Balancing exploration and exploitation in the memetic algorithm via a switching mechanism for the large-scale VRPTW

Abstract

This paper presents an effective memetic algorithm for the large-scale vehicle routing problem with time windows (VRPTW). Memetic algorithms consist of an evolutionary algorithm for the global exploration and a local search algorithm for the exploitation. In this paper, a switching mechanism is introduced to balance quantitatively between exploration and exploitation in the search process, to improve the convergent performance. Specifically, a similarity measure and a sigmoid function is defined to guide the crossover of individuals, instead of randomly crossing them. Experimental results on Gehring and Homberger's benchmark show that this algorithm outperforms previous approaches and improves 33 best-known solutions out of 180 large-scale instances. Although this paper focuses on the VRPTW, the proposed switching mechanism can be applied to accelerate more general genetic algorithms.

Introduction

The vehicle routing problem with time windows (VRPTW) has been one of the most important and widely investigated NP-hard optimization problems in transportation, supply chain management, and logistics. It is an important problem that is faced by every delivery company every day when they have to divide up their packages to a set of trucks. The VRPTW is an extension of the vehicle routing problem (Dantzig and Ramser 1959), which involves finding a set of routes, starting and ending at a depot, that together cover a set of customers. In the VRPTW, each customer should be visited exactly once by a single vehicle within a given time interval, and no vehicle can service more customers than its capacity permits.

The VRPTW is defined on a complete directed graph $G = (V, E)$, where $V = \{0, 1, ..., N\}$ is the set of vertices and $E = \{(i, j)|i \neq j \in V\}$ is the set of edges. Vertex 0 represents the depot and the other vertices represent the customers. Associated with each vertex i is a non-negative demand q_i, and a time window $[e_i, l_i]$, where e_i and l_i represent the earliest and latest time to visit vertex i. Associated with each edge (i, j) is a travel cost d_{ij} and a non-negative travel time t_{ij}, where t_{ij} includes the service time at vertex i.

A feasible solution to the VRPTW is a set of m routes in graph G such that the following conditions hold: (1) each route starts and ends at the depot, (2) each customer i belongs to exactly one route, (3) the total demand of the visited customers does not exceed the vehicle capacity Q, and (4) the service at each customer i begins between e_i and l_i, and if the vehicle arrives at i before e_i, the service is delayed to time e_i. The cost of one route is equal to the sum of the distance of the edges traveled. The problem often has a hierarchical objective: 1) Minimize number of vehicles 2) Minimize total distance.

Literature Review

Some exact algorithms have been proposed for solving the VRPTW, e.g. (Chabrier 2006; Irnich and Villeneuve 2006; Kallehauge *et al.* 2006; Bettinelli *et al.* 2011). For comprehensive review up to 2016, the reader is referred to (Kallehauge 2008; Baldacci *et al.* 2012; Braekers *et al.* 2016). These exact algorithms work well for small-scale problems. By now, most of the instances in Solomons benchmarks (100 customers) (Solomon 1987) have been solved to optimality.

However, due to the NP-hardness of the VRPTW, the computation time of exact methods for large-scale problems is not acceptable. Therefore, recent studies mainly focus on various heuristic algorithms, which can find acceptable, but not necessarily optimal solutions quickly. The current state-of-the-art heuristics for the VRPTW consist of simulated annealing (Chiang and Russell 1996; Wang *et al.* 2015), tabu search (Cordeau *et al.* 2001), ant colony optimization (Yu *et al.* 2009), particle swarm optimization (Gong *et al.* 2012), evolutionary algorithms (Bräysy *et al.* 2004; Mester and Bräysy 2005), and genetic and memetic algorithms (Nagata *et al.* 2010; Blocho and Czech 2013; Nalepa and Blocho 2016).

Among all these heuristics, memetic algorithms (MA) prove to be extremely effective, and our method belongs to this category. MA is a population based heuristic approach that combines evolutionary algorithm for the global search (exploration), with local search algorithm for the intensive search (exploitation) (Nagata *et al.* 2010). Currently used memetic algorithms usually apply an edge assembly crossover (EAX) operator to generate new solutions (children) from parents, followed by a repair operator to eliminate the infeasibility and a local search algorithm for the further improvement. (Nagata 2006; 2007; Nalepa and Blocho 2016) all demonstrate the potential of EAX in routing problems. However, in each iteration of the EAX operation, individual solutions are *randomly ordered* in order to conduct global exploration.

