Soft Comput (2016) 20:2309–2327
DOI 10.1007/s00500-015-1642-4

METHODOLOGIES AND APPLICATION

Adaptive memetic algorithm for minimizing distance in the vehicle routing problem with time windows

Jakub Nalepa · Miroslaw Blocho

Published online: 11 March 2015

Abstract This paper presents an adaptive memetic algorithm to solve the vehicle routing problem with time windows (VRPTW). It is a well-known NP-hard discrete optimization problem with two objectives—to minimize the number of vehicles serving a set of geographically dispersed customers, and to minimize the total distance traveled in the routing plan. Although memetic algorithms have been proven to be extremely efficient in solving the VRPTW, their main drawback is an unclear tuning of their numerous parameters. Here, we introduce the adaptive memetic algorithm (AMA-VRPTW) for minimizing the total travel distance. In AMA-VRPTW, a population of solutions evolves with time. The parameters of the algorithm, including the selection scheme, population size and the number of child solutions generated for each pair of parents, are adjusted dynamically during the search. We propose a new adaptive selection scheme to balance the exploration and exploitation of the solution space. Extensive experimental study performed on the well-known Solomon's and Gehring and Homberger's benchmark sets confirms the efficacy and convergence capabilities of the proposed AMA-VRPTW. We show that it is very competitive compared with other state-of-the-art techniques. Finally, the influence of the proposed adaptive schemes on the AMA-VRPTW behavior and performance is investigated in a thorough sensitivity analysis. This analysis is complemented with the two-tailed Wilcoxon test for verifying the statistical significance of the results.

Keywords Memetic algorithm · Adaptation · Parameter control · Selection scheme · Vehicle routing problem with time windows

Communicated by V. Loia.

J. Nalepa (✉)
Institute of Informatics, Silesian University of Technology, Akademicka 16, 44-100 Gliwice, Poland
e-mail: jakub.nalepa@polsl.pl

M. Blocho
ABB ISDC, Zeganska 1, 04-713 Warsaw, Poland
e-mail: miroslaw.blocho@pl.abb.com

1 Introduction

Route scheduling is one of the most important real-life problems and became a core issue in transportation, supply chain management and logistics. Its numerous practical applications include the bus route planning, post, parcels, food and beverage delivery, cash delivery to banks and ATM terminals, industrial waste collection, maintenance operations, and many more. While constructing the routing schedule for a given distribution problem, it is necessary to consider a large number of practical issues, e.g., the available fleet size, truck capacities, travel costs between geographically dispersed customers, possible time intervals in which customers should be visited, and numerous other circumstances. These factors affect the feasibility of a routing plan.

Minimizing the number of trucks and their total distance traveled during the service contributes to reducing the fleet exploitation costs and fuel consumption, it lessens the price of delivered goods, and helps in reducing the environmental pollution and traffic congestion (Hosny and Mumford 2010). Numerous variants of vehicle routing problems (VRPs) reflect real-life scheduling scenarios (Dantzig and Ramser 1959; Creput and Koukam 2008). In the multiple traveling salesman problem (mTSP)—an extension of a standard traveling salesman problem—more than one salesman can serve customers (Bektas 2006). Each customer specifies a non-negative demand, which should be satisfied by the

salesman. Clearly, the number of vehicles (salesmen) should be as minimum as possible. In the capacitated vehicle routing problem (CVRP), vehicle capacities cannot be exceeded (Niu et al. 2014). In commercial transportation problems, customers usually expect their orders within a specified time slot. The vehicle routing problem with time windows (VRPTW) incorporates delivery time constraints to address this issue (Kallehauge 2008). It is a two-objective NP-hard discrete optimization problem (Garey and Johnson 1990). Its main objective is to minimize the number of homogeneous vehicles serving customers scattered around the map. Then, the total travel distance is to be minimized. There exist a plethora of other variants of the VRPs which consider additional scheduling constraints (Marinakis and Marinaki 2014; Masson et al. 2014).

State-of-the-art algorithms for tackling the VRPTW include exact and approximate methods. Since this problem is NP-hard (Garey and Johnson 1990), the former approaches can be applied only for relatively small problem instances. Therefore, various heuristic algorithms that do not guarantee obtaining the optimal solution but execute very fast have been introduced to solve the VRPTW in a short time, and became a main stream of development in this field. They encompass simulated annealing (Zhong and Pan 2007), tabu searches (Ho and Haugland 2004), ant colony systems (Gambardella et al. 1999; Gomez et al. 2014), swarm optimization algorithms (Hu et al. 2013), evolutionary approaches (Repoussis et al. 2009), genetic and memetic algorithms (GAs and MAs) (Ghoseiri and Ghannadpour 2010; Nagata et al. 2010; Nalepa and Czech 2013; Vidal et al. 2013; Nalepa and Blocho 2014), and more (Bräysy and Gendreau 2005).

Although evolutionary algorithms were shown to be extremely effective, these techniques require their numerous parameters to be given prior to the optimization, which is difficult in practice. These algorithms need to be run multiple times in a very time-consuming tuning process to find the most appropriate set of parameter values. Due to the difficulty of the VRPTW, it became a significant disadvantage of the mentioned GAs and MAs. This issue was initially addressed in our preliminary study which showed that adapting MA parameters using basic mechanisms helps improve its convergence capabilities (Nalepa 2014). In this paper, we propose a new adaptive memetic algorithm.

1.1 Contribution

As previously stated, the main drawback of the MAs applied for solving the VRPTW lies in an unclear tuning of their numerous parameters. Since the state-of-the-art methods use static parameter values, they must be set a priori, and do not change during the optimization process. Clearly, improperly determined parameters can easily jeopardize the convergence of the MA and deteriorate its performance. In this paper, we propose a new adaptive MA (termed AMA-VRPTW) which dynamically adapts itself according to the current state of the search without any additional knowledge given prior to the algorithm execution. This mitigates the necessity of performing a time-consuming tuning process.

The adaptation in AMA-VRPTW includes controlling the selection scheme, population size, and the number of child solutions generated during the recombination. Also, we introduce a new adaptive selection scheme which balances the exploration and exploitation capabilities of AMA-VRPTW based on the current search progress. Experimental results obtained for two standard and widely used benchmark sets empirically demonstrate that our adaptive MA efficiently controls its parameters on the fly which leads to high convergence capabilities of the proposed method, and show that AMA-VRPTW is very competitive compared with other techniques reported in the literature. Finally, we present a thorough sensitivity analysis on various method components, followed by the two-tailed Wilcoxon test for assessing statistical significance of the results to investigate how the proposed adaptation schemes contribute to the performance of AMA-VRPTW.

1.2 Paper outline

The remainder of this paper is organized as follows. The problem is formally defined in Sect. 2. Section 3 reviews the state-of-the-art algorithms for solving the VRPTW. Section 4 discusses in detail the proposed adaptive memetic algorithm. The results of an extensive experimental study performed on standard Solomon's and Gehring and Homberger's benchmark sets, along with the sensitivity analysis on various method components followed by the two-tailed Wilcoxon test, are reported and analyzed in Sect. 5. Section 6 concludes the paper and highlights directions of our future work.

2 Problem formulation

The VRPTW is formulated as a problem of serving M customers by a fleet of K vehicles. The vehicles have a constant capacity Q, which makes the fleet homogeneous. A single depot (v_0) is the start and the finish point of each route. The customers v_i, $i \in \{1, 2, \ldots, M\}$, define their own service times s_i, $i \in \{1, 2, \ldots, M\}$. Serving the depot does not take any time ($s_0 = 0$), whereas the customer service times are non-negative. A non-negative demand d_i, $i \in \{1, 2, \ldots, M\}$, is given for each customer. The travel costs between each pair of travel points (i.e., distances in the Euclidean metric) are given as $c_{(i,j)}$, where $i \neq j$, $i, j \in \{0, 1, \ldots, M\}$. These travel costs correspond to the travel times. Each customer and the depot specifies its earliest and latest time of starting the service (i.e., time window), e_i and l_i, respectively ($i \in \{0, 1, \ldots, M\}$).

More formally, the VRPTW is defined on a directed graph $G = (V, E)$ with a set V of $M + 1$ vertices representing the customers and the depot, along with a set of edges $E = \{\langle v_i, v_{(i+1)}\rangle | v_i, v_{(i+1)} \in V, v_i \neq v_{(i+1)}\}$, representing the connections between the travel points. Intuitively, the 0th vertex v_0 represents the depot. Each vehicle is assigned a set of customers from exactly one route for service. The ith route is an ordered list of m_i customers to serve: $r_i = \langle v_0, v_{(r_i(1))}, \ldots, v_{(m_i+1)}\rangle$, where $v_0 = v_{(m_i+1)}$ is the depot, and $v_{(r_i(j))}$ denotes the jth customer visited in r_i.

