A knowledge-based evolutionary algorithm for the multiobjective vehicle routing problem with time windows

To appear in Computers & Operations Research, vol. 45, 2014

http://dx.doi.org/10.1016/j.cor.2013.11.014

Authors	Tsung-Che Chiang* and Wei-Huai Hsu
Affiliation	Department of Computer Science and Information Engineering, National Taiwan Normal University
Postal address	No.88, Sec. 4, Tingzhou Rd., Wenshan District, Taipei City 116, Taiwan, R.O.C.
Email	tcchiang@ieee.org
Telephone	+886-2-77346692
Fax	+886-2-29322378

A knowledge-based evolutionary algorithm for the multiobjective vehicle routing problem with time windows

Abstract

This paper addresses the multiobjective vehicle routing problem with time windows (MOVRPTW). The objectives are to minimize the number of vehicles and the total distance simultaneously. Our approach is based on an evolutionary algorithm and aims to find the set of Pareto optimal solutions. We incorporate problem-specific knowledge into the genetic operators. The crossover operator exchanges one of the best routes, which has the shortest average distance, the relocation mutation operator relocates a large number of customers in non-decreasing order of the length of the time window, and the split mutation operator breaks the longest-distance link in the routes. Our algorithm is compared with 10 existing algorithms by standard 100-customer and 200-customer problem instances. It shows competitive performance and updates more than 1/3 of the net set of the non-dominated solutions.

Keywords: Vehicle routing problem; Time windows; Multiobjective; Pareto optimal; Evolutionary algorithm

1. Introduction

The Vehicle Routing Problem (VRP) aims to find the optimal set of routes for a fleet of vehicles to serve customers under specific constraints. In its basic form, the VRP involves a single depot as the start and end points of the routes. Each customer is associated with a location and a demand quantity. Each vehicle serves the customers along the designated route, and the total demand cannot exceed the maximum capacity. The VRP is a combination of two classical NP-hard combinatorial optimization problems, the bin packing problem and the traveling salesman problem (TSP). Similar to bin packing, solving the VRP requires partitioning customers into vehicles to minimize the required number of vehicles without violating the capacity constraint. For each vehicle, the VRP asks to find the lowest-cost (usually the shortest-distance) driving path, which is the same as what the TSP needs. The Vehicle Routing Problem with Time Windows (VRPTW) is an extension of the VRP. In the VRPTW, each customer has a predefined time window. A vehicle can start to serve a customer only within the time window. In this study, we take the time window as a hard constraint. If a vehicle arrives at the customer’s location earlier, it must wait until the beginning of the time window; if the vehicle arrives later than the end of the time window, the solution is not acceptable. The VRPTW has many real-world applications, such as postal delivery, waste collection, school bus routing, and so on. Due to the challenging problem complexity and high practical value, the VRPTW is a very important research topic in the fields of operations research, transportation science, and computer science.

Several objectives have been considered in the VRPTW, and minimization of the number of vehicles and the total travel distance are the most common objectives in the literature. The classical way to address these two objectives is to minimize the number of vehicles first and then to minimize the total distance with the minimal number of vehicles. The number of vehicles is related to the investment of purchasing vehicles and the cost of hiring drivers; the travel distance, on the other hand, is related to the fuel cost. Optimizing the two objectives in the classical way implies that the vehicle-related cost is much higher than the distance-related cost. In many cases, however, fleet managers want to know the trade-off between these two objectives before determining the best routing plan. To accomplish this goal, another stream of research was started by searching for the set of Pareto optimal solutions rather than a single optimal solution. Hereafter, Pareto approaches refer to the approaches for which the goal is to find the Pareto set. The definition of Pareto optimal solutions and the Pareto set will be given in the next section. Simply speaking, solutions in the Pareto set are not worse than any other in both objectives simultaneously. By looking into the trade-off between these solutions, managers can get more information and make a better decision.

This paper proposes a knowledge-based evolutionary algorithm (KBEA) to solve the VRPTW. The remainder of this paper is organized as follows. Section 2 defines the target problem and the objectives. Section 3 gives the literature review. The proposed approach is elaborated in Section 4, and the experiments and results are detailed in Section 5. Section 6 draws the conclusions and provides future research directions.

2. Multiobjective Vehicle Routing Problem with Time Windows (MOVRPTW)

The VRPTW involves two types of objects: locations and vehicles. A special location 0 represents the depot. The remaining N locations correspond to N customers. For each customer i ($1 \le i \le N$), the demand q_i, the service time s_i, and the time window $[e_i, l_i]$ are known in advance. The travel distance and travel time between two locations i and j are denoted by d_{ij} and t_{ij}. (In this study, we assume that d_{ij} equals t_{ij}.) The vehicles are homogeneous and have the same maximum capacity Q. A feasible solution to the VRPTW must satisfy the following constraints:

(1) Each customer must be served by exactly one vehicle exactly one time.

(2) The route of each vehicle must start from and end at the depot.

(3) The total demand of the customers served by each vehicle cannot exceed the maximum capacity Q.

(4) A vehicle must arrive at customer i no later than the end of the time window: $a_i \le l_i$, where a_i denotes the arrival time at customer i.

(5) The service cannot start before the beginning of the time window: $b_i = \max\{a_i, e_i\}$, where b_i is the service start time at customer i.

(6) Assume that customer j is served immediately after customer i; then, the arrival time at j is defined by $a_j = b_i + s_i + t_{ij}$.

For each feasible solution, we calculate two objective values: the number of required vehicles and the total travel distance. We say that one solution x *dominates* another solution y if x is not worse than y in both objectives and is better in at least one objective. Taking Fig. 1 as an example, x dominates y, but x and z do not dominate each other. A solution is *Pareto optimal* if it is not dominated by any other solution. The set of Pareto optimal solutions is called the *Pareto set*, and the set of objective vectors of the Pareto optimal solutions is called the *Pareto front*. Take Fig. 1 as an example again. If we solve the VRPTW in the classical way (minimizing the number of vehicles and then the total distance), the fleet manager obtains a single optimal solution w; if we consider the total distance only, then the solution z is obtained. Compared with these two traditional approaches, the Pareto approach will output the Pareto optimal set of solutions {*w*, *x*, *z*} to the manager. It offers the manager the opportunity of choosing solution *x* as the final plan based on the trade-off between two objectives that are of concern.

<< Insert Fig. 1 about here >>

3. Literature review

The NP-hard problem complexity means that currently no algorithm can solve the VRPTW optimally in polynomial time. In the literature, Jepsen et al. (2006) could solve 45 out of 56 100-customer instances in Solomon's data set (1987) optimally in terms of the total distance within hours. However, the exponentially growing computation time would limit the use of exact algorithms when the problem scale becomes larger and larger. Metaheuristics such as genetic algorithms (GA) and tabu search (TS) are promising approximation algorithms that have addressed hard optimization problems in recent decades. They have already demonstrated good performance in solving the VRPTW (Bräysy and Gendreau 2005b, Potvin 2009). Here, we will focus on the literature that applies metaheuristics to solve the VRPTW. We classify the existing studies according to how they addressed the two objectives. Section 3.1 reviews the studies that minimized the objectives in the lexicographical way, and Section 3.2 reviews the studies that consider a minimization of the total distance only. Studies based on Pareto approaches are presented in Section 3.3.

3.1 Classical (lexicographical minimization) approaches

In the literature on the VRPTW, the classical way to address the two most common objectives is to minimize the number of vehicles and then to minimize the total distance with the minimal number of vehicles. This subsection will review past studies that belong to this category and describe the featured design concepts and techniques.

To cater to the lexicographical minimization of the two concerned objectives, many algorithms are composed of two (or more) phases. Examples include Gehring and Homberger (1999), Bräysy (2003), Bent and Hentenryck (2004), Bräysy et al. (2004), Homberger and Gehring (2005), and Lim and Zhang (2007). The first phase attempted to minimize the number of vehicles, and then, the best solution entered the second phase to minimize the total distance. Many different combinations of metaheuristics have been proposed. Gehring and Homberger (1999) used an evolution strategy (ES) in the first phase and a TS in the second phase, while Bent and Hentenryck (2004) used simulated annealing (SA) and a large neighborhood search. Another idea to fit the lexicographical minimization is using two populations simultaneously. Gambardella et al. (1999) developed an ant colony system (MACS-VRPTW) that had two colonies. One colony aimed at minimizing the number of vehicles, and the other colony aimed at minimizing the total distance. Both colonies used independent pheromone trails but shared the global best solution. The first colony attempted to turn an infeasible solution that uses one less vehicle than the global best solution into a feasible solution. Once it found a feasible solution, it restarted after decreasing one more vehicle, and the newly found solution was sent to the other colony to minimize the total distance. A similar concept can be seen in Berger et al. (2003), where they proposed a GA that had two populations.

Based on the lexicographical minimization order, the natural way to compare two solutions during the search process is to compare the number of vehicles first and then the total distance if the solutions use the same number of vehicles. However, Homberger and Gehring (1999) noted that the minimization of the total distance does not inevitably lead to a reduction in the number of vehicles. (In fact, this arrangement implies that there is a conflict between the two objectives and motivates the use of Pareto approaches in the MOVRPTW.) Thus, they introduced two auxiliary objectives in the environmental selection stage of their ES. One objective was the number of customers that were served in the smallest route (the route comprising the fewest customers) in the solution, C_R; the other objective was a measure called the minimal delay, D_R, which estimates the difficulty of moving the customers on the smallest route to other routes. The auxiliary objectives aim to identify potential solutions for further vehicle reduction when multiple solutions use the same number of vehicles. When two solutions use the same number of vehicles, the solution that has a smaller C_R is regarded as the better solution. The idea is simple: it would be easier to remove a route that has fewer customers. In case of a tie in C_R, the solution that has a smaller D_R is better. The minimal delay was also used in Bent and Hentenryck (2004) and Homberger and Gehring (2005). Bent and Hentenryck (2004) took the square of the number of customers in the smallest route as an auxiliary objective. Bouthillier and Crainic (2005) ranked the solutions first by the number of vehicles and then by the weighted sum of the travel time, total distance, waiting time, and residual time.