Recently, several studies find that route diversity and similarity are very important in evolutionary algorithms. The parent selection techniques used in the simple genetic algorithm (GA) only take into account the individual fitness, regardless of the parenthood or likeness. In GA, this process is random mating, while (Fernandes and Rosa 2001) choose the parents according to their Hamming distance. They find that their algorithm maintain a higher genetic diversity thus avoiding getting trapped in local optima. (Garcia-Najera and Bullinaria 2009) propose a similarity measure based on Jaccard's similarity coefficient (Jaccard 1908) and incorporate it into an evolutionary algorithm. Parents are recombined under both fitness and similarity. They demonstrate the importance that the similarity measure has, in the sense that the exploration of the search space is wider. (Shunmugapriya *et al.* 2016) also use the Jaccard's similarity coefficient for the measurement embedding in an artificial bee colony approach. They show that the similarity measure could avoid early convergence and get an appropriate balance between the exploration and exploitation.

Our Focus and Contributions

The classical memetic algorithm includes a random paired EAX operation. We find that the convergence of the algorithm could be slow, especially for large-scale problems. Although the local search operator helps improve the solution, the improvement to the acceleration of convergence is small (Mester and Bräysy 2005). Referring to the distance measurement, we focus on guiding the "ordering procedure" quantitatively, instead of random ordering.

The main idea is to control the order of individual solutions been paired, so that the EAX operation can realize both global exploration and local exploitation. Specifically, a *similarity measure* is defined to quantify the likeness between two individual solutions. Low similarity means that two solutions are far away from each other in the solution space, and vice versa. Thus, if solutions are paired with *low* similarity, the EAX operation is more like conducting global exploration. On the contrary, if solutions are paired with *high* similarity, the EAX operation is more like conducting local exploitation. A steady step counter C is defined to record the non-improvement iterations. As the increase of the counter, the EAX operation will switch from exploitation to exploration gradually. Once the value of the objective function gets improved, the counter will return to 0, meaning that a local exploitation is performed on the improved solution.

Our method is tested on the well-known benchmark problems of (Gehring and Homberger 1999). Experimental results show that by introducing the switching mechanism,

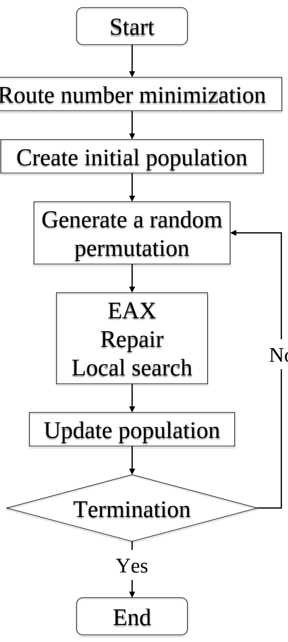
Start
Route number minimization
Create initial population
Generate a random permutation
EAX
Repair
Local search
Update population
Termination
No
Yes
End

Figure 1: The framework of memetic algorithms

this method can significantly speed up the solving procedure. Moreover, this switching mechanism can also be applied to other memetic and genetic algorithms. To the best of our knowledge, this method is the first to quantitatively control exploration and exploitation regarding crossover operation in memetic algorithms.

The rest of the paper is organized as follows. Next section presents the memetic algorithm with switching mechanism, termed as MASM. Results of the experimental study performed on the Gehring and Homberger benchmark are reported in the latter section. Finally, the last section concludes the paper and discusses some potential directions for future research.

Our Method

In this section, we will first introduce the currently used MA for the VRPTW in the literature. Then, we will explain the proposed MASM line by line. Finally the switching mechanism used to order solutions for EAX operation is provided, including a similarity measure, a switching function, and a new process of EAX.

An overview of the MASM

Figure 1 illustrates the solution procedure of standard MA, and proposed MASM also follows this procedure. Algorithm 1 shows the flow schema. The MASM is based on a two-stage approach where the number of routes and the total travel distance are independently minimized. At the first stage, the minimum number of routes/vehicles m is determined by the Guided Ejection Search (GES) algorithm (Nagata and Bräysy 2009) (line 1). GES is a route minimization heuristic based on the ejection pool, powerful insertion and guided local search strategies. An initial population of N_{pop} feasible

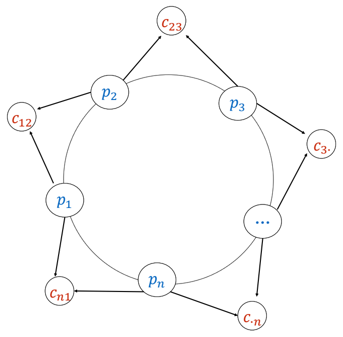

Figure 2: Ordering of individuals

solutions, each consisting of m routes is created (line 2-4). σ_i represents an individual solution in the population σ.