2.1 Objectives

The VRPTW is a hierarchical objective discrete optimization problem. The primary objective is to minimize the fleet size K (i.e., the number of vehicles serving M customers). It is easy to note that $K \geq K_{\min}$, where $K_{\min} = \lceil D/Q \rceil$. Here, $D = \sum_{i=1}^{M} d_i$ denotes the total customer demands. Secondly, the total distance T traveled by the vehicles is to be minimized:

$$T = \sum_{i=0}^{M}\sum_{j=0}^{M}\sum_{k=1}^{K} x_{(i,j,k)} c_{(i,j)}. \tag{1}$$

The problem is characterized by three decision variables:

$$x_{(i,j,k)}\ (i, j \in \{0, 1, \ldots, M\}, i \neq j, \forall k \in K), \tag{2}$$

$$a_i\ (i \in \{0, 1, \ldots, M\}), \text{ and} \tag{3}$$

$$w_i\ (i \in \{1, 2, \ldots, M\}). \tag{4}$$

If the kth vehicle travels from v_i to v_j (where $i \neq j$), then $x_{(i,j,k)} = 1$ (0 otherwise). Two other decision variables indicate the arrival and the waiting time at v_i (a_i and w_i, respectively). There is no waiting time at the depot (and $a_0 = e_0$).

Let σ_A and σ_B be two solutions to the VRPTW. Then, σ_A is of a higher quality than σ_B, if $(K(\sigma_A) < K(\sigma_B))$ or $(K(\sigma_A) = K(\sigma_B)$ and $T(\sigma_A) < T(\sigma_B))$.

2.2 Constraints

The VRPTW constraints may be formally expressed by the following equations:

$$\sum_{k=1}^{K}\sum_{j=0, j\neq i}^{M} x_{(i,j,k)} = \sum_{k=1}^{K}\sum_{i=0, j\neq i}^{M} x_{(i,j,k)} = 1 \quad (\forall i, j \in V), \tag{5}$$

$$\sum_{j=1}^{M} x_{(i,j,k)} = \sum_{j=1}^{M} x_{(j,i,k)} = 1 \quad (i = 0, \forall k \in K), \tag{6}$$

$$\sum_{j=1}^{M}\sum_{k=1}^{K} x_{(i,j,k)} = K \quad (i = 0), \tag{7}$$

$$\sum_{i=1}^{M} d_i \sum_{j=0, j\neq i}^{M} x_{(i,j,k)} \leq Q \quad (\forall k \in K), \tag{8}$$

$$e_i \leq a_i + w_i \leq l_i \quad (\forall i \in V), \tag{9}$$

$$\max\{a_{(i-1)} + s_{(i-1)} + c_{((i-1),i)}, e_i\} \leq l_i \quad \big(i \in \{1, 2, \ldots, m_k + 1\}, \forall k \in K\big). \tag{10}$$

It has to be assured that every customer is visited exactly once (Eq. 5), all the routes start and finish at the depot (Eq. 6), and the fleet size is equal to K (Eq. 7). The capacity (Eq. 8) and the time window constraints (Eqs. 9, 10) must hold for each route. Thus, the total amount of goods delivered to customers by a vehicle cannot exceed Q, and the service of each customer must be started before its time window elapses.

Arriving at a customer $v_{(i+1)}$ (at a certain time point) is visualized in Figs. 1, 2, 3 (axes show the elapsing time τ). If a vehicle visits $v_{(i+1)}$ within its time window ($e_{(i+1)} \leq a_{(i+1)} \leq l_{(i+1)}$), then the service is immediate (Fig. 1).

Alternatively, the vehicle may arrive at $v_{(i+1)}$ before the beginning of the time window ($a_{(i+1)} < e_{(i+1)}$). However, the service cannot be initiated before $e_{(i+1)}$ (we consider hard time windows), and the vehicle waits until the time window is open (see Fig. 2—the dotted line indicates the waiting time $w_{(i+1)}$ at $v_{(i+1)}$).

If a vehicle visits $v_{(i+1)}$ after closing its time window ($a_{(i+1)} > l_{(i+1)}$), it violates the time window constraint, and

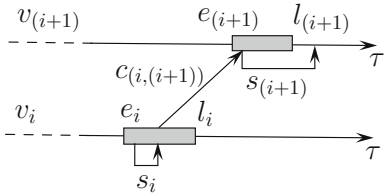

Fig. 1 Visiting a customer $v_{(i+1)}$ within its time window

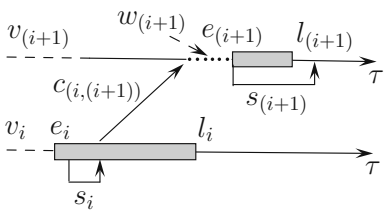

Fig. 2 Visiting a customer $v_{(i+1)}$ before its time window starts

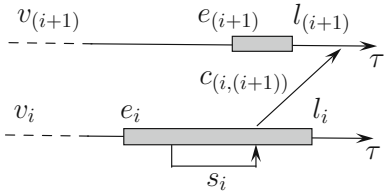

Fig. 3 Visiting a customer $v_{(i+1)}$ after closing its time window

the service is not feasible (Fig. 3). The entire route r containing $v_{(i+1)}$ becomes infeasible, so as σ.

3 Related literature

3.1 Vehicle routing problem with time windows

Due to the NP-hardness of the VRPTW (Garey and Johnson 1990), and its wide practical applicability, it was extensively studied over the years. Since the computation time of exact approaches is not acceptable for large-scale problem instances, a plethora of heuristic algorithms (which find high-quality, but not necessarily optimal, solutions quickly) have been proposed. In this section, we review the state-of-the-art techniques for solving the VRPTW.

Most exact approaches consider minimizing the total travel distance as the single optimization objective. In many aspects, they inherit from works devoted to solving the TSP (Kallehauge 2008). They encompass branch-and-cut (Bard et al. 2002), and branch-cut-and-price procedures (Abdallah and Jang 2014), algorithms utilizing many different problem formulations (Kolen et al. 1987; Feillet et al. 2004; Larsen 2004; Chabrier 2006; Righini and Salani 2006; Baldacci et al. 2011), and adopting other problems for the VRPTW (Irnich and Villeneuve 2006). Exact methods were summarized in thorough surveys and reviews (Cordeau et al. 2002; Kallehauge 2008; El-Sherbeny 2010; Baldacci et al. 2012). Although exact algorithms are continuously being developed, they are still not applicable to large real-life VRPTW instances due to their execution time. Also, they are strongly dependent on test characteristics (Vidal et al. 2013).

In heuristic techniques, two VRPTW objectives are usually considered independently. Therefore, two-stage algorithms (both sequential and parallel), in which the number of vehicles is minimized at first, and then the travel distance is optimized, are a vital research topic. A two-stage approach enables designing effective algorithms for both optimization stages independently. In construction heuristics, unserved customers are iteratively inserted into a partial solution (Potvin and Rousseau 1993; Petch and Salhi 2003; Tavares et al. 2009; Pang 2011). Improvement heuristics modify an initial solution to explore the solution space (Bräysy and Gendreau 2005; Nagata and Bräysy 2009). Meta-heuristic algorithms, which often couple exploring the search space with its intensive exploitation, allow for existing infeasible solutions and deteriorating their quality temporarily. Such approaches include simulated annealing (Chiang and Russell 1996), tabu searches (Ho and Haugland 2004), swarm optimization (Hu et al. 2013), ant colony systems (Gambardella et al. 1999; Gomez et al. 2014), hybrid techniques (Liu et al. 2014), and many more (Bräysy and Gendreau 2005; Coltorti and Rizzoli 2007; Banos et al. 2013).

Evolutionary algorithms (EAs) have been very extensively explored for tackling the VRPTW (Thangiah et al. 1991; Zhu 2000; Ombuki et al. 2006; Repoussis et al. 2009). Genetic algorithms (GAs) consist in evolving a population of solutions. It is then continuously improved in the biologically inspired manner, in which chromosomes are successively selected, crossed-over, and mutated. Similarly, memetic algorithms (MAs) are built upon the population-based approach, and combine EAs for exploring the solution space with local refinement procedures for exploiting solutions already found. MAs (both sequential and parallel) were shown to be very effective in solving the VRPTW (Nagata et al. 2010; Blocho and Czech 2012a, b, 2013; Nalepa and Czech 2013; Vidal et al. 2013, Nalepa and Blocho 2014). Memetic techniques have been applied to a bunch of other optimization and pattern recognition problems in a variety of science and engineering domains (Li et al. 2013, 2014; Guan et al. 2014; Jin et al. 2014; Marinaki and Marinakis 2014; Nalepa and Kawulok 2014), and they outperformed other evolutionary algorithms in terms of the convergence capabilities.