Most studies either ensure that no infeasible solution is generated by the neighborhood functions or discard infeasible solutions. Only a few studies allowed the infeasible solutions to appear during the search process. Gambardella et al.'s ant colony system (1999) put infeasible solutions in one colony and attempted to maximize the number of served customers. These infeasible solutions obeyed the capacity and time window constraints but did not serve all of the customers. Cordeau et al. (2001) and Berger et al. (2003) intended to minimize the weighted sum of the original objective values and the amount of constraint violation for the infeasible solutions. Nagata et al. (2010) allowed the temporary occurrence of infeasible solutions in the local search-based repair procedure. If the local search could not decrease the constraint violation to zero, however, the infeasible solution was still discarded and did not participate in the evolution process. Vidal et al. (2013) proposed a GA, which evolved feasible and infeasible individuals in two separate sub-populations. The fitness of the infeasible individuals was calculated based on the rankings of the amount of constraint violation and the similarity to other individuals. Two parents were chosen from the union of the two sub-populations by 2-tournament selection. After an offspring was produced by crossover and local search, it was placed into the corresponding population according to its feasibility. Cordeau et al. (2001), Nagata et al. (2010), and Vidal et al. (2013) relaxed both the capacity and time window constraints, and Berger et al. (2003) relaxed only the time window constraints.

Another research point in the VRPTW literature is to develop efficient neighborhood functions. The 2-opt operator removed two links in a route and connected the head (resp. tail) customer in one link to the head (resp. tail) customer in the other link. This procedure required reversing the directions of the links between the tail customer in one link and the head customer in the other. This process can easily cause a violation in the time windows. Potvin et al. (1996) developed the 2-opt* operator. This operator selected two routes and removed one link in each route. Two new routes were formed by connecting the head customer of one link to the tail customer of the other link. This operator preserves the orientation of the links and is more suitable for the VRPTW. Potvin and Bengio (1996) proposed a sequence-based and a route-based crossover operator for GA. Bräysy and Gendreau (2005a) reviewed many operators from the simple relocate operator to the sophisticated GENI-Exchange operator. Nagata et al. (2010) developed an edge assembly crossover (EAX) for their memetic algorithm (MA). Let V denote the set of customers, and let E_A (E_B) denote the sets of edges in the routes of the parent p_A (p_B). The EAX first defined a graph $G_{AB} = (V, E_A \cup E_B \backslash E_A \cap E_B)$. Then, it generated cycles by randomly selecting a starting node on G_{AB} and tracing, in turn, the edges that belong to p_A in the forward direction and p_B in the reverse direction until a cycle is formed. Given a cycle C of links and a parent solution p_A, an intermediate solution was generated by removing the edges $E_A \cap C$ and adding the edges $E_B \cap C$. The edges of the cycle were taken from the parents alternately and were in the opposite orientation; a cycle looks like $e_{12}{}^A \rightarrow e_{32}{}^B \rightarrow e_{34}{}^A \rightarrow e_{14}{}^B$, where $e_{ij}{}^A$ ($e_{ij}{}^B$) means an edge from customer i to customer j in solution p_A (p_B). Customers in the cycle have either one outgoing edge from each parent (e.g., customers 1 and 3 in the mentioned example) or one incoming edge from each parent (e.g., customers 2 and 4). Thus, removing edges $E_A \cap C$ and adding edges $E_B \cap C$ keeps the customers on a single route. Finally, possible subtours are connected to existing routes one at a time in a random order by the 2-opt* operator. Reducing the number of vehicles is a difficult task in solving the VRPTW, and tailored operators are very helpful. Bräysy (2003) proposed a route-elimination procedure that is based on the ejection chains that were originally proposed for solving the TSP (Glover 1992). This approach involved an intelligent reordering, which inserted the target customer into the least-extra-cost position without violating the time window constraint and then reordered the customers ahead of the first time-window-violated customer. Bräysy et al. (2004) presented the injection tree procedure, whose advantage over the ejection chain is the allowance of multiple ejections. In the ejection pool of Lim and Zhang (2007), they carefully determined the customer to eject from the route and the position to insert the customer. Nagata and Bräysy (2009) proposed a route minimization procedure by combining the ejection pool, a local search, and a customer ejection heuristic.

3.2 Distance only

Tan *et al*. (2001) tackled the VRPTW by three metaheuristics, SA, TS, and GA. The SA and TS adopted a 2-interchange neighborhood and had a diversification procedure. The SA used a re-heating action, and the TS used a random sequence of 2-interchange and a relinking operator. The GA used permutation encoding and three crossover operators. It also had an adaptive mutation probability scheme. These methods were tested by Solomon's problem set, and the 18 best-known solutions were updated. Ting and Huang (2005) proposed a GA and developed an elitism strategy, which collected the best individuals from the current population and the offspring to perform a 2-opt improvement. Watanabe and Sakakibara (2007) intended to minimize the total distance, and they took the multiobjectivization approach to add two more objectives. They measured the density and connectivity of the partitions of customers to vehicles. This problem was formulated as a multiobjective problem and was solved by NSGA-II (Deb et al. 2002). Experimental results showed that the multiobjectivization approach performed more effectively and stably. Alvarenga et al. (2007) proposed a two-phase algorithm. One phase was responsible for diversification. A GA was used to solve the original VRPTW over multiple times, and the routes in the best solutions that were obtained by multiple runs were collected. Through a formulation of a set partitioning problem and a mixed integer programming (MIP) solver, an improved solution was obtained. Based on this solution, several reduced VRPTW instances were generated and solved by the GA again. This process served as the intensification phase. The diversification and intensification phases alternated until the computation time limit was reached. Last, the set partitioning problem was solved again to obtain further improvement. This algorithm updated approximately the 30 best known solutions in Solomon's problem set. Labadi et al. (2008) developed an MA that took the weighted sum of the number of vehicles and the total distance as the objective. Their MA had two versions. One version used a large weight on the number of vehicles and solved the problem by lexicographical minimization, and the other version used zero for the weight of the number of vehicles and attempted to minimize the total distance only. This approach used permutation encoding and adapted the route split procedure in Prins (2004) for the VRP to the VRPTW. Population diversity was maintained by a cost-spacing rule and a partial renewal procedure. The former allowed an offspring to join the population only if there was no old individual that had close fitness, while the latter replaced all but several best individuals in the population by randomly generated individuals at specific points in timing. Their algorithm performed better than Alvarenga et al.'s algorithm for the R2 and RC2 categories of instances in Solomon's problem set. Yu et al. (2011) hybridized the ant colony optimization (ACO) and TS. The ACO and TS were executed alternately. When the ACO could not update the best solution for three iterations, the current best solution underwent the TS.

3.3 Pareto approaches

Jozefowiez et al.'s review (2008) indicated that the domain of multiobjective routing problems is still young. Compared with the classical (lexicographical optimization) VRPTW literature, the number of studies for solving MOVRPTW by Pareto approaches is much smaller. This lack motivates us to contribute an approach to MOVRPTW and conduct a comprehensive comparison between existing approaches, which will be detailed in Sections 4 and 5. Geiger's study (2001) is among the earliest Pareto approaches to the MOVRPTW. He resorted to GA and calculated the fitness based on the number of dominating individuals. Four objectives, including the number of vehicles, the total distance, the time window violation, and the number of violated time windows, were considered. He also developed a program to visualize the obtained solutions on the objective space. Testing his GA by a 21-customer instance, the experimental results showed that his GA could reduce all four objective values effectively. Tan et al. (2003)(2006) proposed a hybrid multiobjective evolutionary algorithm (HMOEA) to minimize the number of vehicles and the total distance. The individual fitness was also calculated based on the number of dominating individuals. A route-exchange crossover was proposed, and a multi-mode mutation (2-opt*, route splitting, and route merging) was used. One of three local search algorithms was randomly chosen to apply to all of the individuals every 50 generations. They tested their algorithm on Solomon's benchmark instances and examined whether the two concerned objectives conflicted. The results showed that instances in the categories of C1 and C2 have no conflict but that many instances in other categories do. Barán and Schaerer (2003) proposed a multiobjective ant colony system to minimize the number of routes, the total distance, and the total travel time simultaneously. The amount of deposited pheromone depended on the product of the three average objective values of the solutions in the current Pareto set. They used four multiobjective performance metrics to compare their algorithm with the MACS-VRPTW (Gambardella et al. 1999), but only one problem instance (C101 in Solomon's problem set) was tested. Ombuki et al. (2006) devised a multiobjective GA to minimize the number of vehicles and the total distance. The weighted sum method and the Pareto ranking method were taken as the fitness assignment mechanisms. Their GA used permutation encoding and built a solution by appending customers at the ends of the routes in the same order that the customers appear in the chromosomes. A route crossover and a reversal mutation were applied to generate new individuals. By testing their algorithm on Solomon's problem set, they showed that all of the instances in the R2 and RC2 categories have conflicts between the two objectives. Ghoseiri and Ghannadpour (2010) proposed a GA by combining algorithm components from several past studies. They followed the encoding scheme in Tan et al. (2006), took the Pareto ranking scheme from Ombuki et al. (2006) and used operators from Tan et al. (2006), Ombuki et al. (2006), and Potvin and Bengio (1996). Garcia-Najera and Bullinaria (2011) also proposed a multiobjective evolutionary algorithm (MOEA) whose main feature is the consideration of similarity between individuals during mating selection and environmental selection. Similarity between individuals was calculated by Jaccard's similarity coefficient. In 2-tournament mating selection, one parent was chosen by the Pareto rank while the other was chosen by similarity. The individual that had less similarity to the first parent or to the whole population was selected with a higher probability. The environmental selection mechanism is almost the same as that in NSGA-II. The difference is that the crowding distance was replaced by the similarity measure.

In addition to the number of vehicles and the total distance, research studies have also been conducted to solve the MOVRPTW regarding other objectives. Hong and Park (1999) considered the total travel time and the total customer waiting time. They solved the problem in two phases. In the first phase, customers were inserted into the routes for minimizing the weighted sum of the extra travel time and the waiting time. In the second phase, the two objectives were treated in a lexicographical way depending on the weights of the objectives. Muňoz-Zavala et al. (2009) also took the waiting time as one objective in solving the MOVRPTW. The numerical results showed that there was a large amount of conflict between the waiting time and the total distance in some of the instances in Solomon's problem set. Several researchers regarded the time window as a soft constraint and took the minimization of a constraint violation as an objective. Xu et al. (2008) proposed an Or-opt NSGA-II to minimize the number of vehicles, the total distance, and the constraint violation; Castro et al. (2009) considered the two common objectives, which were the time window violation and the capacity violation; Müller (2010) used the ε-constraint method to address the two objectives, which were the cost (a weighted sum of the number of vehicles and the total distance) and the constraint violation.