The second stage minimizes the total distance traveled. A steady step counter C is initiated in Line 6. The procedures on line 8-34 correspond to a single generation. In each generation, the individual solutions are ordered according to $p_k(\sigma_i)$, which determines the probability for each individual solution σ_i to be selected as the k_{th} one in the tour. The method used to calculate $p_k(\sigma_i)$ will be presented later. The N_{pop} individuals are arranged in a tour for pairing and crossover shown in Figure 2. An EAX operation (Line 15) is performed on each pair of adjacent parents (p_A the left-hand side, p_B the right-hand side) to induce a child (c_{AB}). Note that the number of individuals keeps unchanged in the next generation.

The repair operator (Line 16) is then used to recover the offspring solutions which violate the constraints of capacity or time window. Some local search methods (Line 17) are also employed to further improve the already found solutions by slightly changing their structures. The best feasible offspring selected from N_{ch} ones will replace the position of p_A in the next generation (Line 18-20, 22). Note that if the selected pair of parents are the same (the similarity is 1), the offspring solution will be generated by perturbing the left-hand side parent p_A (Line 24).

At the end of each generation, if the value of the objective function $F(\sigma_{gbest})$ decreases, C will be set to 0. Otherwise, C will be incremented by 1 (Line 30-34).

In the next three subsections, we will present the method used to determine $p_k(\sigma_i)$, which relies on another two parameters: a similarity measure and a switching indicator.

A Similarity Measure

To quantify the similarity between two individual solutions, we refer to the Jaccard Index (Jaccard 1908), which has been one of the most widely used indices for comparing the similarity of sample sets. The Jaccard Index between sample sets is defined as the size of the intersection divided by the size of the union of the sample sets. In our problem, let σ_A and σ_B be two solutions in one generation, and E_A and E_B be the sets of directed edges consisting of σ_A and σ_B, respectively. The similarity between σ_A and σ_B is defined

Algorithm 1 The memetic algorithm with switching mechanism (MASM)

```
1: $m \leftarrow$ NumRoute(GES(0));
2: **for** $i \leftarrow 1$ to $N_{pop}$ **do**
3:     $\sigma_i \leftarrow GES(m)$; $\sigma_i \leftarrow LocalSearch(\sigma_i)$;
4: **end for**
5: $\sigma_{gbest} \leftarrow$ the best individual in the population $\sigma$;
6: $C = 0$;
7: **repeat**
8:     $\sigma_{temp} \leftarrow \sigma_{gbest}$;
9:     Generate a permutation $r(i)$ according to $p_k(\sigma_i)$;
10:    **for** $i \leftarrow 1$ to $N_{pop}$ **do**
11:        $p_A \leftarrow \sigma_{r(i)}$; $p_B \leftarrow \sigma_{r(i+1)\%N_{pop}}$;
12:        **if** $p_A \neq p_B$ **then**
13:            $\sigma_{best} \leftarrow p_A$;
14:            **for** $j \leftarrow 1$ to $N_{ch}$ **do**
15:                $\sigma_{ch} \leftarrow EAX(p_A, p_B)$;
16:                $\sigma_{ch} \leftarrow$ Repair($\sigma_{ch}$);
17:                $\sigma_{ch} \leftarrow$ Localsearch($\sigma_{ch}$);
18:                **if** $F(\sigma_{ch}) < F(\sigma_{best})$ **then**
19:                    $\sigma_{best} \leftarrow \sigma_{ch}$;
20:                **end if**
21:            **end for**
22:            $\sigma_{r(i)} \leftarrow \sigma_{best}$;
23:        **else**
24:            $\sigma_{r(i)} \leftarrow$ Perturb($p_A$);
25:        **end if**
26:        **if** $F(\sigma_{r(i)}) < F(\sigma_{gbest})$ **then**
27:            $\sigma_{gbest} \leftarrow \sigma_{r(i)}$;
28:        **end if**
29:    **end for**
30:    **if** $F(\sigma_{gbest}) < F(\sigma_{temp})$ **then**
31:        $C = 0$;
32:    **else**
33:        $C = C + 1$;
34:    **end if**
35: **until** termination condition is satisfied
36: **return** $\sigma_{gbest}$
```

as:

$$s(\sigma_A, \sigma_B) = \frac{|E_A \cap E_B|}{|E_A \cup E_B|}, \tag{1}$$

where $|E_A \cap E_B|$ is the number of edges in the intersection of E_A and E_B, and $|E_A \cup E_B|$ is the number of edges in the union of E_A and E_B. Obviously, $s(\sigma_A, \sigma_B) \in (0, 1)$.