3.2 Adaptive evolutionary algorithms

The most important shortcoming of the mentioned GAs and MAs is an unclear selection of their numerous parameters to ensure the proper convergence speed, exploration and exploitation capabilities. Since these methods use static parameters (i.e., they do not change during the execution), this decision significantly influences the performance, and should be undertaken very carefully. Thus, the current state-of-the-art GAs and MAs for VRPTW must be executed multiple times to determine an acceptable set of appropriate parameters, which is usually a computationally intensive and time-consuming task. In our recent work, we showed how the choice of the co-operation scheme (defining the co-operation of parallel processes, called islands) in the parallel MA influences the search (Nalepa and Blocho 2014). Here, we address the issue of an automatic control of various algorithm parameters in the proposed adaptive MA. Thus, the necessity of performing the tuning process is mitigated.

A significant research effort has been put into proposing approaches for improving EAs by optimizing their parameters. Two main streams of development include parameter tuning and control. The former techniques determine "good" parameter values before the algorithm run. In a majority of such methods, a single parameter is tuned at a time, which may result in sub-optimal choices since the parameters are not independent. Clearly, this process is very time consuming and does not necessarily lead to optimally selected values.

Also, trying all parameter combinations is practically impossible. Each EA run is inherently dynamic and adaptive, and different parameters may be "optimal" at various steps of the optimization. This is not considered in tuning schemes—they find a single set of parameters used during the entire execution.

Control techniques aim at setting parameters of dynamic EAs on the fly (i.e., during the algorithm execution). Control EAs are classified based on many aspects: what is changed (selection scheme, mutation rate, etc.), how is it done (deterministic, feedback-based, and self-adaptation), what is the scope (level) of change (population level, individual level, etc.), and what is the evidence upon which the change is carried out (monitoring the progress of the search, population diversity, performance of operators, etc.) (Eiben et al. 1999). However, by tradition, the main criteria of classifying strategies for parameter control are the (straight-forward) what, and (less-obvious) how. Control strategies are divided into three categories based on the "how aspect": deterministic (the parameter is updated without any feedback from the running EA), adaptive (there is a feedback from the EA, and the parameters are adjusted accordingly), self-adaptive (parameters are encoded into individuals and evolve). Following this taxonomy, the proposed AMA-VRPTW is an adaptive MA.

4 Adaptive memetic algorithm

In this section, we present in detail the proposed adaptive MA (AMA-VRPTW). We introduce a new adaptive selection scheme, termed the adaptive exploration–exploitation scheme (AE^2), which balances the exploration and exploitation of the solution space based on the current optimization state. The dynamic adaptation of various algorithm parameters, including the population size, selection scheme and the number of children generated for each pair of parents during the recombination process, is discussed. It is worth pointing out that the crossover, mutation, and education procedures applied in AMA-VRPTW have been already utilized in our earlier works and other MAs (Nagata et al. 2010; Blocho 2013; Nalepa et al. 2014; Nalepa and Blocho 2014), and proved to be very robust. Here, we briefly discuss them.

4.1 Chromosomes

In AMA-VRPTW (Algorithm 1), each individual p_i, $i \in \{1, 2, \ldots, N\}$, represents a solution σ_i with K routes (therefore, $K(p_i) = K(\sigma_i) = K$, and $T(p_i) = T(\sigma_i)$) in a population of size N, where $N = N_I$ at the beginning (line 1). The initial population is generated using a guided ejection

Algorithm 1 Adaptive memetic algorithm (AMA–VRPTW).

1: $done \leftarrow$ **false**; $N \leftarrow N_I$; $\Delta N \leftarrow N_I$, $s \leftarrow 0$; $T(p_p^B) \leftarrow \infty$; $\mathcal{S} \leftarrow AB$;
2: Generate N solutions with K routes each; ▷ GES
3: **while** (**not** $done$) **do**
4: Determine N pairs (p_a, p_b); ▷ Selection
5: **for all** (p_a, p_b) **do**
6: $T(p_c^B) \leftarrow \infty$; $N_c \leftarrow 1$; $reproduce \leftarrow$ **true**;
7: **while** ($reproduce$) **do**
8: $p_c \leftarrow$ GenerateChild(p_a, p_b); ▷ Fig. 4
9: $p_c^B \leftarrow$ UpdateBestChild(p_c^B, p_c);
10: **if** $(T(p_c^B) < T(p_a)$ **or** $T(p_c^B) < T(p_b)$ **or** $N_c > N/2)$ **then**
11: $reproduce \leftarrow$ **false**;
12: **end if**
13: $N_c \leftarrow N_c + 1$;
14: **end while**
15: **end for**
16: Form the next population of size N and update p^B;
17: **if** $T(p^B) < T(p_p^B)$ **then**
18: $s \leftarrow 0$; ▷ Reset the steady state counter
19: **else**
20: $s \leftarrow s + 1$; ▷ Increase the steady state counter
21: **if** $(s > s_M$ **and** $s \leq P)$ **then**
22: $\mathcal{S} \leftarrow$ LES; ▷ Switch to LES
23: **else if** $s > P$ **then**
24: $\mathcal{S} \leftarrow$ AB; $N \leftarrow N + \Delta N$; ▷ Switch to AB, increase N
25: $s \leftarrow 0$; ▷ Reset the steady state counter
26: **end if**
27: **end if**
Adaptation
28: $p_p^B \leftarrow p^B$; ▷ Update the best individual in the previous generation
29: Verify termination condition and update $done$;
30: **end while**
31: **return** the best solution p^B (in the last generation);

Fig. 4 Generation of a child solution in AMA-VRPTW. The repair procedure is executed only if the child is not feasible (rendered in *light pink*) (color figure online)

search (GES) (Nagata and Bräysy 2009), recently improved in our works (Nalepa and Czech 2012; Nalepa et al. 2014).

Guided ejection search is employed to minimize K at first, and then to create N feasible solutions (each containing K routes) (line 2). In GES, the search is started from a feasible solution in which each customer is served within a separate route (thus, $K = M$). Then, the algorithm repeatedly attempts to decrease K by one at a time. A route to be deleted is selected randomly, and the customers from this route are inserted into the ejection pool (EP), which contains unserved customers. Afterwards, the attempts of re-inserting the EP customers into the partial solution are undertaken. If there are no feasible (i.e., not violating capacity and time window constraints) insertions for a given customer taken from the EP, then the other customers from the partial solution are removed and inserted into the EP. GES executes until $K = K_{\min}$ (see Sect. 2.1), or its computation time exceeds the limit τ_K. It is then executed until N solutions with K routes are found, or the execution time of this process surpasses the limit τ_N (in this case, the solutions already found are copied and mutated—see Sect. 4.4—until N individuals are generated). The maximum computation time of minimizing K and generating the initial population is then $\tau_I = \tau_K + \tau_N$, where τ_K denotes the maximum time of minimizing K, and τ_N is the maximum time of generating N solutions.

4.2 Selection

The population of solutions is subsequently evolved in AMA-VRPTW to optimize T (lines 3–31). At first, N pairs (p_a, p_b) of individuals from the ith generation G_i are selected to create the generation $G_{(i+1)}$, according to the selection scheme $\mathcal{S}$ (Algorithm 1, line 4, see also Fig. 4). This selection scheme is adaptively determined during the optimization process (see Sect. 4.6). In this paper, we propose to combine the AB selection (AB), which has high exploration capabilities (Kawulok and Nalepa 2012; Nalepa and Czech 2013), with the scheme locally exploiting best individuals (we term it the local exploitation selection, LES) into the adaptive exploration–exploitation scheme (AE^2). AE^2 adaptively determines the execution mode (either explorative or exploitative) based on the optimization progress.

In AB, each individual p_i, $i \in \{1, 2, \ldots, N\}$, is selected as p_a at first. Then, the individual p'_i is chosen as p_b, such that $p_i \neq p'_i$. Each individual can be selected once as p_a, and once as p_b. Clearly, this selection takes $O(N)$ time. In LES, the population is sorted according to the fitness of the individuals (the lower T, the higher fitness), and divided into ϵ parts [it requires $O(N \log N)$ time]. Then, N/ϵ pairs of parents are drawn and crossed-over for each population part to exploit them independently. AE^2 dynamically switches between AB and LES to balance the exploration capabilities of the former scheme with the LES exploitation behavior (see Sect. 4.6 for details).

