7)	4.7	23.0 (1041, 4399, 7)	25.0 (1051, 4959, 8)	8.7	40.3 (823, 15987, 4)	54.1 (851, 9172, 6)	25.6
MRC1	MRC101	40.1 (1875, 314, 20)	42.0 (1924, 352, 22)	4.6	25.5 (2093, 17, 25)	27.2 (2161, 49, 26)	6.2	56.8 (1865, 398, 19)	61.3 (1878, 422, 21)	7.3
	MRC102	42.6 (1710, 3478, 19)	45.5 (1772, 3796, 19)	6.4	36.1 (2109, 2131, 23)	38.5 (2096, 2518, 24)	6.3	51.1 (1690, 3216, 17)	56.1 (1727, 3787, 19)	5.5
	MRC103	42.2 (1604, 6049, 15)	44.8 (1648, 6595, 17)	5.9	42.1 (1877, 4627, 22)	43.6 (1897, 4950, 22)	3.4	53.9 (1574, 8745, 14)	58.3 (1590, 7823, 16)	8.0
	MRC105	38.2 (1645, 991, 18)	39.4 (1677, 1033, 19)	2.9	28.0 (2068, 203, 25)	29.7 (2148, 262, 26)	5.8	53.7 (1622, 1022, 17)	53.9 (1637, 1048, 17)	0.3
	MRC106	35.1 (1588, 1024, 17)	36.4 (1614, 1122, 18)	3.6	26.0 (1985, 207, 25)	27.7 (2093, 322, 24)	6.1	49.1 (1522, 1346, 16)	49.6 (1540, 1547, 16)	1.0
MRC2	MRC201	20.7 (945, 573, 7)	22.8 (1009, 692, 8)	9.2	16.0 (1079, 156, 9)	17.7 (1188, 175, 10)	9.7	39.1 (927, 2487, 6)	43.2 (1029, 1014, 7)	9.5
	MRC202	22.7 (915, 6145, 7)	24.8 (982, 6569, 8)	8.5	22.1 (1146, 4117, 10)	23.5 (1208, 4834, 10)	5.7	40.6 (880, 8873, 6)	40.9 (891, 9975, 6)	0.8
	MRC203	26.0 (934, 13322, 7)	27.9 (1043, 14328, 7)	6.3	25.8 (1188, 7856, 10)	27.6 (1229, 8344, 11)	6.6	41.2 (900, 12542, 5)	55.3 (969, 14834, 7)	25.5
	MRC205	21.2 (899, 2589, 5)	22.5 (923, 2626, 6)	5.8	17.1 (1037, 821, 8)	18.1 (1042, 832, 9)	5.4	37.3 (1015, 5678, 5)	40.8 (905, 3177, 6)	8.4
	MRC206	20.0 (935, 1504, 6)	22.0 (964, 1659, 7)	6.9	16.2 (1011, 1048, 7)	17.7 (1027, 1176, 8)	7.9	35.1 (921, 3156, 5)	40.7 (914, 3615, 6)	13.8

[a]ST: weighted sum of three objectives; VT: values of three objectives; RR: reduction rate.
[b](Weight of total travel time, weight of total weighted tardiness, weight of fleet size).
[c](Total travel time, total weighted tardiness, fleet size).

Table 3
Summary of reduction rate (%)

(w_1, w_2, w_3)	Set MR1	Set MR2	Set MRC1	Set MRC2	Average
(0.6, 0.3, 0.1)	5.9	6.7	4.7	7.3	6.2
(0.3, 0.6, 0.1)	6.9	6.3	5.6	7.0	6.4
(0.3, 0.1, 0.6)	8.6	20.1	5.1	11.6	11.4
Average	7.1	11.0	5.1	8.6	8.0

It is observed from the experiments that the computation time required by the HGAV is approximately proportional to the number of generations and local optimizations executed. So its long computation time is mainly due to the large number of generations (GEN = 1000) and relatively short interval of local optimization ($m = 5$). Therefore, the computation time can be reduced to about 3.6–18 CPU minutes (or 6×0.6–30×0.6 CPU minutes) with a little sacrifice of the solution quality, by stopping the evolution at 600 generations based on the experimental outcome in the example. It is also observed that the computation time required by the HGAV is varied depending on the number of routes in the problem. In general, the problem of more routes requires longer computation time. As a matter of fact, the high quality solution obtained by the HGAV is a result of a high computation cost.

5. Conclusions

In this paper, I have presented a hybrid genetic algorithm incorporating a greedy interchange local optimization procedure for the vehicle scheduling problem with due times and time deadlines where three conflicting objectives of the minimization of total travel time, total weighted tardiness, and fleet size are explicitly considered. The algorithm employs a mixed farming and migration strategy along with a variant of the partially mapped crossover and several mutation operators newly developed while maintaining the solution feasibility during the whole evolutionary process.