Considering that the variance of similarity values between solution pairs may change along the process of evolution, we normalize $s(\sigma_A, \sigma_B)$ by

$$S(\sigma_A, \sigma_B) = \frac{s(\sigma_A, \sigma_B) - \min\limits_{\sigma_A \neq \sigma_B} s(\sigma_A, \sigma_B)}{\max\limits_{\sigma_A \neq \sigma_B} s(\sigma_A, \sigma_B) - \min\limits_{\sigma_A \neq \sigma_B} s(\sigma_A, \sigma_B)}. \tag{2}$$

Note that the calculation of similarity measures and the normalization are repeated in each generation.

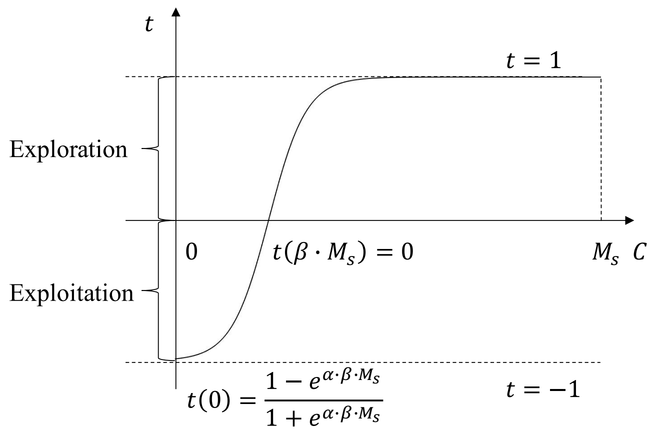

Figure 3: The indicator function for switching between exploration and exploitation

The Switching Mechanism Between Exploration and Exploitation

In the current memetic algorithms, exploration is conducted by EAX, and exploitation is conducted by different local search methods. The main feature of our method is that we embed both exploration and exploitation in the process of EAX by controlling the parent solutions been paired. The basic idea is that if the offspring solution is generated from two parents with low similarity, the EAX operation is regarded as conducting exploration; while if the offspring solution is generated from two parents with high similarity, the EAX operation is regarded as conducting exploitation.

Next, we introduce the switching mechanism between exploration and exploitation. The process starts with exploitation and repeats until the value of the objective function becomes steady. A variable C is used to count the steady steps. Then, we define an indicator t to control the switching from exploitation to exploration according to C, which is defined by the following sigmoid function

$$t = t(C) = \frac{1 - e^{\alpha}(\beta M_s - C)}{1 + e^{\alpha}(\beta M_s - C)}, \tag{3}$$

where M_s is the maximum steady steps used to control the termination of the algorithm, α and β are two parameters used to control the shape and location of the sigmoid function. As shown in Figure 3, when $-1 < t < 0$, the EAX operator conducts exploitation, and solutions with high similarity are more likely to be paired as parents to generate offspring solutions; when $0 \leq t < 1$, the EAX operator conducts exploration, and solutions with low similarity are more likely to be paired as parents to generate offspring solutions. The parameter β is used to control the frequency of switching from exploitation to exploration, and α together with β is used to control the shape of the function.

Once the value of the objective function gets decreased, the variable C returns to 0. And the process repeats until termination. Next we will show how to determine the probability $p_k(\sigma_i)$, which is used to order the individual solutions in one generation according to their similarity value and the switching indicator.

The New Process of Edge Assembly Crossover (EAX)

At the beginning of each generation, all the individual solutions are arranged in a tour, and for each pair of adjacent solutions, the EAX operator will generate an offspring solution. In the previous methods, the individual solutions are ordered randomly, while in this method, the probability for each individual to be selected as the next one is co-determined by (1) the similarity between it and the last individual and (2) the switching indicator. Specifically, the first solution $\sigma_{r(1)}$ is randomly selected, then the probability for each of the rest individuals to be selected as the second one is determined by

$$p_{r(2)}(\sigma_i) = \frac{0.5 - t[S(\sigma_{r(1)}, \sigma_i) - 0.5]}{\sum\limits_{\sigma_i \in \Delta_2} \{0.5 - t[S(\sigma_{r(1)}, \sigma_i) - 0.5]\}}, \tag{4}$$

where Δ_2 is the set of individual solutions that have not been selected. Generally, when $k-1$ individuals have been put into the tour, the probability for each of the remaining individuals to be selected as the k_{th} one is determined by

$$p_{r(k)}(\sigma_i) = \frac{0.5 - t[S(\sigma_{r(k-1)}, \sigma_i) - 0.5]}{\sum\limits_{\sigma_i \in \Delta_k} \{0.5 - t[S(\sigma_{r(k-1)}, \sigma_i) - 0.5]\}}. \tag{5}$$