4.3 Crossover

For each pair of selected parent solutions (p_a, p_b), a new individual p_c is generated using the edge assembly crossover operator (EAX) (Algorithm 1, line 8). This operator was introduced for the TSP (Nagata 2006), and later adapted to the CVRP (Nagata 2007). Originally, the EAX was defined for undirected graphs. Then, it was enhanced for directed graphs to handle time window constraints of the VRPTW (Nagata et al. 2010). Its operation [taking $\mathcal{T}_{\text{EAX}}(M) = O(M^2)$ time, where M is a number of customers in each parent (Blocho 2013)], is visualized in Fig. 5 [the figure is inspired by Nalepa and Blocho (2014)]. It is worth mentioning that a solution obtained using the EAX may be infeasible, and its feasibility needs to be restored (see Sect. 4.4).

4.4 Repair, education and mutation

The repair, education and mutation procedures are the hill-climbing methods based on traditional neighborhoods for the VRPTW (Potvin and Rousseau 1995; Kindervater and Savelsbergh 1997; Nagata et al. 2010). Let $\mathcal{N}(\sigma)$ denote the neighborhood of σ, obtained by applying the following operators (moves): 2-opt* (Fig. 6), out-exchange (Fig. 7), out-relocate (Fig. 8), in-exchange (Fig. 9), and in-relocate (Fig. 10). To decrease an extremely large neighborhood size, we consider N_{v_i} nearest (in the Euclidean metric) customers for each v_i (Nagata et al. 2010).

A solution σ_c (represented by p_c) may be infeasible, and it undergoes the repair. In this process, local search moves are performed to decrease the penalty term of the generalized cost function $F_g(\sigma)$ (Nagata et al. 2010):

$$F_g(\sigma) = T(\sigma) + \beta_1 \cdot P_c(\sigma) + \beta_2 \cdot P_{tw}(\sigma), \tag{11}$$

where $T(\sigma)$ denotes the total travel distance of σ, and $P_c(\sigma)$ along with $P_{tw}(\sigma)$ are the penalty components representing the violations of the capacity and time window constraints. $P_c(\sigma)$ is the sum of the total capacity which exceeds in σ, and $P_{tw}(\sigma)$ denotes the sum of all time window violations.

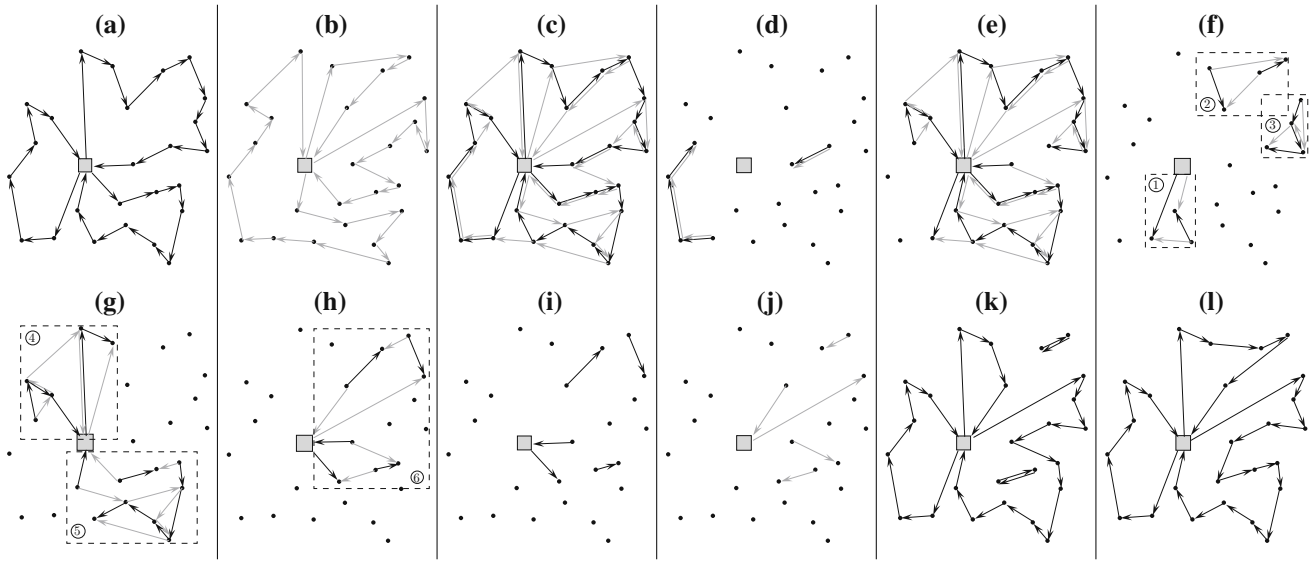

Fig. 5 Illustration of the EAX operator (Nagata et al. 2010) applied to p_a and p_b (solutions σ_a and σ_b, respectively): **a** the graph G_a corresponding to σ_a, **b** the graph G_b corresponding to σ_b, **c** the union of edges from G_a and G_b ($E_a \cup E_b$), **d** the intersection of edges from G_a and G_b ($E_a \cap E_b$), **e** the edges from ($E_a \cap E_b$) are removed from ($E_a \cup E_b$) to form G_{ab} (($E_a \cup E_b$)\($E_a \cap E_b$)), **f–h** six AB-cycles (consisting of G_{ab} edges traces alternately—E_a edges are traced forwardly, and E_b edges reversely), **i** the intersection of E_a and the selected E-set (E_S) (a random AB-cycle); here we pick up the AB-cycle from **h** as the E-set, **j** the intersection of E_b and E_S, **k** the intermediate solution with subroutes (($E_a \setminus (E_a \cap E_S)$) $\cup$ ($E_b \cap E_S$)), **l** the graph G_c corresponding to the child p_c after applying local moves (see Sect. 4.4) to the intermediate solution **k** for removing the subroutes

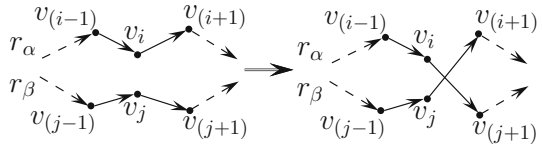

Fig. 6 2-opt* operator applied to the routes r_α and r_β

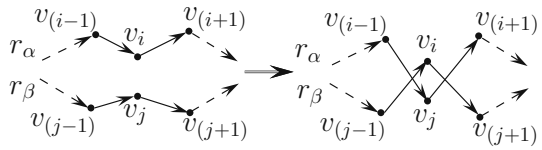

Fig. 7 Out-exchange operator applied to the routes r_α and r_β

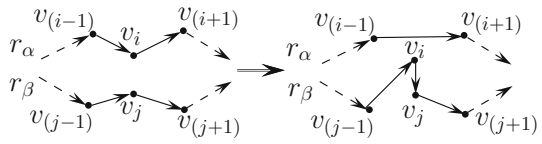

Fig. 8 Out-relocate operator applied to the routes r_α and r_β

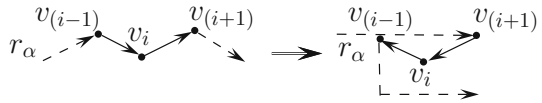

Fig. 9 In-exchange operator applied to the route r_α

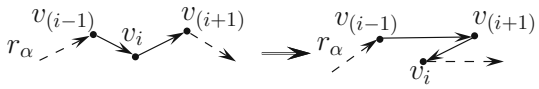

Fig. 10 In-relocate operator applied to the route r_α

The analysis of the beta scaling coefficients suggested to set $\beta_1 = \beta_2 = 1.0$, and $N_{v_i} = 50$ (Nagata et al. 2010).

In the repair procedure, the subneighborhoods [denoted as $\bigcup_{v \in r} \mathcal{N}(\sigma_c, v)$] for each infeasible customer (from a random infeasible route r belonging to σ_c) are created at first. A new solution σ_c', $\sigma_c' \in \bigcup_{v \in r} \mathcal{N}(\sigma_c, v)$, which minimizes the value of $(\beta_1 \cdot P_c(\sigma_c) + \beta_2 \cdot P_{tw}(\sigma_c))$, replaces the infeasible σ_c. This process executes until σ_c is feasible, or there are no repair moves left.

If p_c is feasible, then it is educated. Here, only feasible moves improving the solution quality, i.e., decreasing $T(p_c)$, are accepted. If there are no more improvement moves, then the education finishes. Afterwards, p_c is mutated by at most I_M moves (not violating the constraints). The best feasible child p_c^B is updated if $T(p_c) > T(p_c^B)$ (Algorithm 1, line 9).