From the extensive computational experiments, three important observations are summarized as follows:

1. A better solution is always obtained by the HGAV in much longer CPU time in comparison with the BC-saving algorithm. The reduction rate of the weighted sum of three objectives by the HGAV is computed as 8.0% on the average.
2. The HGAV quickly finds a very good solution right after the first round of evolution and converges to the final solution much faster than the modified HGAV that allows infeasible solutions by associating the penalties with all constraint violations.
3. In applying the HGAV, the population size over a certain level has no significant impact on the solution quality, a solution is improved as the local optimization interval is reduced while the computation time is increased to about double as the interval is reduced to half, and the variation of the crossover or mutation rate has no significant influence on the solution quality.

In some practical situations of applying HGAM, the individual values of three objectives for different weight vectors in the form of three-dimensional vector might be helpful in examining the Pareto-optimal solutions. Finally, it is noted that further research should be focused on the reduction in computation time required by the HGAV. The reduction may be possible by developing a new method of generating better initial population and a new crossover operator that enables to introduce many good schemata (a template allowing exploration of similarities among chromosomes) [19] to the sub-populations in subsequent generations.

References

[1] L. Bodin, B. Golden, A. Assad, M. Ball, Routing and scheduling of vehicles and crews: the state of the art, Computers and Operations Research 10 (1983) 62–212.
[2] B. Golden, A. Assad (Eds.), Vehicle routing with time window constraints: algorithmic solutions, American Journal of Mathematical and Management Sciences, Vol. 15, American Sciences Press, Columbia, OH, 1986.
[3] M.M. Solomon, Time window constrained routing and scheduling problems: A survey, Transportation Science 22 (1) (1988) 1–13.

[4] G. Laporte, The vehicle routing problem: An overview of exact and approximate algorithms, European Journal of Operational Research 59 (1992) 345–358.
[5] S. Anily, Vehicle routing and the supply chain, in: S. Tayur, R. Ganeshan, M. Magazine (Eds.), Quantitative Models for Supply Chain Management, Kluwer Academic Publishers, Dordrecht, 1999, pp. 149–196 (Chapter 6).
[6] P. Gabbert, D. Brown, C. Huntley, B. Markowicz, D. Sappington, A system for learning routes and schedules with genetic algorithms, Proceedings of Fourth International Conference on Genetic Algorithms, 1991, pp. 430–436.
[7] S.R. Thangiah, R. Vinayagamoorthy, A.V. Gubbi, Vehicle routing with time deadlines using genetic and local algorithms, Proceedings of the Fifth International Conference on Genetic Algorithms, Los Altos, CA, Morgan Kaufmann, Los Altos, CA, 1993, pp. 506–513.
[8] J.L. Blanton, R.L. Wainwright, Multiple vehicle routing with time and capacity constraints using genetic algorithms, Proceedings of the Fifth International Conference on Genetic Algorithms, Los Altos, CA, Morgan Kaufmann, Los Altos, CA, 1993, pp. 452–459.
[9] S.R. Thangiah, An adaptive clustering method using geometric shape for vehicle routing with time windows, Proceedings of the Sixth International Conference on Genetic Algorithms, Los Altos, CA, Morgan Kaufmann, 1995, pp. 536–543.
[10] R. Cheng, M. Gen, Fuzzy vehicle routing and scheduling problem using genetic algorithms, in: F. Herrera, J. Verdegay (Eds.), Genetic Algorithms and Soft Computing, Springer, Berlin, 1996, pp. 683–709.
[11] J. Potvin, D. Dube, C. Robillard, A hybrid approach to vehicle routing using neural networks and genetic algorithms, Applied Intelligence 6 (1996) 241–252.
[12] C. Malmborg, A genetic algorithm for service level based vehicle scheduling, European Journal of Operational Research 93 (1996) 121–134.
[13] X. Chen, W. Wan, X. Xu, Modeling rolling batch planning as vehicle routing problem with time windows, Computers and Operations Research 25 (12) (1998) 1127–1136.
[14] M. Gen, R. Cheng, Genetic Algorithms & Engineering Design, Wiley, New York, 1997.
[15] C. Reeves, Genetic algorithms and neighborhood search, Evolutionary Computing, Springer, Berlin, 1994, pp. 115–130.
[16] F. Marin, O. Trelles-Salazar, F. Sandoval, Genetic algorithm on LAN-based message passing architectures using PVM: Application to the routing problem, Proceedings of the Third Conference on Parallel Problem Solving from Nature, Israel, 1994, pp. 534–543.
[17] D.E. Goldberg, Genetic Algorithms in Search, Optimization & Machine Learning, Addison-Wesley, Reading, MA, 1989.
[18] M. Dror, L. Levy, Avehicle routing improvement algorithm comparison of a "GREEDY" and a matching implementation for inventory routing, Computers and Operations Research 13 (1) (1986) 33–45.
[19] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, 2nd Edition, Springer, Berlin, 1994.
[20] Y.B. Park, Asolution of the bicriteria vehicle scheduling problems with time and area-dependent travel speeds, Computers and Industrial Engineering 38 (2000) 173–187.