Specially,

$$p_{r(k)}(\sigma_i) = \begin{cases} \frac{S(\sigma_{r(k-1)}, \sigma_i)}{\sum\limits_{\sigma_i \in \Delta_k} S(\sigma_{r(k-1)}, \sigma_i)}, & t = -1 \\ \frac{[1 - S(\sigma_{r(k-1)}, \sigma_i)]}{\sum\limits_{\sigma_i \in \Delta_k} [1 - S(\sigma_{r(k-1)}, \sigma_i)]}, & t = 1 \end{cases},$$

so $t = -1$ indicates that the probability for each individual to be selected is positively correlated with its similarity with the current individual in the end of the tour. In this way, individual solutions with high similarity will be paired to generate offspring for exploitation. On the contrary, $t = 1$ indicates that the probability for each individual to be selected is negatively correlated with its similarity with the current individual in the end of the tour. In this way, individual solutions with low similarity will be paired to generate offspring for exploration. With the increase of C, t changes from -1 to 1, and the EAX operator gradually switches from exploitation to exploration.

Experimental Results

The memetic algorithm is implemented in C++ under CUDA V8.0 environment and run on Nvidia P100 GPU (3584 CUDA cores, 16GB memory). It is tested on the well-known benchmarks of (Gehring and Homberger 1999). Two groups of experiments are conducted. In the first group, the MA with random EAX operations in the literature and the proposed MASM in this paper are compared on four representative benchmarks. In the second group, proposed MASM is applied to the 180 largest benchmarks. In what follows, we will introduce the benchmark problems briefly, give the experimental settings, and present the results.

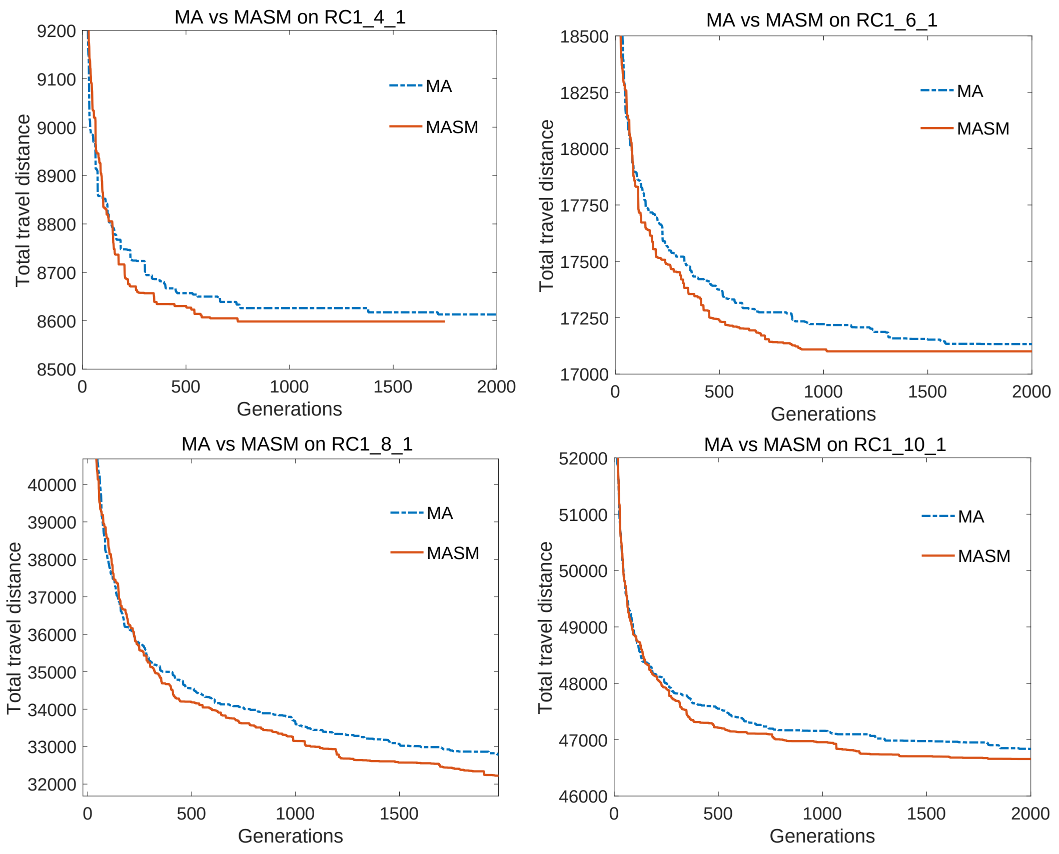

Figure 4: The performance of MA with random EAX operation and the MASM on four instances