4.5 Adaptive number of children

The child solution inherits structure information from both parents (see Fig. 5). This inheritance strongly depends on the E-set selection process. Thus, creating a larger number

of children ($N_c > 1$) for each pair (p_a, p_b) is beneficial and increases the probability of obtaining a well-adapted individual. However, if N_c is very large, then the search convergence and computation time may be jeopardized and significantly slowed down since creating a feasible child takes $O(M^2)$ time (Blocho 2013). Intuitively, the number of children should depend on the current optimization progress, since it is intrinsically dynamic and adaptive.

Here, we propose to keep generating child individuals p_c until a solution which is better (i.e., it has a shorter total travel distance T) than at least one of its parents is found (Algorithm 1, lines 10–12). This suggestion is based on the mentioned inheritance observation. More precisely, it means that the parent solutions could not be further improved by local refinement moves, and combining them with other (perhaps less-fitted) individuals is necessary to guide the search efficiently. Therefore, more global structure changes introduced by the EAX operator are crucial to escape from locally optimal solutions. Since crossing over two individuals does not guarantee obtaining a child better than the parents, we define the maximum number of children ($N/2$) which cannot be exceeded during the recombination process. This upper bound dynamically increases along with the increase of the population size N. On the one hand, it maximizes the probability of exploiting (usually) more diversified individuals. On the other hand, this limit prevents from generating too many children which do not contribute to the population layout and will be discarded.

4.6 Adaptation

The recombination process, which is conducted for each pair (p_a, p_b), is followed by forming the next population (Algorithm 1, line 16). It depends on the current selection scheme $\mathcal{S}$ in AE^2 (see Sect. 4.2). In the case of AB selection, the best child p_c^B, generated for a pair (p_a, p_b), replaces the parent p_a in $G_{(i+1)}$, only if $T(p_c^B) < T(p_a)$. In LES, N best solutions are selected to form $G_{(i+1)}$ from the set of $2 \cdot N$ individuals, containing N best children and the current population G_i (N solutions). It is worth noting that in both cases the best individual (i.e., with the largest fitness) in the population survives. Finally, the best solution p^B found up to date is updated if it is necessary (line 16).

After creating a new population, the adaptation process is carried out. First, it is verified if the best solution has been improved since the last generation (line 17). If so, the current configuration of the AMA-VRPTW settings is kept for further exploitation. Also, the steady-state counter s, indicating the number of consecutive generations without any improvement in the fitness of the best individual, is reset (line 18). Otherwise, if p^B has not been updated, s is increased (line 20).

In the proposed selection scheme (AE^2), AB and LES selections are dynamically switched between each other to balance both exploration and exploitation of the solution space. The selection scheme $\mathcal{S}$ is changed to LES (it is AB at the beginning to explore the initial population, see line 1) for a better local exploitation of the subpopulations of size N/ϵ once s exceeds s_{M} (line 22). Here, $s_{\mathrm{M}} = N/4$ is the maximum steady-state selection counter. If s surpasses $P = N/2$ (the maximum steady-state population counter), then $\mathcal{S}$ is set back to AB, and ΔN new individuals, where $\Delta N = N_I$, are added to the current population, to explore new regions of the search space (line 24). Additionally, the counter s is reset after introducing new genetic material (line 25). Note that other mechanisms for introducing new individuals, e.g., the population re-generation process, are not employed in AMA-VRPTW, since the population diversity is increased while appending new individuals to the population.[1] It is worth noting that s_{M} and P depend on the current population size N. This approach allows for increasing the probability of crossing over a larger number of unique pairs of individuals in both explorative (AB) and exploitative (LES) schemes. Thus, the probability of obtaining a well-fitted individual inheriting valuable information from both parents grows.

Finally, the best individual in the previous generation (p_p^B) is updated (line 28), and the stopping condition is verified (line 29). The algorithm is terminated if its execution time exceeds the maximum time limit τ. Also, it can be stopped if p^B is not improved for a given number of consecutive generations, or if a solution of an acceptable quality is already found. Then, the best solution in the last generation (p^B) is returned (line 31). The crossover, repair, and education procedures are the most time-consuming parts of AMA-VRPTW, hence its time complexity is $O(M^2)$.

5 Experimental results

5.1 Setup

Extensive experiments were conducted to investigate the performance of AMA-VRPTW, and to compare its efficacy with other state-of-the-art techniques. AMA-VRPTW was implemented in C++ language and run on a computer equipped with an Intel Core i7 2.3 GHz (16 GB RAM) processor. Its maximum execution time was limited by $\tau = 3.2$ min for Solomon's set, and $\tau = 4.5$ min for Gehring and Homberger's set (τ includes τ_I). The initial population is very small ($N_I = 10$, and $\Delta N = N_I$). In LES, we divide the population into two equinumerous parts ($\epsilon = 2$). The maximum number of local moves in the mutation procedure is $I_M = 100$.

[1] New individuals usually are of a lower quality than those already optimized, and differ from them significantly.

Table 1 Abbreviations of the state-of-the-art methods taken for comparison (S—Solomon's set, GH—Gehring and Homberger's set)

Symbol	Set	Method	References
2SHLS	S	Two-stage hybrid local search	Bent and Van Hentenryck (2004)
TSH	S	Tabu search heuristics	Ho and Haugland (2004)
HACS	S	Hybrid ant colony system	Chen and Ting (2005)
2SH-EP	S|GH	Two-stage heuristics with ejection pools	Lim and Zhang (2007)
GH	GH	A general heuristics for VRPs	Pisinger and Ropke (2007)
ILS	S|GH	Iterated local search	Ibaraki et al. (2008)
AGEA	S|GH	Arc-guided evolutionary algorithm	Repoussis et al. (2009)
BPLNS	S|GH	A branch-and-price-based LNS	Prescott-Gagnon et al. (2009)
EAMA$^{\alpha}$	S	Edge assembly memetic algorithm ($N = 200$)	Nagata et al. (2010)
EAMA$^{\beta}$	S|GH	Edge assembly memetic algorithm ($N = 100$)	Nagata et al. (2010)
GP–GA	S	Goal programming and genetic algorithm	Ghoseiri and Ghannadpour (2010)
AC-IH	S	Ant colony with insertion heuristics	Balseiro et al. (2011)
LNS	S	Large neighborhood search	Hong (2012)
MA-DM	S|GH	Memetic algorithm with diversity management	Vidal et al. (2013)
CPSO	S	Hybrid chaos-particle swarm optimization	Hu et al. (2013)
AC-CH	S	Ant colony with characterization heuristics	Gomez et al. (2014)

5.2 Datasets

AMA-VRPTW was tested on two classical benchmarks of large-scale VRPTW problem instances proposed by Solomon (1987), and Gehring and Homberger (1999). They became the standard sets for evaluating emerging algorithms to solve the VRPTW, since they reflect various real-life scheduling circumstances. All large-scale tests (in both sets) are split into subclasses, containing customers grouped into clusters (C subclass), dispersed randomly on the map (R subclass), and those containing a mix of both clustered and randomized customers (RC subclass). Among these subclasses, it is possible to distinguish problems with smaller vehicle capacities and considerably short time windows (C1, R1, and RC1), and those with larger vehicle capacities and longer scheduling horizons (C2, R2, and RC2). These characteristics strongly influence the structure of final solutions, e.g., a larger fleet is necessary to serve customers in the case of small truck capacities and tight time windows.

In each Solomon's instance, there are $M = 100$ customers to serve, whereas in Gehring and Homberger's set, there are problems with various M's, where $M \in \{200, 400, 600, 800, 1000\}$. The number of tests varies (from 8 to 12) between Solomon's subclasses (56 instances in total). Each Gehring and Homberger's subclass contains 10 problems (60 instances in total for each M). Tests are distinguished by their unique names: $\alpha\gamma$ in Solomon's set, and α_β_γ in Gehring and Homberger's set, where α denotes the subclass (C1, C2, R1, R2, RC1, and RC2), β relates to M (2 for 200, 4 for 400, and so forth), γ is the test identifier ($\gamma \in \{01, 02, \ldots, 12\}$ for Solomon's set, $\gamma \in \{1, 2, \ldots, 10\}$ for Gehring and Homberger's set). In this study, we consider the entire Solomon's set, and only 200-customer Gehring and Homberger's instances.

We compare AMA-VRPTW with a number of state-of-the-art methods for solving the VRPTW. Table 1 summarizes the algorithms taken for comparison for Solomon's and Gehring and Homberger's sets (it is common that the authors report the results obtained using their algorithms only for one benchmark set). Also, we present the world's best (currently known) results published at the SINTEF website[2] Finally, we report the sensitivity analysis on various AMA-VRPTW components followed by the two-tailed Wilcoxon test to verify statistical significance of the results. This study shows how these components influence the AMA-VRPTW performance, and how they contribute to the convergence capabilities of the proposed MA.