4. Knowledge-based evolutionary algorithm (KBEA)

In this section, we will detail the proposed algorithm, KBEA. It follows the typical flow of an EA. Chromosome representation will be presented in Section 4.1, and the algorithm parameter settings will be given in Section 5.2.

Step 1. Generate the initial population of N_P individuals (Section 4.2). Set the generation number to $t = 1$.

Step 2. Generate N_P offspring. Set $i = 0$.

Step 2.1 Select two parents by 2-tournament mating selection (Section 4.3).

Step 2.2 Generate two offspring by crossover (Section 4.4) and relocation mutation (Section 4.5).

Step 2.3 If the offspring are dominated by both parents, then perform split mutation (Section 4.5).

Step 2.4 Set $i = i + 2$. If $i = N_P$, then go to Step 3; otherwise, go back to Step 2.1.

Step 3. Select the best N_P individuals from the original population and offspring (Section 4.3). If there are duplicate individuals, then perform split mutation.

Step 4. Set $t = t + 1$. If $t = N_G$, then stop; otherwise, go back to Step 2.

4.1 Chromosome representation

To run a metaheuristic, the first and perhaps most important step is to determine the solution encoding scheme. In EA, this step is also known as the chromosome representation. In the literature on VRPTW, chromosome representations include the permutation representation (Ombuki et al. 2006, Labadi et al. 2008), sector representation (Muňoz-Zavala et al. 2009), and direct representation (Garcia-Najera and Bullinaria 2011, Hsu and Chiang 2012). The permutation representation encodes a sequence of customers on a chromosome. Ombuki et al. (2006) decoded a chromosome to a VRPTW solution by appending the customers one by one at the end of the currently constructed route in the order that the customers appear in the sequence. Labadi et al. (2008) extended the Split procedure in Prins (2004) to consider time windows. The Split procedure can partition the sequence of customers into multiple routes and find the solution that has the shortest total distance under this sequence. The sector representation mainly records a set of pairs of polar angles. Taking the depot as the center point, a pair of polar angles corresponds to two partition lines that start from the depot. Muňoz-Zavala et al. (2009) assigned customer locations within a pair of partition lines to a vehicle and allow vehicles to serve customers in the non-decreasing order of the end time of their service time windows.

In this study, we adopt a direct representation for two reasons. First, it encodes all of the information of the solution directly onto the chromosome and requires no extra decoding action and computational effort. Second, the complete information helps to design guided genetic operators. We can count the number of served customers for each route and calculate the average distance by the ratio of the total distance to the number of served customers. Then, we can exchange the shortest average-distance routes between parents or remove the route that has the smallest number of customers. In our implementation, we use a list of sub-lists to represent the solution. One sub-list corresponds to one route. The depot is omitted because it always serves as the start and end points.

Fig. 2 provides an illustration.

<< Insert Fig. 2 about here >>

4.2 Initialization

The initial population determines where we start to search at the beginning of an EA. Starting from totally random locations could require a long computation time to find the promising regions and solutions, but concentrating on only a few regions could also get the search process stuck at a local optimum. A popular and effective way to initialize the population is to combine several heuristically constructed solutions and simple random solutions. In our approach, we include two heuristic solutions in the initial population.

The first heuristic solution is constructed by the time-oriented nearest-neighbor heuristic (Solomon 1987). Starting with an empty route, a cost is calculated for each unrouted customer assuming that he/she is appended at the end of the route. The cost to append a customer j after a customer i is defined by

$$c_{ij} = \delta_1 d_{ij} + \delta_2(b_j - (b_i + s_i)) + \delta_3(l_j - (b_i + s_i + t_{ij})). \quad (1)$$

When the route is empty, the cost of adding a customer j into the route is calculated by c_{0j}. Simply speaking, the cost is a weighted sum of distances, the time difference between the beginning of the service at customer j and the completion of the service at customer i, and the urgency of the service. The unrouted customer that has the smallest cost is appended at the end of the route. In other words, the customers that are closer to the current ending customer i in the distance ($\delta_1 d_{ij}$) and time ($\delta_2(b_j - (b_i + s_i))$) and that are urgent ($\delta_3(l_j - (b_i + s_i + t_{ij}))$) are prioritized. Only feasible insertions are considered. When no unrouted customer can be appended without violating any constraint, a new route is created. We keep the sum of the three weights at 1 ($\delta_1 + \delta_2 + \delta_3 = 1$) and adjust the values by a unit of 0.1 (δ_i = 0, 0.1, 0.2, …, 1.0, $\forall$ i = 1, 2, 3.). Thus, we construct $H_3^{10} = C_2^{12} = 66$ solutions and take the best one as the initial solution.

The second heuristic solution is generated by Solomon's I1 heuristic (Solomon 1987). That heuristic starts by selecting from the unrouted customers the one that is farthest from the depot.

Then, the cost is calculated for each unrouted customer at each possible insertion position in the current route. The cost to insert a customer *u* between customers *i* and *j* is defined by

$$c(i, u, j) = \alpha_1(d_{iu} + d_{uj} - d_{ij}) + \alpha_2(b_j' - b_j) - \lambda d_{0u}. \qquad (2)$$

The symbol b_j' denotes the service start time at customer *j* after the insertion of customer *u*. Similarly, the cost is a weighted sum of the distance and time delay. The customer that has the smallest cost is inserted into the corresponding best position. The distance d_{0u} is introduced to prefer inserting customers that are located at a distance from the depot earlier. This approach could avoid creating direct routes from the depot to reach those customers. We keep the sum of two weights (α_1 and α_2) as 1 and adjust the values by a unit of 0.1. We test two values, 1 and 2, for λ. Thus, we construct $2 \cdot H_2^{10} = 2 \cdot C_1^{11} = 22$ solutions and take the best solution as another initial solution.

The remaining individuals are generated randomly. We order the customers randomly and append them one by one at the end of the route. When no unrouted customer can be appended without violating any constraint, a new route is created.

4.3 Selection

There are two selection steps in an EA. The mating selection chooses parents to perform crossover and mutation to generate the offspring, and the environmental selection decides which individuals will survive to the next generation. We use the two selection steps in NSGA-II in our algorithm. A difference is that we allow the offspring that are produced by crossover and mutation to enter the population temporarily and to be candidates of parents. This procedure is motivated by the idea of immediate replacement in a multiobjective differential evolution algorithm (DEMO, Robič and Filipič 2005). High-quality offspring can produce offspring immediately and improve the performance. Individuals are assigned Pareto ranks and crowding distances. The individuals that are not dominated by any other in the population are assigned rank 1. Disregarding individuals with rank *r* or smaller ranks, the individuals that are not dominated by any other in the remaining population are assigned rank (*r*+1). Individuals of the same rank are further evaluated by the crowding distance, which measures the density of neighboring solutions in the objective space. For the detailed calculation, please refer to the original paper. An individual i is better than an individual j if (1) i has a smaller rank than j or (2) i and j have the same rank and i has a larger crowding distance.

We use a 2-tournament for the mating selection. Two individuals are selected randomly, and the better one serves as a parent. We use the ($\mu + \lambda$) strategy for the environmental selection ($\mu = \lambda = N_P$) . Through $N_P/2$ times of mating selection, crossover, and mutation, N_P offspring are generated. Among the $2 \cdot N_P$ individuals, we choose the better N_P individuals to continue the evolution process again, according to their ranks and crowding distances.

4.4 Crossover

Crossover is responsible for exchanging genetic features between selected parents. Given VRPTW solutions, it is intuitive to exchange (partial) routes between the parents. Our crossover operator is an enhanced version of the operator in Tan et al. (2006). In the original version, the first step is to copy the parents to be the offspring. Then, the single best route is chosen from each offspring. Here, the best route refers to the route that has the shortest average distance, i.e., the smallest ratio of total distance to the number of customers. Next, the best route of one offspring is added into the other offspring. Customers in the newly added route will appear twice in the solution, and we remove them from the old routes. Because the newly added route is from a feasible solution, it is certainly feasible. For the modified routes, removing customers cannot cause a violation in the vehicle capacity and time window constraints. Thus, the offspring is also feasible. The crossover operator is simple, efficient, and effective.

One weakness of the original crossover is that the produced offspring often have one more route than the parents. (There will be one more route when the customers in the new route are located in different old routes.) To reduce the number of vehicles, we remove the worst route (the route that has the longest average distance) and very short routes (the routes that have one or two customers). The customers in these routes are re-inserted into the least-extra-distance position among all of the feasible positions. The extra distance for inserting a customer k between customers i and j is calculated by ($d_{ik} + d_{kj} – d_{ij}$).

The idea behind exchanging the best route (rather than a random route) is to allow the offspring to inherit good features from the parents explicitly. Through exchanging the best route, however, some individuals in the population could have the identical best route, especially toward the end of evolution. In this condition, the crossover has no effect and the offspring are identical to the parents. Thus, we add small randomness in the selection. One of the best N_C routes is selected to be exchanged. The effect of the value of N_C on the performance will be examined in Section 5.2.

4.5 Mutation

Many mutation operators have been proposed in the literature on VRPTW. Common operators include relocating one or multiple customers, swapping one or a sequence of customers, reversing a sequence of customers, reordering a sequence of customers, and route splitting/merging. Two mutation operators are used in our KBEA. The first operator, relocation mutation, is based on customer relocation because it is the most basic form of all of the operators. As the evolution proceeds, individuals become better and better, which means that the number of vehicles and the total distance are getting smaller. This approach also implies that the routes become more and more compact and that it is harder to find feasible positions to relocate the customers. Thus, we aim to remove a substantial number of customers simultaneously to create more space. Then, these removed customers are reinserted one by one. Considering the order of reinserting these customers, Ropke and Pisinger (2006) proposed the regret-k heuristic. It calculated for each customer the insertion cost of the best feasible position in each route. When k is one, the heuristic chose the customer whose insertion cost in the best route is the smallest. When k is greater than one, the heuristic chose the customer whose sum of cost difference between inserting him/her into the best route and the next (k – 1) best routes is the largest. Simply speaking, it chooses the customer that we will regret most if he/she is not inserted now. In our algorithm, we propose a rather simple method. We think that it would be more difficult to find a feasible insertion position for the customer that has a smaller time window. Hence, we reinsert these removed customers one by one in non-decreasing order of the length of the time window. In our algorithm, both of the offspring that are generated by crossover undergo mutation. The number of customers to be removed and reinserted is denoted by N_M, and the customers are selected randomly. The effect of its value on the performance will be investigated in Section 5.2. We will also examine different insertion orders in Section 5.3.