Table 1: Computation time (in seconds) for one generation among different population size in RC1_6_1

N_{pop}	100	200	500	1000
MA	3.5	4.8	8.2	13.5
MASM	3.5	4.9	8.3	13.7

Benchmark Problems

The instances in the benchmark of Gehring and Homberger consist of five sets of 200, 400, 600, 800, 1000 customers. Each set is divided into six groups: C1, C2, R1, R2, RC1, and RC2, each containing 10 instances. The customers in C1 and C2 classes are located in clusters, while the customers in R1 and R2 are at random positions. The RC1 and RC2 classes contain a mix of clustered and random customers. The C2, R2 and RC2 classes have longer service time horizons and larger vehicle capacities than C1, R1 and RC1 classes. The best-known solutions are collected from the Sintef website [1].

[1]Sintef website: https://www.sintef.no/projectweb/top/vrptw/.

Experimental Setting

In the first group of experiments, the selected instances consist of the first one from the RC categories, i.e., RC1_4_1, RC1_6_1, RC1_8_1 and RC1_10_1. This category covers both the random and cluster customers. The traditional MA and the proposed MASM are then applied. The population size N_{pop} is set to 100. The maximal number of generations is 2000. In the second group of experiments, all the instances from the six categories are selected, and only the MASM is applied. The population size is set to 1000 and the maximal number of generations is 5000.

Other parameters remain the same in all experiments. In detail, the maximum number of steady generation M_s is 1000, the number of children in each reproduction step N_{ch} is 20. Parameters α and β are determined by grid search. The proposed sigmoid function is a transformation of

$$f(x) = \frac{1-e^x}{1+e^x} \sim \begin{cases} -1, & x \leq -5 \\ 1, & x \geq 5 \end{cases} \tag{6}$$

Since $M_s = 1000$, we try different β, and βM_s is the number of steady steps before the algorithm switches from exploitation to exploration. α is determined by making $\alpha\beta M_s$ close to 5. We find it work well with $\alpha = \frac{1}{20}$ and $\beta = \frac{1}{10}$.

Computational results

The results of the first group of experiments are illustrated in Figure 4. As we can see, the MASM outperforms the previous MA with random EAX operation in terms of both convergence speed and the solution quality. For instance RC1_4_1, the MASM terminates early as the condition of maximum steady generations is satisfied. If the population size increases, the performances of both MA and MASM will be improved, and in most cases, the MASM is better than the MA.

Note that the computation time for calculating similarity is very short, compared with the EAX operation, and thus can be ignored. To calculate the similarity between two individual solutions, we only need to compare the neighborhood for each customer, and the computational complexity is $O(N)$. Take RC1_6_1 as an example, Table 1 shows that the computation time for one generation slightly changes between MA and MASM.

The results of the second group of experiments are shown in Table 2. We compare our results with those published on the Sintef website and mark the new world-best solutions in bold. The benchmark of Gehring and Homberger has been a standard set for evaluating the performance of emerging techniques for solving the VRPTW. A lot of teams have focused on this benchmark problem and kept updating the world-best solutions (Blocho and Czech 2013; Vidal *et al.* 2013; Nalepa and Blocho 2015). In total, 33 best-known solutions out of 180 instances are improved.

We set the max number of generations as 5000, and most results are obtained within 2000 generations. What's more, most new best solutions are obtained within 10 hours, which is close to those reported in the literature (e.g., 11 hours in (Nalepa and Blocho 2015) and 6 hours in (Vidal *et al.* 2013)) who also focus on solving the GH benchmark. Finally, we need to make a note about the computing platform. According to the recent literature, when solving the GH benchmarks, servers are usually used. For example, the performance of the supercomputer used in (Nalepa and Blocho 2015) is 5 times better than ours. The records on the Sintef website are updated by different teams around the world monthly. Our algorithm is running and continuing to find better solutions.

Conclusion and Future Work

The paper describes a memetic algorithm to solve large instances of the vehicle routing problem with time windows (VRPTW). The proposed algorithm incorporates a mechanism to switch between the exploration and exploitation phases of the search. The central aspects of the proposed mechanism are a similarity measure and a sigmoid function to control which solutions are paired for crossover. Specifically, the contribution is a modification to the application of the edge assembly crossover (EAX) so that instead of on solutions ordered at random, a similarity measure is used to pair individuals for crossover. The rationale is that such guidance will accelerate the convergence of the memetic algorithm on large VRPTW instances.