5.3 Creating the initial population

The first stage of AMA-VRPTW consists in minimizing the number of routes K, and generating the initial population of N solutions (see Sect. 4 for more details). Here, we utilized GES (Nagata and Bräysy 2009), which has a relatively high time complexity $O(M^{2.7})$, where M is the number of clients to serve (Blocho 2013). Hence, the maximum execution time τ_I is imposed on this procedure. As already mentioned, if τ_I is exceeded, then the algorithm is terminated and the solutions

[2] See http://www.sintef.no/Projectweb/TOP/VRPTW/; reference date: June 20, 2014. Note that the world's best results are a set of solutions obtained using *various* algorithms—both sequential and parallel.

Table 2 Average time τ_A^K (in s) of generating a feasible solution with K_B routes using GES (out of 200 runs) for Solomon's and Gehring and Homberger's sets (superscripts S and GH, respectively)

Subclass	$\tau_A^{K(S)}$	$\tau_A^{K(GH)}$
C1	0.06	0.23
C2	0.16	0.38
R1	5.25|0.62	0.08
R2	2.22|0.11	0.52|0.23
RC1	1.71|0.61	0.17
RC2	0.15	0.79|0.27

For some subclasses indicated the average time τ_E^K excluding the most time-consuming instances ($\tau_A^K|\tau_E^K$)—see Table 3

Table 3 Average time $\tilde{\tau}_A^K$ (in s) of generating a feasible solution with K_B routes using GES (out of 200 runs) for the most time-consuming VRPTW instances

Solomon's		Gehring and Homberger's	
Instance	$\tilde{\tau}_A^K$	Instance	$\tilde{\tau}_A^K$
R104	37.60	R2_2_1	3.15
R112	19.15	RC2_2_5	2.66
R204	1.31	RC2_2_6	3.04
R207	9.44	–	
R211	12.79	–	
RC106	19.15	–	

already found are copied and mutated until N individuals are generated. We generate a pool of $20 \cdot N_I$ solutions at first (within τ_I). Solutions are taken from this pool when N is increased on the fly, as discussed earlier. If $N > 20 \cdot N_I$ at some point during the optimization, then GES is run to generate new (missing) individuals.

Although the theoretical complexity of GES is significant, it runs very fast in practice, and the above-mentioned situation of copying and mutating already found solutions happened for less than 2.6 % of the considered instances. The average execution time τ_A^K (out of 200 independent runs) of generating a single feasible solution (with $K = K_B$, where K_B is the world's best minimum K) is given in Table 2. For some subclasses (R1, R2, and RC1 for Solomon's set, and R2 and RC2 for Gehring and Homberger's set), we present the average time (τ_E^K) excluding the most computationally intensive tests (Table 3). In most cases, the average time necessary for generating a solution with $K = K_B$ is well below 0.4 s for both benchmarks (Table 2), and it is negligible compared with the execution time of AMA-VRPTW. Solving only four Solomon's instances took more than 10 s on average (Table 3). If these tests are excluded from τ_A^K for Solomon's set, then it is $\tau_A^K = 0.29$ s. The results indicate that less-structured tests containing randomly scattered customers (R and RC subclasses) are more difficult to solve by GES in a short time. Since AMA-VRPTW is independent from the stage of generating the initial population of solutions, GES can be conveniently replaced by another, perhaps more efficient, route minimization algorithm without affecting the performance of AMA-VRPTW.

5.4 Comparison with other algorithms

We compare the performance of AMA-VRPTW with other state-of-the-art techniques mentioned in Sect. 5.2. The results are presented for each subclass (C1, C2, R1, R2, RC1, and RC2), for both benchmark sets. Each test (i.e., for each problem instance) was repeated 5 times, and the best results were averaged across subclasses (see Tables 4, 6). We indicate the processor on which a given algorithm was executed (P3, P4, Opt, Cent, and Xe stand for Pentium 3, Pentium 4, Opteron, Centrino and Xeon, respectively), along with its execution time τ for one problem instance, including τ_I [if a given method was run (x) times for a test, then the best result out of x executions was selected $(1/x)$, and $\tau = x \times \tau_s$, where τ_s is the time of a single run]. The quoted computation times are difficult to compare directly due to different computer architectures used for experiments; however, they give a rough overview about the algorithms' performance and behavior. For some techniques, the authors did not quote certain experimental settings. This is indicated by the "n/a" symbol in the appropriate column. Finally, the last rows in Tables 4 and 6 indicate the world's best results. The results are given in a form $K|T$, where K and T are the best minimum number of routes and the best minimum total travel distance found using the corresponding method, averaged for each subclass. Additionally, we show the best and the average AMA-VRPTW results for each instance separately (Tables 5, 7).

For the Solomon's tests (Table 4), a majority of the investigated state-of-the-art algorithms converged to the best-known K_B. It is worth noting that solutions in which customers are served by a larger number of trucks are usually characterized by a shorter total travel distance. However, the main objective of the VRPTW is to minimize K, thus these solutions are of a lower quality than those with a smaller K. The most competitive results are delivered by the edge-assembly MA (EAMA), MA with the diversity management (MA-DM), and the proposed AMA-VRPTW. For EAMA, two variants of the algorithm were executed, with different population sizes ($N = 20{,}000/M = 200$, where M denotes the number of customers, in EAMA$^\alpha$, and $N = 100$ in EAMA$^\beta$) as suggested by Nagata et al. (2010). The results show that N significantly affects the algorithm performance. Similarly, all MA-DM parameters are fixed and need to be tuned before the execution. This is an important issue since the whole optimization process must be run multiple times to determine (i.e., to tune) adequate parameter values. Contrary to that, AMA-VRPTW adaptively controls its parame-

Table 4 Comparison of the results obtained using various methods on Solomon's VRPTW instances (100 customers)

Method ↓	CPU↓	Subclass → τ ↓ (min.)	C1 $K\|T$	C2 $K\|T$	R1 $K\|T$	R2 $K\|T$	RC1 $K\|T$	RC2 $K\|T$
2SHLS (1/5)	SUN Ultra	5 × 120.0	10.0|828.38	3.0|589.86	11.92|1213.25	2.73|966.37	11.50|1384.22	3.25|1141.24
TSH (1/1)	SPARCII–440M	25.45[a]	10.0|833.76	3.0|592.60	13.50|1247.17	3.64|962.18	13.38|1431.94	4.00|1146.29
HACS (1/1)	P3–1.0G	6.3[a]	10.0|828.76	3.0|589.86	12.83|1203.56	3.09|932.23	12.50|1363.84	3.75|1079.81
2SH–EP (1/1)	P4–3.0G	38.5	10.0|828.38	3.0|589.86	11.92|1213.61	2.73|961.05	11.50|1385.56	3.25|1121.82
ILS (1/1)	P4–2.8G	16.7	10.0|828.38	3.0|589.86	12.00|1217.99	2.73|967.97	11.63|1384.67	3.25|1128.77
AGEA (1/3)	P4–3.0G	3 × 17.9	10.0|828.38	3.0|589.86	11.92|1210.82	2.73|952.67	11.50|1384.30	3.25|1119.72
BPLNS (1/5)	Opt–2.3G	5 × 30.0	10.0|828.38	3.0|589.86	11.92|1210.34	2.73|955.74	11.50|1384.16[d]	3.25|1119.44
EAMA$^{\alpha}$ (1/5)	Opt–2.4G	5 × 5.0	10.0|828.38	3.0|589.86	11.92|1210.34	2.73|951.03	11.50|1384.16[d]	3.25|1119.24
EAMA$^{\beta}$ (1/1)	Opt–2.4G	3.2	10.0|828.38	3.0|589.86	11.92|1210.34	2.73|952.08	11.50|1384.72	3.25|1119.45
GP–GA (1/10)	(n/a)–1.6G	10× (n/a)	10.0|828.38	3.0|591.49	12.92|1228.60	3.36|1063.94	12.75|1392.09	3.75|1162.40
AC-IH (1/10)	Cent–1.7G	10 × 5.0	10.0|828.38	3.0|589.86	11.92|1210.60	2.73|952.30	11.50|1384.21	3.25|1119.41
LNS (1/1)	Core2Duo–2.40G	3.02[a]	10.0|833.10	3.0|590.31	12.25|1218.28	3.27|964.11	12.13|1369.57	3.75|1131.18
MA-DM (1/5)	Xe–2.93G	5 × 2.68	10.0|828.38	3.0|589.86	11.92|1210.69	2.73|951.51	11.50|1384.17	3.25|1119.24
CPSO[b] (n/a)	AMDTurion–2.1G	n/a	10.0|828.38	3.0|589.86	11.92|1215.78	2.73|952.98	11.50|1414.24	3.25|1136.00
AC-CH (1/1)	Xe–1.86G	30.0	10.0|829.59	3.0|593.88	12.50|1234.88	2.82|1057.42	11.88|1441.89	3.38|1146.50
AMA-VRPTW (1/5)	i7–2.3G	5 × 3.2[c]	10.0|828.38	3.0|589.86	11.92|1210.34	2.73|951.80	11.50|1384.17	3.25|1119.24
World's best	–	–	10.0|828.38	3.0|589.86	11.92|1210.34	2.73|951.03	11.50|1384.17	3.25|1119.24