To create opportunities for customer relocation, our second mutation operator, split mutation, selects one route randomly and splits that route into two routes. Although it increases the number of vehicles, it also creates space for customer relocation. From the perspective of a search process, applying the split mutation helps to escape from the local optimum by moving to a worse solution temporarily. To avoid abuse of the split mutation and a careless increase in the number of vehicles, we use the split mutation with two guidelines. First, it is applied to an offspring only when the offspring is dominated by both parents. Second, we do not split the routes at a random point. We check the link distance along the route and break the link with the longest distance.

5. Experiments and results

5.1 Benchmark instances and algorithms

In the literature on VRPTW, Solomon’s (1987) 100-customer problem set has been taken as a standard benchmark. The problem set consists of 56 instances, which are categorized into six groups based on the distribution of customer locations and the lengths of the time windows. Customers’ locations are distributed randomly in the R category, and they are clustered in the C category. The RC category is a mix of the above two distributions. In each of the R, C, and RC categories, sub-category 1 consists of instances that have short time windows while sub-category 2 consists of instances that have long time windows. Because the 17 instances of the C category are easy to solve and almost all of the algorithms can solve them well, we consider only the remaining 39 instances in our experiment.

The research on solving VRPTW by searching for the Pareto set is still young, and there are few existing approaches. We take Ombuki et al. (2006), Ghoseiri et al. (2010), and Garcia-Najera and Bullinaria (2011) as benchmark algorithms. All of them and our algorithm are based on MOEA. In addition, we take three algorithms that are dedicated to the lexicographical minimization (the number of vehicles first, the total distance second) and three algorithms that are dedicated to distance minimization as benchmarks to verify the quality of the best solutions with respect to each objective.

5.2 Parameter analysis

In addition to the two standard parameters of an EA, the population size (N_P) and generation number (N_G), our algorithm has two more parameters, the number of candidate routes (N_C) to be exchanged in the crossover operator and the number of customers (N_M) to be reinserted in the mutation operator. Note that one of the N_C routes is chosen randomly and then exchanged. The product of N_P and N_G determines roughly the number of solutions that are generated during the search process. In general, the more solutions that are generated, the higher the chance that high-quality solutions can be found. The setting ($N_P \times N_G$) in the three benchmark MOEAs are 350×300 (Ombuki et al. 2006), 100×700 (Ghoseiri and Ghannadpour 2010), and 100×500 (Garcia-Najera and Bullinaria 2011). We followed the setting in Garcia-Najera and Bullinaria (2011), which generates the smallest number of solutions among the tested algorithms.

To examine the effect of the two parameters in our proposed crossover and mutation operators, we tested six values for each parameter. The tested values of N_C were 1, 2, …, and 6, and the tested values of N_M were 10, 20, …, and 60. Thus, thirty-six variants of KBEA were tested. Each variant solved each problem instance 10 times. Let V denote the set of compared algorithm variants, and let A_{ij} denote the net set of non-dominated solutions that is obtained by variant i over 10 runs for problem instance j. We evaluated the performances of these variants by the average coverage ratio. The coverage $C(i, j)$ of the algorithm variant i for problem instance j is defined by

$$C(i,j) = \frac{\left|\{s \mid s \in A_{kj}, k \in V/\{i\} \wedge \exists s' \in A_{ij} \text{ such that } s' \prec s\}\right|}{\sum_{k \in V/\{i\}} \left|A_{kj}\right|} \tag{3}$$

In (3), then $s' \prec s$ means that the solution s' found by the algorithm variant i can dominate the solution s found by another algorithm variant. Fig. 3 illustrates how to calculate the coverage of the three algorithm variants. The solutions of algorithm variant 1 can dominate 3 of 4 solutions that are obtained by variants 2 and 3. Therefore, the coverage of variant 1 is 3/4. Similarly, we can calculate the coverage of variants 2 and 3 by 1/(2+3) and 0/(2+3), respectively. Table 1 summarizes the average coverage of each of 36 KBEA variants with respect to four problem categories. The last column presents the average value over the four categories. A higher value means a better solution quality.

<< Insert Fig. 3 about here >>

<< Insert Table 1 about here >>

We first check the proper value of N_M for each value of N_C. The best result for each problem category is marked by a gray color. With N_C equal to 1, for example, the best coverage among six values of N_M for problem category R1 is 66.2%. We can see that, in general, the proper range of the values of N_M is between 30 and 50. The findings indicate that the relocation of a small number of customers is not sufficient to find good solutions. When all four categories are considered (the last column), setting N_M to 30 achieves the best average performance for all six values of N_C. Next, we mark the best result among all of the pairs of N_C and N_M in bold. The best combination of (N_C, N_M) is (5, 30), (6, 50), (3, 30), and (5, 50) for problem categories R1, R2, RC1, and RC2, respectively. On average, the setting (3, 30) provides the best performance. We took this setting in the following experiments. One more observation is that setting N_C by 1 obtains the worst average quality. This finding justifies our idea of considering more routes to exchange in the crossover operator.

<< Insert Table 2 about here >>

To understand the effect of the N_M values on the evolutionary process, we calculated two measures: (1) the average number of common arcs between the selected parents and (2) the average number of individuals who have the same objective values. These two measures were used to observe the population diversity in the genotypic and phenotypic spaces. We collected the above information by running six KBEA variants (N_C = 3; N_M = 10, 20, …, 60) to solve one problem instance from each category once. Table 2 summarizes the results. First, we can see that both measures decrease as the value of N_M increases, which implies that the relocation of more customers in the mutation operator can lead to a higher population diversity. Second, we noticed that when we set N_M to 30, which was the best value that was found in the previous parameter tuning experiment, the average number of individuals that have the same objective value decreases to below one. This finding might give us a clue for designing an adaptive parameter control mechanism in the future. We also performed the same experiment on six KBEA variants with N_M as 30 and N_C from 1 to 6. Unfortunately, there was no significant difference between the two observed measures among the variants. Other measures should be examined to understand the effects of the values of N_C, and we leave this task to our future work.

5.3 Effects of the initial solutions, customer insertion order, and split mutation

After examining the parameter values, we want to know the effects of the proposed strategies in KBEA. In our relocation mutation operator, we reinsert customers in non-decreasing order of the length of the time window; in other words, the customer that has a shorter time window is inserted earlier. We ran another six KBEA variants with different insertion orders. We let H±L, H±U, and H±D denote the variants that insert customers in the order of considering the length (L) of the time window, the upper bound (U) of the time window, and the distance (D) from the depot. The symbol + (−) means that the insertion is in non-decreasing (non-increasing) order of the mentioned values. For example, H+L stands for our proposed strategy. The variant H+R represents the variant with a random customer insertion order. In addition, we tested two other variants, one with all of the initial solutions generated randomly (R+L) and one without using the split mutation. The values of N_C and N_M are 3 and 30, respectively. Table 3 summarizes the performance of the proposed strategy (H+L) and the other eight variants.

<< Insert Table 3 about here >>

Table 3 shows that our proposed strategy performs the best among all of the variants in the three problem categories. The variant that has random initial solutions (R+L) is in the second place, and the variant that does not use split mutation is in the third. Inserting customers in the reverse order of the length of the time window (H−L) causes a large performance degradation and ranks 8th among the nine variants. Using the upper bound of the time window (H±U) or the distance from the depot (H±D) to determine the insertion order does not lead to good performance. Those alternatives are even worse than using a random order (H+R).

5.4 Comparison with multiobjective approaches

Next, we compare KBEA with three benchmark MOEAs that are dedicated to MOVRPTW in the literature. Garcia-Najera and Bullinaria (2011) reported solutions for only 29 problem instances, and thus, we calculated the average coverage by two sets of instances, including 39 and 29 instances, respectively. In the upper part of Table 4, we list the average coverage of three algorithms using 39 instances, and in the lower part, we list the average coverage of four algorithms using 29 instances.

<< Insert Table 4 about here >>

The results show that KBEA outperforms three benchmark algorithms for all four categories except for in one case, where Garcia-Najera and Bullinaria (2011) performs slightly better than our algorithm (72% vs. 70.3%). On average, KBEA can cover 81.7% of the solutions that are found by the other three algorithms. The second best algorithm can cover only 56.3% of the solutions. Looking into the design of these algorithms, Ombuki et al. (2006) used the permutation representation of the solutions while the other three encoded the solutions directly on the chromosome. Given a chromosome, Ombuki et al. generated a solution by appending the customers at the ends of the routes in the order that the customers appear in the encoded permutation. This decoding procedure might not be good enough to generate high-quality solutions. Moreover, the crossover and mutation operators in KBEA are different from those in the benchmark algorithms. For the crossover operator, Ombuki et al. and Ghoseiri and Ghannadpour (2010) selected one route in one parent and then reinserted the customers in this route into the other parent. This operation appears to not follow the concept of crossover, and the offspring do not inherit features from the parents. Garcia-Najera and Bullinaria (2011) combined the routes from the parents to produce the offspring. However, the routes were selected randomly. We should focus on inheriting good routes from the parents and not only on taking routes randomly. For the mutation operator, Ombuki et al. proposed a route reversal mutation, which reversed no more than 3 customers. Ghoseiri and Ghannadpour took the sequence-based crossover in Potvin and Bengio (1996) as the mutation operator. Basically, two partial routes in two parents were connected. Garcia-Najera and Bullinaria relocated and exchanged customers in a partial sequence of one or two routes and then reinserted a single customer. The experimental results in Section 5.2 showed that small values (≤ 20) of N_M do not maintain sufficient population diversity and cannot provide good performance. The mutation operators in the benchmark algorithms involved only one or two partial sequences of customers on the routes. The insufficient mutation intensity could be the cause of the worse performance.

5.5 Average quality of extreme solutions

In the literature on solving the VRPTW, a common practice to report the algorithm performance is to provide the average number of vehicles and the average total distance of the best solutions for each problem category. Here, we compare our algorithm with nine existing algorithms in this standard way. Table 5 summarizes the results. The first three algorithms focus on lexicographical optimization, the next four focus on the total distance, and the last four are Pareto approaches. The Pareto approaches provided a set of non-dominated solutions for each problem instance, and we took the two extreme solutions, i.e., the solution with the minimal number of vehicles and the solution with the minimal total distance. Labadi et al. (2008) solved the VRPTW instances in two ways, and thus, we put it in both the first and second groups of algorithms.