The memetic algorithm is tested on the instances in the benchmark of Gehring and Homberg. First, the algorithm is compared with the traditional memetic algorithm on some instances that contain a mix of clustered and random customers. In the second step the algorithm is tested on all instances with 600, 800 and 1000 customers and new results are obtained on some instances. Experimental results show that this algorithm outperforms previous approaches and improves 33 best-known solutions out of 180 large-scale instances.

Although this paper focuses on the VRPTW, the proposed switching mechanism can be applied to more general genetic algorithms to accelerate the searching of good solutions. Crossover is one of the most important operators in genetic algorithms. We present in this paper that by controlling the parents been paired for crossover rather than conducting this operation randomly, the algorithm can find better solutions much more quickly. There are several potential directions for future research. One is to automatically determine the parameters used in the switching mechanism for different problems. Another one is to further analyze the mechanism of this algorithm, and investigate the conditions under which the algorithm performs best.

Table 2: The best results for GH benchmark problems (new world-best solutions are highlighted in bold)

No.	C1	C2	R1	R2	RC1	RC2
600 customers						
1	60/14095.64	18/7774.16	**59/21394.95**	**11/18205.58**	55/17006.26	15/12884.87
2	56/14163.31	**17/8258.20**	54/18611.77	**11/14754.13**	**55/15914.70**	12/11555.51
3	56/13778.75	17/7517.14	54/16917.31	11/11190.52	55/15239.45	11/9440.39
4	56/13558.54	17/6909.58	54/15840.71	**11/8010.13**	55/14838.22	**11/7057.94**
5	60/14085.72	18/7575.2	54/19400.86	11/15083.58	55/16564.01	13/11688.25
6	59/15847.24	18/7471.17	54/17935.80	11/12501.36	55/16603.10	**11/11913.11**
7	58/14862.98	18/7512.07	54/16531.60	11/10071.64	55/16211.69	11/10722.67
8	57/14072.05	17/7588.00	54/15618.25	11/7572.76	55/16004.35	**11/9990.40**
9	56/13715.94	**17/7911.61**	54/18554.55	11/13377.56	55/15940.27	**11/9574.99**
10	56/13658.27	17/7256.55	54/17621.89	11/12202.28	55/15701.07	11/9059.39
800 customers						
1	80/25184.38	24/11662.08	80/36839.91	15/28112.36	72/30651.96	19/20385.26
2	**72/26554.44**	23/12321.66	**72/32322.85**	**15/22795.79**	72/28571.38	**16/18151.95**
3	72/24258.48	23/11411.35	72/29359.77	15/17754.48	72/27588.41	15/14443.39
4	72/23841.37	22/11007.83	72/27902.68	15/13424.86	72/26905.25	**15/10999.03**
5	80/25166.28	24/11425.23	**72/33529.73**	15/24255.00	72/29609.58	**15/19074.02**
6	80/25160.85	24/11347.64	72/30957.14	**15/20412.02**	72/30426.46	**15/18143.04**
7	78/25912.48	24/11370.84	72/28961.87	15/16647.67	72/29069.79	15/16818.96
8	75/24862.75	23/11298.89	72/27671.21	15/12695.11	72/28990.32	**15/15759.14**
9	72/24388.59	23/11637.31	72/32319.05	15/22329.42	72/28629.65	15/15338.16
10	72/24076.85	23/10983.86	72/30984.44	**15/20358.61**	72/29235.00	15/14413.74
1000 customers						
1	100/42478.95	30/16879.24	100/53494.81	**19/42182.57**	90/45933.10	**20/30276.27**
2	**90/42222.96**	29/17126.39	91/49142.72	19/33421.78	90/43788.75	19/25409.02
3	90/40227.77	**28/16829.47**	91/44792.33	19/24921.46	90/42205.04	**18/19913.46**
4	90/39547.79	28/15658.27	91/42528.89	19/17853.77	90/41429.25	**18/15693.28**
5	100/42469.18	30/16561.29	91/50523.42	19/36261.78	90/45135.66	18/27127.22
6	100/42471.28	30/16335.07	**91/46992.54**	**19/29998.44**	90/44956.59	**18/26741.27**
7	99/42592.74	30/16426.56	91/44098.80	19/23307.43	90/44468.88	**18/25017.97**
8	94/41775.84	29/16169.09	91/42362.56	19/17449.43	90/44018.28	18/23600.88
9	90/40569.73	29/16388.86	91/49240.78	**19/32995.71**	90/43960.01	18/22945.56
10	90/40010.1	29/15826.91	91/48109.97	19/30214.91	90/43597.97	18/21853.62

References

Roberto Baldacci, Aristide Mingozzi, and Roberto Roberti. Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, 218(1):1–6, 2012.