[a] The average execution time
[b] The authors quoted the full results only for the initial 500 iterations of their algorithm
[c] For R104, R112, R211, and RC106 instances, we increased the maximum AMA-VRPTW time to $\tau = 5.0$ min
[d] Note that $T < T_B$ (for $K = K_B$), but these results have not been confirmed by SINTEF

Table 5 The average and the minimum total travel distance ($T_A|T$) (out of 5 runs) obtained using AMA-VRPTW on Solomon's VRPTW instances (100 customers) (in boldface indicated T's equal to the world's best T_B)

Subclass → Instance ↓	C1 $T_A\|T$	C2 $T_A\|T$	R1 $T_A\|T$	R2 $T_A\|T$	RC1 $T_A\|T$	RC2 $T_A\|T$
1	**828.94|828.94**	**591.56|591.56**	**1650.80|1650.80**	1253.02|**1252.37**	**1696.95|1696.95**	**1406.94|1406.94**
2	**828.94|828.94**	**591.56|591.56**	**1486.12|1486.12**	**1191.70|1191.70**	**1554.75|1554.75**	1366.93|**1365.64**
3	**828.06|828.06**	**591.17|591.17**	**1292.68|1292.68**	941.88|**939.50**	1261.95|**1261.67**	1063.26|**1049.62**
4	**824.78|824.78**	**590.60|590.60**	**1007.31|1007.31**	833.09|828.78	**1135.52|1135.52**	**798.46|798.46**
5	**828.94|828.94**	**588.88|588.88**	**1377.11|1377.11**	995.93|994.43	**1629.44|1629.44**	**1297.65|1297.65**
6	**828.94|828.94**	**588.49|588.49**	1255.59|**1252.03**	910.51|**906.14**	**1424.73|1424.73**	1149.32|**1146.32**
7	**828.94|828.94**	**588.29|588.29**	1108.28|**1104.66**	895.90|**890.61**	1231.52|**1230.48**	1064.87|**1061.14**
8	**828.94|828.94**	**588.32|588.32**	963.48|**960.88**	728.02|**726.82**	1141.34|**1139.82**	829.18|**828.14**
9	**828.94|828.94**	–	1195.27|**1194.73**	910.37|**909.16**	–	–
10	–	–	**1118.84|1118.84**	944.32|**939.37**	–	–
11	–	–	1096.73|1096.73	892.61|891.11	–	–
12	–	–	986.83|**982.14**	–	–	–

$K = K_B$ for each problem instance

Table 6 Comparison of the results obtained using various methods on Gehring and Homberger's large-scale VRPTW instances (200 customers)

Method ↓	CPU↓	Subclass → τ ↓ (min.)	C1 $K\|T$	C2 $K\|T$	R1 $K\|T$	R2 $K\|T$	RC1 $K\|T$	RC2 $K\|T$
2SH–EP (1/1)	P4–3.0G	93.2	18.9|2726.11	6.0|1834.24	18.2|3639.60	4.0|2950.09	18.0|3205.51	4.3|2574.10
GH (1/10)	P4–3.0G	10 × 7.7	18.9|2721.52	6.0|1832.95	18.2|3631.23	4.0|2949.37	18.0|3212.28	4.3|2556.87
ILS (1/1)	P4–2.8G	33.0	18.9|2732.03	6.0|1834.83	18.2|3665.77	4.0|2965.64	18.0|3287.61	4.3|2562.56
AGEA (1/3)	Opt–2.3G	90.0	18.9|2721.90	6.0|1833.36	18.2|3640.11	4.0|2941.99	18.0|3224.63	4.3|2554.33
BPLNS (1/5)	Opt–2.3G	5 × 53.0	18.9|2718.77	6.0|1831.59	18.2|3615.69	4.0|2937.67	18.0|3192.56	4.3|2559.32
EAMA[a] (1/5)	Opt–2.4G	5 × 4.1	18.9|2718.41	6.0|1831.64	18.2|3612.36	4.0|2929.41	18.0|3178.68	4.3|2536.22
MA-DM (1/5)	Xe–2.93G	5 × 8.4	18.9|2718.41	6.0|1831.59	18.2|3613.16	4.0|2929.41	18.0|3180.48	4.3|2536.20
AMA-VRPTW (1/5)	i7–2.3G	5 × 4.5	18.9|2718.41	6.0|1831.59	18.2|3627.30	4.0|2930.06	18.0|3226.78	4.3|2537.72
World's best	–	–	18.9|2718.41	6.0|1831.59	18.2|3611.86	4.0|2929.41	18.0|3176.23	4.3|2535.88

[a] Only one variant of EAMA is tested here since $N = 20{,}000/200 = 100$ [see Nagata et al. (2010)]

ters during the search—it starts from a very small population, which is subsequently increased only if it is necessary. The results show that AMA-VRPTW matched the best-known solutions for almost all subclasses.

In Table 5, we report the detailed AMA-VRPTW results for Solomon's benchmark, showing the average and the best travel distances, T_A and T, respectively. For 32 (out of 56) problem instances (57 %) T_A matched T, whereas for 52 tests (93 %) AMA-VRPTW managed to find the best-known solution within 5 runs. It indicates high convergence capabilities of the algorithm, and this observation is confirmed by the maximum relative difference between T_A and T, given as $\delta_M = \max\{\delta^{(\alpha\gamma)}\}$, where $\delta^{(\alpha\gamma)} = (T_A^{(\alpha\gamma)} - T^{(\alpha\gamma)})/T^{(\alpha\gamma)}$ for each problem instance $\alpha\gamma$, which is $\delta_M = 1.3 \cdot 10^{-2}$ for RC203. The subclasses with clustered customers are not only easy to solve with respect to the fleet size (Table 2), but they are also very efficiently optimized by AMA-VRPTW when the total distance is considered (see C1 and C2 in Table 5).

The results obtained for Gehring and Homberger's benchmark tests are presented in Table 6, along with the detailed AMA-VRPTW results in Table 7. Although the search space is significantly larger here, C1 and C2 subclasses are solved successfully by most of the methods. However, subclasses containing randomized customers (especially with tight time windows) appeared to be challenging (see R1 and RC1 in Table 6) for both minimizing K and T. Similarly, the MAs (EAMA, MA-DM, and AMA-VRPTW) deliver the best (and the most stable among different subclasses) results. Also, a branch-and-price-based neighborhood search (BPLNS) offers competitive results, yet its computation time is significantly (at least 6.31× compared with MA-DM) larger than the time of the mentioned evolutionary techniques. As remarked previously, EAMA and MA-DM parameters must

Table 7 The average and the minimum total travel distance ($T_A|T$) (out of 5 runs) obtained using AMA–VRPTW on Gehring and Homberger's large-scale VRPTW instances (200 customers) (in boldface indicated T's equal to the world's best T_B)

Subclass → Instance ↓	C1 $T_A\|T$	C2 $T_A\|T$	R1 $T_A\|T$	R2 $T_A\|T$	RC1 $T_A\|T$	RC2 $T_A\|T$
1	**2704.57**|**2704.57**	**1931.44**|**1931.44**	4795.04|4795.04	**4483.16**|**4483.16**	3746.25|3636.70	3114.17|**3099.53**
2	2933.45|**2917.89**	**1863.16**|**1863.16**	4078.52|4055.95	3651.66|3621.20	3377.44|3312.89	2825.82|**2825.33**
3	2716.19|**2707.35**	1775.65|**1775.08**	3421.16|3388.03	2895.16|2881.15	3094.08|3034.74	2614.86|2604.09
4	2648.69|**2643.31**	1710.09|**1703.43**	3090.56|3075.22	1986.45|**1981.30**	2931.00|2872.10	2069.18|2048.77
5	**2702.05**|**2702.05**	1879.22|**1878.85**	4126.45|4111.84	3368.69|**3366.79**	3511.51|3419.76	2916.82|**2911.46**
6	**2701.04**|**2701.04**	**1857.35**|**1857.35**	3666.99|3618.34	2925.76|2914.11	3465.78|3401.36	2884.27|**2873.12**
7	**2701.04**|**2701.04**	**1849.46**|**1849.46**	3185.21|3171.89	2462.35|2452.66	3308.23|3273.47	2547.05|2525.83
8	2779.26|**2775.48**	1821.87|**1820.53**	2969.87|2958.19	1863.07|1849.98	3204.66|3168.96	2323.47|2297.44
9	2690.41|**2687.83**	1830.84|**1830.05**	3811.18|3792.26	3101.64|3095.27	3192.70|3121.80	2190.84|2175.98
10	2652.79|**2643.51**	1807.11|**1806.58**	3312.62|3306.21	2660.30|**2654.97**	3076.42|3025.99	2022.87|**2015.61**

$K = K_B$ for each problem instance

be set beforehand, and each configuration requires a separate execution (e.g., for three different N's, EAMA would require $3 \times 5 \times 4.1 = 61.5$ min. in this scenario). AMA-VRPTW does not suffer from this drawback as it adapts itself on the fly.