<< Insert Table 5 about here >>

While Table 5 gives detailed results, Fig. 4 helps to visualize the algorithm performance easily. The square symbols represent the average number of vehicles and the average total distance of the two extreme solutions found by our KBEA. The asterisk symbols represent the results that were obtained by the other nine algorithms. Fig. 4 shows that in all of the problem categories, our results are not dominated by any other algorithm. Considering lexicographical optimization of the number of vehicles and total distance, Nagata et al. (2010) is always the best, and both Lim and Zhang (2007) and Nagata et al. (2010) are better than KBEA. We are the second best in terms of the number of vehicles and the total distance (the best in terms of the total distance in the R1 category), which shows the ability to achieve good performance in both objectives simultaneously. In categories R1 and RC1, our results dominate 9 of 13 points; in categories R2 and RC2, the advantage is smaller, but we still dominate 6 points.

<< Insert Fig. 4 about here >>

From Table 5, we see that KBEA dominates the three distance-only algorithms for three problem categories and dominates Labadi et al. (2008) for two categories. Although these algorithms were dedicated to minimizing the total distance, KBEA improves their results by having not only a shorter distance but also fewer vehicles. Compared with the three MOEA benchmarks, KBEA's results dominate all of the results from Ghoseiri et al. (2010) and 7 of 8 results from Ombuki *et al*. (2006). We also dominate Garcia-Najera and Bullinaria's results (2011) for the R1 and RC1 categories regarding the minimal number of vehicles and for the RC1 and RC2 categories regarding the minimal total distance.

Table 6 summarizes the average computation time, the number of runs to obtain the best results, and the computing environment for the compared algorithms for reference. Because the CPU speed and implementation language are different, we cannot compare the computational efficiency directly. However, KBEA shows the ability to produce high-quality solutions efficiently (within a half minute).

<< Insert Table 6 about here >>

5.6 Net set of non-dominated solutions

To facilitate the performance comparison between the Pareto algorithms for the MOVRPTW, we compile the net set of the non-dominated solutions for the 39 tested problem instances based on the results from the nine past algorithms and KBEA. In Table 7, we list the non-dominated solutions for each problem instance and the algorithms that found them. Among the 109 solutions, KBEA found 36 solutions, which is much more than Ombuki et al.'s (2006) 4 solutions, Ghoseiri et al.'s (2010) 3 solutions, and Garcia-Najera and Bullinaria's (2011) 7 solutions. We also update the minimal total distance for the 10 problem instances. The results show that almost all of the tested problem instances (except for RC103 and RC104) have conflicts between the number of vehicles and the total distance.

<< Insert Table 7 about here >>

5.7 Tests on larger-scale problem instances

In the literature on the VRPTW, Solomon's problem set has been intensively studied. Another standard problem set used in the VRPTW studies was proposed by Gehring and Homberger (1999). Hereafter, their instances of *x* customers will be called GH-*x* instances. They followed Solomon's method of generating five sets of problem instances with the number of customers extended to 200, 400, 600, 800, and 1000, respectively. Each set consists of six categories (similar to Solomon's set), and each category consists of ten problem instances. In this subsection, we will examine the performance of KBEA in solving GH-200 instances. To the authors' best knowledge, this paper is the first attempt to solve GH instances by a Pareto approach.

The parameter setting of KBEA was determined by a simple procedure. We kept the population size at 100 and changed the values of N_M and N_C one at a time. We tested three values (30, 50, and 70) for N_M and three values (3, 6, and 9) for N_C on the first two instances in the R1 category. Then, we determined to set N_M at 30 and N_C at 6. The generation number was set to be 4000. We solved each instance 10 times. Because there is no other Pareto approach that solves these instances, we compare KBEA with two recent algorithms (Nagata et al. 2010 and Vidal et al. 2013) that provided very promising results. Both benchmark algorithms are based on MA and solve the VRPTW by lexicographical optimization. Table 8 presents the average number of vehicles and the average total distance in each of the six problem categories for the three tested algorithms. Again, because KBEA provides multiple solutions for each instance, we list the average value for the two extreme solutions. The data in bold are not dominated by other algorithms.

<< Insert Table 8 about here >>

In Table 8, we find that KBEA is not sufficiently good at solving instances in the R1 and RC1 categories. The average solution quality is dominated by the two benchmark algorithms. In category C2, KBEA is slightly worse by 0.17% extra total distance. For the other three categories, however, KBEA can find many non-dominated solutions. The net set of non-dominated solutions is given in Table 9. Among the 115 non-dominated solutions, KBEA finds 64 solutions, and 55 solutions are newly found. An interesting finding is that Solomon's C1 and C2 instances were shown to have no conflict between the number of vehicles and the total distance (Tan et al. 2006, Garcia-Najera and Bullinaria 2011), but GH-200 instances in the C1 and C2 categories are shown to have conflicting objective values in our experiments. The average computation times and computing environments are given in Table 10 for reference.

<< Insert Table 9, 10 about here >>

We also applied KBEA to solve GH-400 instances by using the same parameter settings, but we extended the generation number to 6000. However, the results are not satisfactory. Only a few non-dominated solutions were found. This finding reveals that there is a large amount of room for enhancing KBEA's ability to solve large-scale instances. We will continue this research direction in our future studies.

6. Conclusions

The VRPTW has been intensively studied in recent decades due to its challenging problem complexity and high practical value. In addition to focusing on one (primary) objective, such as the minimization of the number of vehicles, recent research has started to address multiple objectives simultaneously. Pareto approaches aim at finding the set of optimal solutions, to provide managers more choices and to better understand the trade-off between the objectives that are of interest. In this study, we proposed a simple, effective, and efficient MOEA to minimize the number of vehicles and the total distance. This approach has only four parameters, and genetic operators are easy to implement. Incorporation of problem-specific knowledge into these simple operators helps the algorithm to find high-quality solutions.

Our future work aims to make our algorithm simpler and more effective. First, we plan to develop an adaptive parameter control mechanism to simplify the parameter tuning process. Second, we want to follow Nagata et al. (2010) and Vidal et al. (2013) to incorporate local search procedures and to allow the participation of infeasible solutions in the evolution process to improve the solution quality for larger-scale problem instances. Many research opportunities exist that involve methods for adding these techniques without making the algorithm too complicated to use and for setting the parameters. In addition to the algorithmic improvement, combining our optimization algorithm and Google Maps-based interface into a web service will be of practical use.

Acknowledgment

The authors are grateful to the editor and the reviewers for their valuable suggestions and comments.

References

Alvarenga, G.B., Mateus, G.R., de Tomi, G. (2007). A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows. *Computers & Operations Research* 34(6), 1561–1584.

Barán, B. Schaerer, M. (2003). A multiobjective ant colony system for vehicle routing problem with time windows. IASTED International Conference on Applied Informatics, pp. 97–102.

Bent, R., Hentenryck, P.V. (2004). A two-stage hybrid local search for the vehicle routing problem with time windows. *Transportation Science* 38(4), 515–530.

Berger, J., Barkaoui, M., Bräysy, O. (2003). A route-directed hybrid genetic approach for the vehicle routing problem with time windows. *Information Systems and Operational Research* 41, 179–194.

Bouthillier, A.L., Crainic, T.G. (2005). A cooperative parallel meta-heuristic for the vehicle routing problem with time windows. *Computers & Operations Research* 32(7), 1685–1708.

Bräysy, O. (2003). A reactive variable neighborhood search for the vehicle routing problem with time windows. *Journal on Computing* 15(4), 347–368.

Bräysy, O., Hasle, G., Dullaert, W (2004). A multi-start local search algorithm for the vehicle routing problem with time windows. *European Journal of Operational Research* 159, 586–605.

Bräysy, O., Gendreau, M. (2005a). Vehicle routing problem with time windows, Part I: Route construction and local search algorithm. *Transportation Science* 39(1), 104–118.

Bräysy, O., Gendreau, M. (2005b). Vehicle routing problem with time windows, Part II: Metaheuristics. *Transportation Science* 39(1), 119–139.

Castro, J.P., Landa-Silva, D., Pérez, J.A.M. (2009). Exploring feasible and infeasible regions in the vehicle routing problem with time windows using a multi-objective particle swarm optimization approach. In N. Krasnogor et al. (Eds.) *Nature Inspired Cooperative Strategies for Optimization* (pp. 103–114), Springer-Verlag, Berlin.

Cordeau, J.F., Laporte, G., Mercier, A. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society* 52(8), 928–936.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T. (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6(2), 182–197.

Gambardella, L.M., Taillard, E., Agazzi, G. (1999). MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. In D. Corne, M.F. Dorigo, & F. Glover (Eds.), *New Ideas in Optimization* (pp.63–76). McGraw-Hill, London, UK.

Garcia-Najera, A., Bullinaria, J.A. (2011). An improved multi-objective evolutionary algorithm for the vehicle routing problem with time windows. *Computer & Operations Research* 38(1), 287–300.

Gehring, H., Homberger, J. (1999). A parallel hybrid evolutionary metaheuristics for the vehicle routing problem with time windows. EUROGEN, pp. 57–64.

Geiger, M.J. (2001). A genetic algorithm applied to multiple objective vehicle routing problems. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.59.8901

Ghoseiri, K. Ghannadpour, S.F. (2010). Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm. *Applied Soft Computing* 10(4), 1096–1107.

Glover, F. (1992). New ejection chain and alternating path methods for traveling salesman problems. In O. Balci et al. (Eds.) *Computer Science and Operations Research: New Developments in Their Interfaces*. Pergamon Press, Oxford, UK, 449–509.

Homberger, J., Gehring, H. (1999). Two evolutionary metaheuristics for the vehicle routing problem with time windows. *Information Systems and Operational Research* 37, 297–318.

Homberger, J., Gehring, H. (2005). A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. *European Journal of Operational Research* 162(1), 220–238.

Hong, S.C., Park, Y.B. (1999). A heuristic for a bi-objective vehicle routing with time window constraints. *International Journal of Production Economics* 62(3), 249–258.

Hsu, W.H., Chiang, T.C. (2012). A multiobjective evolutionary algorithm with enhanced reproduction operators for the vehicle routing problem with time windows. IEEE Conference on Evolutionary Computation (CEC).

Jepsen, M., Spoorendonk, S., Petersen, B., Pisinger, D. (2006). A non-robust branch-and-cut-and-price algorithm for the vehicle routing problem with time windows. Technical Report no. 06/03 ISSN: 0107-8283. Department of Computer Science, University of Copenhagen.