Andrea Bettinelli, Alberto Ceselli, and Giovanni Righini. A branch-and-cut-and-price algorithm for the multi-depot heterogeneous vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies*, 19(5):723–740, 2011.

Miroslaw Blocho and Zbigniew J Czech. A parallel memetic algorithm for the vehicle routing problem with time windows. In *2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, pages 144–151. IEEE, 2013.

Kris Braekers, Katrien Ramaekers, and Inneke Van Nieuwenhuyse. The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, 99:300–313, 2016.

Olli Bräysy, Wout Dullaert, and Michel Gendreau. Evolutionary algorithms for the vehicle routing problem with time windows. *Journal of Heuristics*, 10(6):587–611, 2004.

Alain Chabrier. Vehicle routing problem with elementary shortest path based column generation. *Computers & Operations Research*, 33(10):2972–2990, 2006.

Wen-Chyuan Chiang and Robert A Russell. Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Annals of Operations Research*, 63(1):3–27, 1996.

Jean-François Cordeau, Gilbert Laporte, and Anne Mercier. A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational research society*, 52(8):928–936, 2001.

George B Dantzig and John H Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959.

Carlos Fernandes and Agostinho Rosa. A study on non-random mating and varying population size in genetic algorithms using a royal road function. In *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546)*, volume 1, pages 60–66. IEEE, 2001.

Abel Garcia-Najera and John A Bullinaria. Bi-objective optimization for the vehicle routing problem with time windows: Using route similarity to enhance performance. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 275–289. Springer, 2009.

Hermann Gehring and Jörg Homberger. A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In *Proceedings of EUROGEN99*, volume 2, pages 57–64. Citeseer, 1999.

Yue-Jiao Gong, Jun Zhang, Ou Liu, Rui-Zhang Huang, Henry Shu-Hung Chung, and Yu-Hui Shi. Optimizing the vehicle routing problem with time windows: a discrete particle swarm optimization approach. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(2):254–267, 2012.

Stefan Irnich and Daniel Villeneuve. The shortest-path problem with resource constraints and k-cycle elimination for k 3. *INFORMS Journal on Computing*, 18(3):391–406, 2006.

Paul Jaccard. Nouvelles recherches sur la distribution florale. *Bull. Soc. Vaud. Sci. Nat.*, 44:223–270, 1908.

Brian Kallehauge, Jesper Larsen, and Oli BG Madsen. Lagrangian duality applied to the vehicle routing problem with time windows. *Computers & Operations Research*, 33(5):1464–1487, 2006.

Brian Kallehauge. Formulations and exact algorithms for the vehicle routing problem with time windows. *Computers & Operations Research*, 35(7):2307–2330, 2008.

David Mester and Olli Bräysy. Active guided evolution strategies for large-scale vehicle routing problems with time windows. *Computers & Operations Research*, 32(6):1593–1614, 2005.

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

Yuichi Nagata, Olli Bräysy, and Wout Dullaert. A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Computers & operations research*, 37(4):724–737, 2010.

Yuichi Nagata. Fast eax algorithm considering population diversity for traveling salesman problems. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 171–182. Springer, 2006.

Yuichi Nagata. Edge assembly crossover for the capacitated vehicle routing problem. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 142–153. Springer, 2007.

Jakub Nalepa and Miroslaw Blocho. Co-operation in the parallel memetic algorithm. *International Journal of Parallel Programming*, 43(5):812–839, 2015.

Jakub Nalepa and Miroslaw Blocho. Adaptive memetic algorithm for minimizing distance in the vehicle routing problem with time windows. *Soft Computing*, 20(6):2309–2327, 2016.

P Shunmugapriya, S Kanmani, P Jude Fredieric, U Vignesh, J Reman Justin, and K Vivek. Effects of introducing similarity measures into artificial bee colony approach for optimization of vehicle routing problem. *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 10(3):651–658, 2016.

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

Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. 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, 2013.

Chao Wang, Dong Mu, Fu Zhao, and John W Sutherland. A parallel simulated annealing method for the vehicle routing problem with simultaneous pickup–delivery and time windows. *Computers & Industrial Engineering*, 83:111–122, 2015.

Bin Yu, Zhong-Zhen Yang, and Baozhen Yao. An improved ant colony optimization for vehicle routing problem. *European journal of operational research*, 196(1):171–176, 2009.