Jozefowiez, N., Semet, F., Talbi, E. G. (2008). Multi-objective vehicle routing problems. *European Journal of Operational Research* 189(2), 293–309.

Labadi, N., Prins, C., Reghioui, M. (2008). A memetic algorithm for the vehicle routing problem with time windows. *RAIRO-Operations Research* 42(3), 415–431.

Lim, A., Zhang, X. (2007). A two-stage heuristic with ejection pools and generalized ejection chains for the vehicle routing problem with time windows. *Journal on Computing* 19(3), 443–457.

Müller, J. (2010). Approximative solutions to the bicriterion vehicle routing problem with time windows. *European Journal of Operational Research* 202(1), 223–231.

Muňoz-Zavala, A., Hernández-Aguirre, A., Villa-Diharce, E. (2009) Particle evolutionary swarm multi-objective optimization for vehicle routing problem with time windows. In C.A. Coello Coello et al. (Eds.) *Swarm Intelligence for Multi-objective Problems* (pp. 233–257) Springer-Verlag, Berline.

Nagata, Y., Bräysy, O. (2009) A powerful route minimization heuristic for the vehicle routing problem with time windows. *Operations Research Letters* 37(5), 333–338.

Nagata, Y., Bräysy, O., Dullaert, W. (2010). A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research* 37(4), 724–737.

Ombuki, B., Ross, B.J., Hanshar, F. (2006). Multi-objective genetic algorithms for vehicle routing problem with time windows. *Applied Intelligence* 24(1), 17–30.

Potvin, J.Y., Kervahut, T., Garcia, B.L., Rousseau, J.M. (1996). The vehicle routing problem with time windows, Part I: Tabu search. *Journal on Computing* 8(2), 158–164.

Potvin, J.Y., Bengio, S. (1996). The vehicle routing problem with time windows, Part II: Genetic search. *Journal on Computing* 8(2), 165–172.

Potvin, J.Y. (2009). Evolutionary algorithms for vehicle routing. *INFORMS Journal on Computing* 21(4), 518–548.

Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research* 31(12), 1985–2002.

Robič, T. and Filipič, B. (2005). DEMO: Differential evolution for multiobjective optimization, *Proceedings of International Conference on Evolutionary Multicriterion Optimization*, 520–533.

Ropke, S. and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows, *Transportation Science* 40(4), 455–472.

Solomon, M.M. (1987). Algorithms for the vehicle routing and scheduling problem. *Operations Research* 35(2), 254–265.

Tan, K.C., Lee, L.H., Zhu, Q.L., Ou, K. (2001). Heuristic methods for vehicle routing problem with time windows. *Artificial Intelligence in Engineering* 15(3), 281–295.

Tan, K.C., Lee, T.H., Chew, Y.H., Lee, L.H. (2003). A multiobjective evolutionary algorithm for solving vehicle routing problem with time windows. IEEE International Conference on Systems, Man, and Cybernetics, vol. 1, pp. 361–366.

Tan, K.C., Chew, Y.H., Lee, L.H. (2006). A hybrid multiobjective evolutionary algorithm for solving vehicle routing problem with time windows. *Computational Optimization and Applications* 34(1), 115–151.

Ting, C.J., Huang, C.H. (2005). An improved genetic algorithm for vehicle routing problem with time windows. *International Journal of Industrial Engineering* 12(3), 216–226.

Vidal, T., Crainic, T.G., Gendreau, M., and Prins, C. (2013). A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research* 40(1), 475–489.

Watanabe, S., Sakakibara, K. (2007). A multiobjectivization approach for vehicle routing problems. *Lecture Notes in Computer Science* 4403, 660–672.

Xu, H., Fan, W., Wei, T., Yu, L. (2008). An Or-opt NSGA-II algorithm for multi-objective vehicle routing problem with time windows. IEEE Conference on Automation Science and Engineering, pp. 309–314.

Yu, B., Yang, Z.Z., Yao, B.Z (2011). A hybrid algorithm for vehicle routing problem with time windows. *Expert Systems with Applications* 38(1), 435–441.

Figures

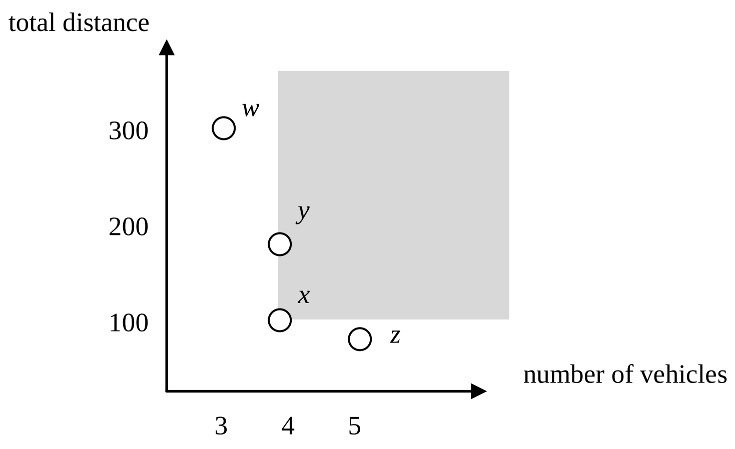

Fig. 1. Solutions to the MOVRPTW on the objective space (solution *x* dominates all of the solutions in the grey region)

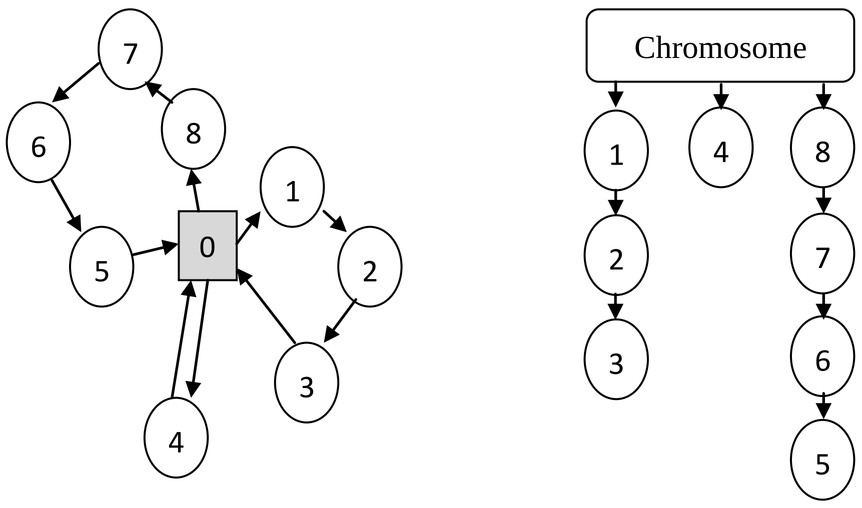

Fig. 2. Chromosome representation

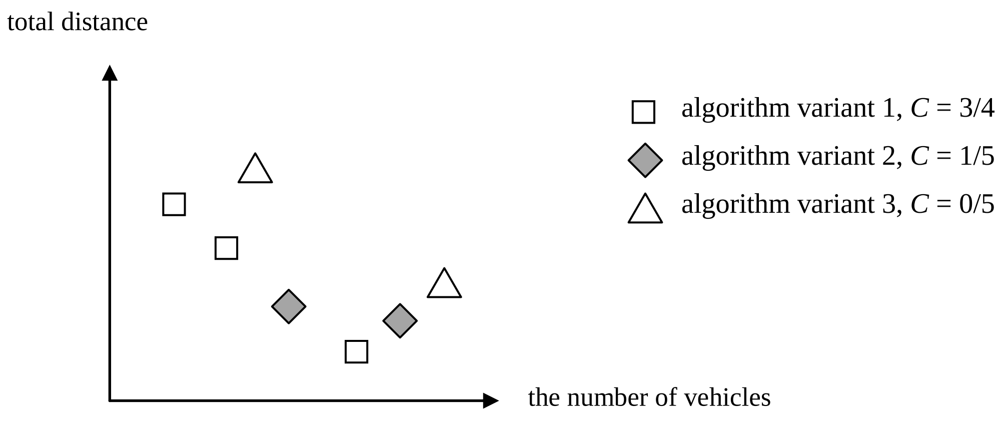
total distance
the number of vehicles
algorithm variant 1, C = 3/4
algorithm variant 2, C = 1/5
algorithm variant 3, C = 0/5

Fig. 3 Illustration of the coverage metric

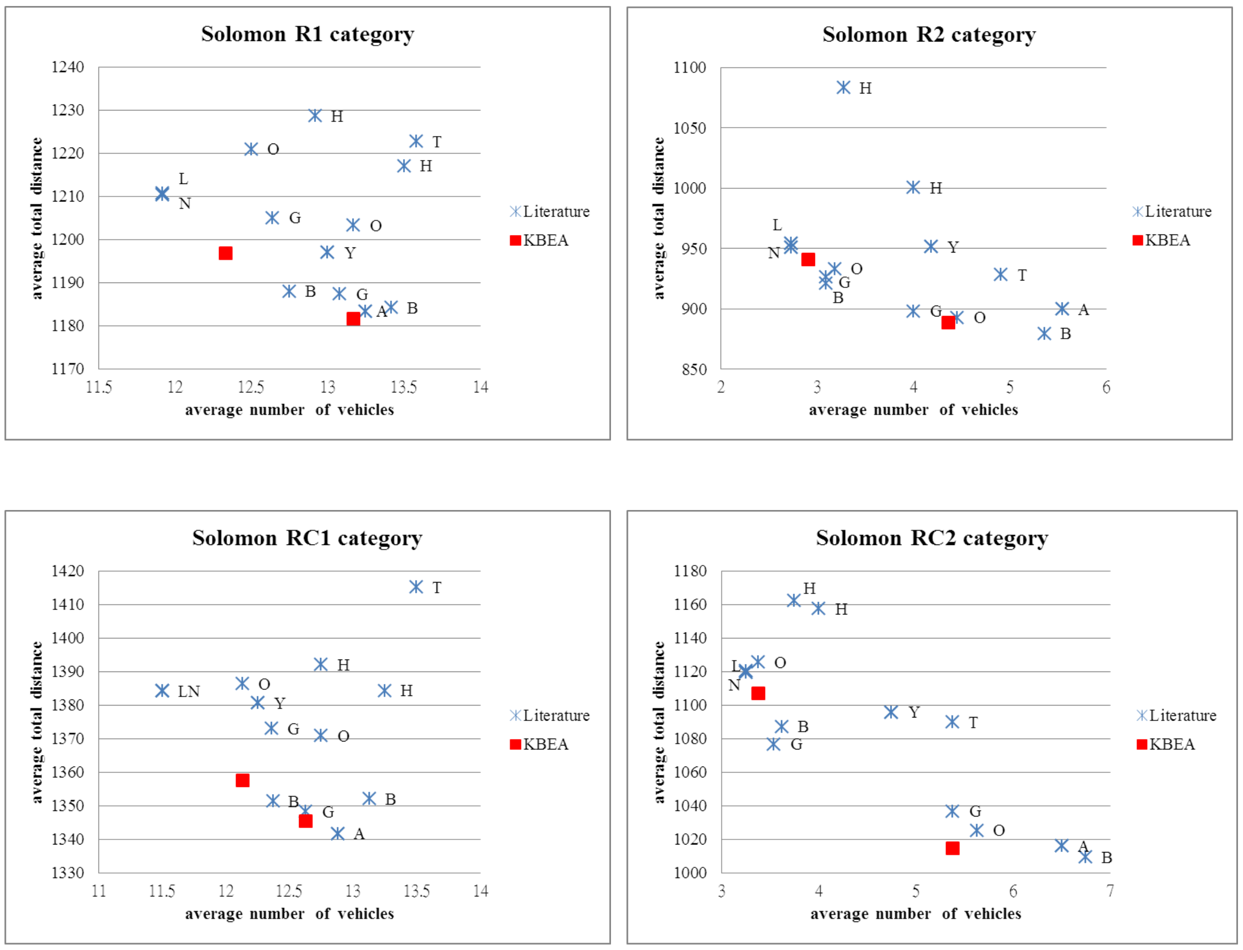

Fig. 4 Average performance of 10 algorithms for four problem categories (Solomon 100-customer instances)

Tables

Table 1 Average coverage (× 100) for KBEA variants with different N_C values in crossover and N_M values in relocation mutation for each problem category

N_C	N_M	R1	R2	RC1	RC2	Avg
1	10	36.8	18.6	51.7	8.5	28.9
	20	57.8	22.1	67.4	23.1	42.6
	30	66.2	37.2	74.3	51.4	57.3
	40	61.8	45.7	56.5	51.0	53.7
	50	31.9	59.4	33.0	45.8	42.5
	60	5.3	40.1	11.5	37.4	23.5
2	10	36.8	11.8	50.9	7.9	26.8
	20	58.6	34.9	70.5	34.5	49.6
	30	68.7	54.8	74.8	53.3	62.9
	40	64.2	59.5	55.0	60.1	59.7
	50	33.7	65.8	38.4	52.3	47.6
	60	9.0	41.1	16.8	39.3	26.5
3	10	44.2	18.7	49.5	14.1	31.6
	20	64.5	38.9	76.5	30.7	52.6
	30	74.4	60.2	**78.1**	56.4	**67.3**
	40	61.1	61.0	52.2	58.7	58.3
	50	31.4	62.6	32.2	52.5	44.7
	60	7.7	37.4	9.7	41.0	23.9
4	10	44.9	19.0	61.3	15.9	35.3
	20	57.5	39.8	70.2	34.4	50.5
	30	74.2	61.5	76.1	56.4	67.0
	40	63.2	59.9	56.5	57.3	59.2
	50	32.6	61.3	37.2	49.0	45.0
	60	9.2	42.7	12.4	47.6	28.0
5	10	36.4	18.4	49.6	22.3	31.7
	20	65.5	39.8	69.1	32.7	51.8
	30	**74.7**	56.3	76.4	55.1	65.6
	40	69.4	65.2	58.7	61.2	63.6
	50	33.7	59.0	35.9	**62.5**	47.8
	60	7.6	49.1	14.3	46.5	29.4
6	10	44.9	21.5	50.8	12.5	32.4
	20	59.4	30.1	76.0	27.1	48.1
	30	73.2	52.3	77.8	57.7	65.3
	40	61.2	65.8	54.5	61.0	60.6
	50	30.6	**67.7**	35.1	55.9	47.3
	60	8.9	45.6	14.5	42.4	27.8

Table 2 Average number of common arcs and average number of individuals with the same objective values for KBEA variants with different N_M values in relocation mutation for four problem instances

	Average number of common arcs				Average number of individuals with the same objectives			
N_M	R101	R201	RC101	RC201	R101	R201	RC101	RC201
10	107.72	71.53	104.10	79.00	14.63	5.95	27.35	7.07
20	99.55	72.86	96.84	67.37	3.93	3.19	5.62	2.36
30	93.18	66.44	86.21	72.48	0.54	0.54	0.70	0.95
40	75.90	63.36	68.63	64.62	0.01	0.13	0.00	0.19
50	67.08	49.62	52.82	55.44	0.00	0.01	0.00	0.00
60	61.61	43.03	45.32	42.63	0.00	0.00	0.00	0.00

Table 3 Average coverage (× 100) for KBEA variants with different insertion orders in relocation mutation, with random initial solutions, and without split mutation.

Version	R1	R2	RC1	RC2	Avg
H+R	46.5	48.5	54.3	46.3	48.9
H+U	57.3	43.4	57.5	36.7	48.7
H–U	18.2	44.0	33.6	42.8	34.6
H+D	5.5	34.6	4.1	34.4	19.6
H–D	63.5	36.8	62.4	36.6	49.8
H+L	68.1	50.6	71.7	45.1	58.8
H–L	28.5	41.0	29.4	36.3	33.8
R+L	57.4	46.2	60.9	51.3	53.9
H+L (without split mutation)	54.2	43.2	59.4	42.3	49.8

First symbol: H: heuristic initial solutions, R: all random initial solutions

Second symbol: R: random order, U: time window upper bound, D: distance from the depot, L: time window length

(+: the smaller the earlier; –: the larger the earlier)

Table 4 Average coverage (× 100) for KBEA and three benchmark MOEAs (Solomon 100-customer instances)

Approach	R1	R2	RC1	RC2	Avg
Ombuki et al. (2006)	34.7	44.7	39.6	20.6	34.9
Ghoseiri and Ghannadpour (2010)	12.5	3.6	15.6	1.8	8.4
KBEA	95.1	87.6	89.6	91.7	91.0
Ombuki et al. (2006)	20.4	35.5	23.3	21.6	25.2
Ghoseiri and Ghannadpour (2010)	16.7	5.4	20.0	1.0	10.8
Garcia-Najera and Bullinaria (2011)	71.7	39.2	72.0	42.4	56.3
KBEA	85.0	82.3	70.3	89.2	81.7

When Garcia-Najera and Bullinaria (2011) is included in the set of compared algorithms, only 29 instances are considered because they provide their solutions for only these 29 instances.

Table 5 Average number of routes and average total distance of KBEA and 9 benchmark algorithms (Solomon 100-customer instances)

Approach	R1	R2	RC1	RC2
Lim and Zhang (2007)	11.92 1210.76	2.73 953.94	**11.50** **1384.17**	3.25 1120.40
Nagata *et al*. (2010)	**11.92** **1210.34**	**2.73** **951.03**	**11.50** **1384.17**	**3.25** **1119.24**
Labadi *et al*. (2008) (minimal number of vehicles)	**12.75** **1188.01**	**3.09** **920.86**	**12.37** **1351.27**	3.62 1087.18
Ting and Huang (2005)	13.58 1222.65	4.91 928.32	13.50 1415.16	5.38 1089.78
Alvarenga *et al*. (2007)	13.25 1183.38	5.55 899.90	**12.88** **1341.67**	6.50 1015.90
Labadi *et al*. (2008) (minimal total distance)	13.42 1184.16	**5.36** **879.51**	13.13 1352.02	**6.75** **1009.37**
Yu et al. (2011)	13.00 1196.96	4.18 951.36	12.25 1380.55	4.75 1095.84
Ombuki *et al*. (2006) (minimal number of vehicles)	12.50 1220.91	3.18 933.14	12.13 1386.35	3.38 1125.44
Ombuki *et al*. (2006) (minimal total distance)	13.17 1203.22	4.45 892.89	12.75 1370.84	5.63 1025.31
Ghoseiri et al. (2010) (minimal number of vehicles)	12.92 1228.60	3.27 1082.93	12.75 1392.09	3.75 1162.4
Ghoseiri et al. (2010) (minimal total distance)	13.50 1217.03	4.00 1000.22	13.25 1384.3	4.00 1157.41
Garcia-Najera and Bullinaria (2011) (minimal number of vehicles)	12.64 1205.04	3.09 926.17	12.36 1372.96	**3.54** **1076.72**
Garcia-Najera and Bullinaria (2011) (minimal total distance)	**13.08** **1187.32**	**4.00** **897.95**	12.63 1348.22	5.38 1036.65
KBEA (minimal number of vehicles)	**12.33** **1196.89**	**2.91** **940.61**	**12.13** **1357.47**	**3.38** **1106.99**
KBEA (minimal total distance)	**13.17** **1181.57**	**4.36** **888.36**	**12.63** **1345.38**	**5.38** **1014.81**

Data in bold are not dominated by the other algorithms. (Fig. 4 visualizes the data points in the objective space.)

Table 6 Average computation time, number of runs, and computing environments (Solomon 100-customer instances)

Approach	Average computation time (s.)				#runs	CPU (Language)
	R1	R2	RC1	RC2		
Lim and Zhang (2007)	1576.8				1	Pentium IV 2.8 GHz (Java)
Nagata *et al*. (2010)	300				5	Opteron 2.4 GHz (C++)
Labadi *et al*. (2008)	151.8	210.91	152.32	199.63	1	3 GHz (Delphi)
Alvarenga *et al*. (2007)	3600				3	–––
Yu et al. (2011)	698	655	317	407	10	Pentium 1 GHz (C++)
Ombuki *et al*. (2006)	–––				10	Pentium IV 1.6 GHz (Java)
Ghoseiri et al. (2010)	> 500	> 900	> 500	> 1300	10	1.6 GHz (Matlab)
Garcia-Najera and Bullinaria (2011)	–––				30	2218 2.6 GHz nodes
KBEA	11.7	22.7	10.4	19.1	10	Intel i7-3770 3.4 GHz (C++)

--- means that we do not find data in the original paper.

Table 7 Net set of non-dominated solutions for Solomon 100-customer instances

Problem instance	Number of routes	Total distance	Algorithms that found the solution
R101	19	1650.8	[L][N][G]
	20	1642.87	[A]
R102	17	1486.12	[L][N]
	18	1472.62	[A]
R103	13	1292.68	[L][N]
	14	1213.62	[B][A]
R104	9	1007.31	[L][N]
	10	974.24	[H]
R105	14	1377.11	[L][N][G]
	15	1360.78	[B][A][K]
R106	12	1252.03	[L][N]
	13	1239.98	[K]
R107	10	1104.66	[L][N]
	11	1074.24	[B]
R108	9	958.66	[Y]
	10	942.895	[K]
R109	11	1194.73	[L][N]
	12	1101.99	[Y]
R110	10	1118.84	[L][N]
	11	1086.82	[K]
	12	1072.42	[B]
R111	10	1096.73	[L][N]
	11	1054.23	[K]
	12	1053.5	[A][K]
R112	9	982.14	[N]
	10	960.675	[A]
R201	4	1252.37	[L][N]
	5	1193.29	[K]
	6	1171.2	[K]
	8	1150.92	[B]
	9	1148.48	[A]
R202	3	1191.7	[L][N]
	4	1079.39	[K]
	5	1041.1	[K]
	7	1037.5	[B]
R203	3	939.5	[L][N]
	4	901.201	[K]
	5	890.5	[O]
	6	874.869	[B]
R204	2	825.52	[N]
	3	749.417	[K]
	4	743.233	[K]
	5	735.861	[B]
R205	3	994.43	[L][N]
	4	959.74	[G][K]
	5	954.16	[O]
R206	3	906.14	[L][N][K]
	4	889.39	[O]
	5	879.893	[B]
R207	2	890.61	[N]
	3	812.755	[K]
	4	800.786	[B]
R208	2	726.82	[L][N]
	3	706.855	[B]
R209	3	909.16	[L][N]
	4	864.149	[K]
	5	859.39	[B]
R210	3	938.58	[H]
	4	924.785	[K]
	5	922.297	[K]
	6	912.533	[B]
R211	2	885.71	[N]
	3	778.041	[K]
	4	755.949	[B]
RC101	14	1650.14	[Y]
	15	1624.97	[K]
RC102	12	1554.75	[L][N]
	13	1477.54	[K]
	14	1461.33	[K]
RC103	11	1261.67	[L][N]
RC104	10	1135.48	[L][N][K]
RC105	13	1629.44	[L][N]
	14	1540.18	[G]
	15	1519.29	[K]
	16	1518.6	[A]
RC106	11	1424.73	[L][N]
	12	1394.43	[G]
	13	1377.35	[A]
RC107	11	1222.1	[H]
	12	1212.83	[B][A]
RC108	10	1139.82	[L][N]
	11	1117.53	[A]
RC201	4	1406.94	[L][N]
	5	1279.65	[Y]
	7	1273.51	[K]
	8	1272.28	[K]
RC202	3	1365.65	[L][N]
	4	1162.54	[G]
	5	1118.66	[K]
	8	1099.54	[B]
RC203	3	1049.62	[N]
	4	945.083	[K]
	5	926.819	[K]
RC204	3	798.46	[L][N]
	4	788.663	[K]
RC205	4	1297.65	[L][N]
	5	1236.78	[K]
	6	1187.98	[K]
	7	1161.81	[A]
RC206	3	1146.32	[L][N]
	4	1081.83	[K]
	5	1068.77	[K]
	7	1054.61	[B]
RC207	3	1061.14	[L][N]
	4	1001.85	[G]
	5	982.58	[O]
	6	966.372	[B]
RC208	3	828.14	[N]
	4	783.035	[K]

[L]: Lim and Zhang (2007): Table 3 [N]: Nagata *et al*. (2010): Table 7
[A]: Alvarenga *et al*. (2007): Table 5 [O]: Ombuki *et al*. (2006): Table 1, 2
[B]: Labadi *et al*. (2008) Table 3, 4 [Y]: Yu et al. (2011): Table 3
[G]: Garcia-Najera and Bullinaria (2011): Table 7
[H]: Ghoseiri and Ghannadpour (2010): Table 1 [K]: KBEA

Table 8 Average number of routes and average total distance of compared algorithms (Gehring and Homberger 200-customer instances)

Approach	R1	R2	C1	C2	RC1	RC2
Nagata *et al*. (2010)	18.2 3612.31	**4.0** **2929.41**	**18.9** **2718.41**	6.0 1831.64	**18.0** **3178.68**	4.3 2536.22
Vidal *et al*. (2013)	**18.2** **3613.16**	**4.0** **2929.41**	**18.9** **2718.41**	**6.0** **1831.59**	18.0 3180.48	**4.3** **2536.20**
KBEA (minimal number of vehicles)	18.2 3724.18	**4.2** **2920.34**	**19.2** **2703.44**	6.0 1834.74	18.4 3263.26	4.5 2557.26
KBEA (minimal total distance)	19.0 3670.07	**5.9** **2770.72**	**19.7** **2679.95**	6.3 1834.12	19.0 3225.19	**6.5** **2355.89**

Data in bold are not dominated by other algorithms.

Table 9 Net set of non-dominated solutions for Gehring and Homberger 200-customer instances

Problem instance	Number of routes	Total distance	Algorithms that found the solution
R1_2_1	20	4784.11	[N][V]
	21	4765.27	[K]
	22	4755.63	[K]
R1_2_2	18	4040.60	[V]
.	19	4017.08	[K]
R1_2_3	18	3381.96	[N][V]
R1_2_4	18	3057.81	[N][V]
R1_2_5	18	4107.86	[N][V]
R1_2_6	18	3583.14	[V]
R1_2_7	18	3150.11	[N][V]
R1_2_8	18	2951.99	[N][V]
R1_2_9	18	3760.58	[V]
R1_2_10	18	3301.18	[N][V]
R2_2_1	4	4483.16	[N][V]
	5	4085.11	[K]
	6	3850.93	[K]
	7	3765.72	[K]
	8	3713.20	[K]
	9	3681.37	[K]
R2_2_2	4	3621.20	[N][V]
	5	3439.6	[K]
	6	3328.95	[K]
	7	3250.73	[K]
R2_2_3	4	2880.62	[N][V]
	5	2812.79	[K]
	6	2720.70	[K]
R2_2_4	4	1981.30	[N][V]
	5	1980.08	[K]
R2_2_5	4	3366.79	[N][V]
	5	3311.86	[K]
	6	3260.93	[K]
R2_2_6	4	2913.03	[N][V]
	5	2843.82	[K]
R2_2_7	4	2451.14	[N][V]
	5	2437.8	[K]
	6	2411.22	[K]
R2_2_8	4	1849.87	[N][V]
R2_2_9	4	3092.04	[N][V]
	5	3026.73	[K]
	6	2958.67	[K]
R2_2_10	4	2654.97	[N][V]
RC1_2_1	18	3602.8	[V]
	19	3574.79	[K]
	20	3571.03	[K]
RC1_2_2	18	3249.05	[V]
RC1_2_3	18	3008.33	[N][V]
RC1_2_4	18	2851.68	[N][V]
RC1_2_5	18	3371.00	[V]
RC1_2_6	18	3324.80	[V]
RC1_2_7	18	3189.32	[N][V]
RC1_2_8	18	3083.93	[N][V]
RC1_2_9	18	3081.13	[N][V]
RC1_2_10	18	3000.30	[V]
RC2_2_1	6	3099.53	[N][V]
	7	2950.99	[K]
	8	2877.59	[K]
	9	2861.54	[K]
RC2_2_2	5	2825.24	[N][V]
	6	2641.21	[K]
	7	2543.62	[K]
RC2_2_3	4	2601.87	[N]
	5	2448.11	[K]
	6	2397.26	[K]
	7	2366.26	[K]
RC2_2_4	4	2038.56	[N][V]
	5	1961.38	[K]
	6	1899.99	[K]
RC2_2_5	4	2911.46	[N][V]
	5	2713.78	[K]
	6	2584.70	[K]
	7	2538.72	[K]
RC2_2_6	4	2873.12	[N][V]
	5	2703.64	[K]
	6	2586.54	[K]
	7	2577.23	[K]
RC2_2_7	4	2525.83	[N][V]
	5	2397.95	[K]
	6	2355.60	[K]
RC2_2_8	4	2292.53	[V]
	5	2237.01	[K]
	6	2225.06	[K]
	7	2220.88	[K]
RC2_2_9	4	2175.04	[N][V]
	5	2167.63	[K]
RC2_2_10	4	2015.61	[N][V]
C1_2_1	20	2704.57	[N][V][K]
C1_2_2	18	2917.89	[N][V]
	19	2796.73	[K]
	20	2700.65	[K]
C1_2_3	18	2707.35	[N][V]
	19	2700.82	[K]
	20	2682.18	[K]
C1_2_4	18	2643.31	[N][V]
	19	2631.77	[K]
C1_2_5	20	2702.05	[N][V][K]
C1_2_6	20	2701.04	[N][V][K]
C1_2_7	20	2701.04	[N][V][K]
C1_2_8	19	2775.48	[N][V]
	20	2690.27	[K]
C1_2_9	18	2687.83	[N][V]
	19	2645.47	[K]
C1_2_10	18	2643.55	[N][V]
	19	2640.45	[K]
C2_2_1	6	1931.44	[N][V][K]
	7	1931.30	[K]
C2_2_2	6	1863.16	[N][V][K]
C2_2_3	6	1775.08	[N][V][K]
C2_2_4	6	1703.43	[N][V]
C2_2_5	6	1878.85	[V]
C2_2_6	6	1857.35	[N][V][K]
C2_2_7	6	1849.46	[N][V][K]
C2_2_8	6	1820.53	[N][V]
C2_2_9	6	1830.05	[N][V]
C2_2_10	6	1806.58	[N][V]
	7	1803.04	[K]

[N]: Nagata *et al*. (2010): Table 8
[V]: Vidal *et al*. (2013): Table C2
[K]: KBEA

Table 10 Average computation time, number of runs, and computing environments (Gehring and Homberger 200-customer instances)

Approach	Avg. computation time (minutes)	#runs	CPU (language)
Nagata *et al*. (2010)	4.1	5	Opteron 2.4 GHz (C++)
Vidal *et al*. (2013)	8.7	5	Xeon 2.93 GHz (C++)
KBEA	4.0	10	i7-3770 3.4 GHz (C++)