The detailed results of AMA-VRPTW (Table 7) confirm that it is very efficient in solving problems with clustered customers, and all C1 and C2 tests were solved to the best-known optimum within 5 algorithm runs. In total, the proposed MA retrieved the world's best results for 29 instances (48 %), among which for 9 tests (15 %) the minimum total travel distance T_B was gathered in every run. For this benchmark set, the average T_A values (out of 5 AMA-VRPTW runs) were close to the best results, what illustrate a good stability of the algorithm. The relative differences averaged for the subclasses ($\delta_A^{\alpha} = \sum_{(\alpha\_\beta\_\gamma)} \delta^{(\alpha\_\beta\_\gamma)}/10$, where $\delta^{(\alpha\_\beta\_\gamma)}$ denotes the relative difference for each instance α_β_γ in a given subclass α) are the largest in the case of RC1 and RC2 tests: $\delta_A^{RC1} = 2.0 \cdot 10^{-2}$ and $\delta_A^{RC2} = 0.5 \cdot 10^{-2}$.

AMA-VRPTW can be terminated once a solution of a desired quality has been found. Alternatively, the search may be finished if the best solution in the population is not improved for a number of consecutive generations g. It usually means that the optimization process converged to a solution which is not likely to be improved further, if g is sufficiently large. In Table 8, we present the average convergence time τ_c (excluding τ_I), along with the average number of generations g_c in AMA-VRPTW, after which the travel distance of the best solution could not be decreased further. The results confirm that AMA-VRPTW is extremely efficient in solving instances with clustered customers (C1 and C2). The subclasses with wide time windows and larger truck capacities (C2, R2, and RC2) are, in general, more difficult to solve in the case of Solomon's set, and take much more time (compared with C1, R1, and RC1 tests). The situation is opposite for Gehring and Homberger's large-scale tests. Here, instances with tight time window characteristics (C1, R1, and RC1) appeared to be computationally intensive. Since the converge times τ_c^{GH} are very close to the imposed time limit τ, increasing AMA-VRPTW maximum execution time can help improve the best individuals in the next generations. Similarly, the number of generations necessary for converging to high-quality results increases for demanding tests (R2 and RC2 for Solomon's set, C1, R1, and RC1 for Gehring and Homberger's set).

Table 8 Average convergence time τ_c (in s) and generation g_c (for Solomon's and Gehring and 200-customer Homberger's sets—superscripts S and GH, respectively)

Subclass	τ_c^{S}	g_c^{S}	τ_c^{GH}	g_c^{GH}
C1	0.77	3.53	81.77	62.93
C2	2.30	2.75	75.60	30.12
R1	50.01	43.65	244.17	134.94
R2	163.62	46.17	200.96	57.91
RC1	50.21	40.80	261.85	181.15
RC2	174.68	53.00	239.36	62.50

5.5 Analysis and discussion on adaptiveness

In a majority of EAs, a fixed number of children N_c, where $N_c \geq 1$, is generated for each pair of parents. It is easy to see that creating a larger number of child solutions may be beneficial, especially for MAs, in which each individual undergoes an additional education procedure. However, if the process of generating a child is time consuming, then it may significantly slow down the search and affect the convergence capabilities of the algorithm. Therefore, selecting an optimal number of children N_c is of a high importance, and is not

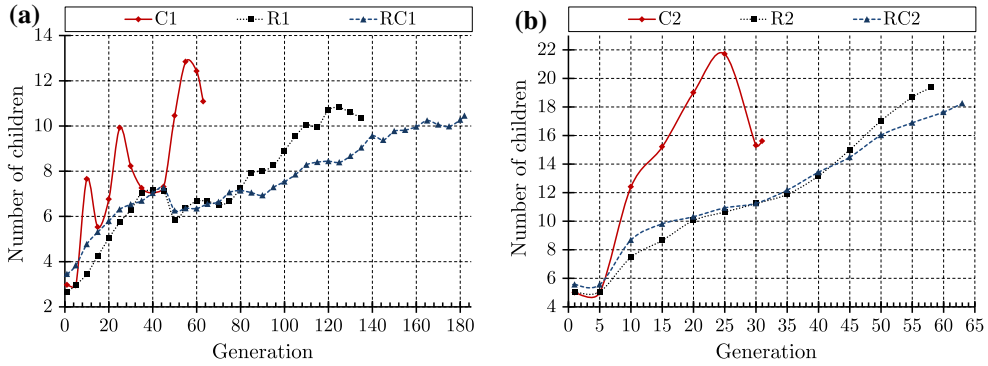

Fig. 11 The average number of children generated for each pair of parents for: **a** C1, R1, and RC1, and **b** C2, R2, and RC2 subclasses of the 200-customer Gehring and Homberger's set

Table 9 Abbreviations of the investigated MA variants—MA (N, N_c, selection scheme); A stands for adaptive

ID	Variant	N	N_c	Selection
MA_1	MA (100, 20, AB)	Constant, $N = 100$	Constant, $N_c = 20$	AB-selection
MA_2	MA (100, 20, A)	Constant, $N = 100$	Constant, $N_c = 20$	Adaptive (AE^2)
MA_3	MA (100, A, AB)	Constant, $N = 100$	Adaptive	AB-selection
MA_4	MA (100, A, A)	Constant, $N = 100$	Adaptive	Adaptive (AE^2)
MA_5	MA (A, 20, AB)	Adaptive	Constant, $N_c = 20$	AB-selection
MA_6	MA (A, 1, A)	Adaptive	Constant, $N_c = 1$	Adaptive (AE^2)
MA_7	MA (A, 20, A)	Adaptive	Constant, $N_c = 20$	Adaptive (AE^2)
MA_8	MA (A, A, AB)	Adaptive	Adaptive	AB-selection
AMA	AMA-VRPTW	Adaptive	Adaptive	Adaptive (AE^2)

a trivial task. In AMA-VRPTW, we proposed to adaptively adjust N_c (see Sect. 4 for details), so as it can change in time. Here, we present the average number of children for Gehring and Homberger's subclasses generated within g_c generations (g_c values are given in Table 8).

The results for all subclasses are depicted in Fig. 11. For difficult tests (with tight time windows), N_c grows slowly compared with the subclasses characterized by a longer scheduling horizon. It indicates that relatively small populations are exploited for a shorter time here, and generating more children does not help improve the quality of best individuals. Thus, the population size N adaptively grows to explore new regions of the search space by adding new genetic material. This increase of N is indicated by peaks in Fig. 11a, e.g., for C1.

Once N is enlarged, the average N_c drops, since the probability of obtaining a better (i.e., with a shorter T) child than at least one parent rapidly increases (new individuals appended to the population are usually of a lower quality than those already optimized). For R2 and RC2 tests, N_c continuously grows without a fast increase of N, since there are still high-quality neighboring solutions, and generating more children is advantageous (see Fig. 11b). Also, it affects the number of generations necessary to converge to the final solutions (see Table 8). Therefore, generating a large number of children is more exploitative and results in aggressive optimization of small populations. On the contrary, small N_c's and more rapid increase of N expose an explorative behavior of AMA-VRPTW. The appropriate search scheme (i.e., more exploitative or explorative) is adaptively controlled according to the current state of the search in the proposed algorithm.

5.6 Sensitivity analysis on method components

In AMA-VRPTW, we introduced some new adaptive components to address the problem of setting numerous MA parameters dynamically. In this section, we analyze how these procedures contribute to the AMA-VRPTW performance and execution time for Gehring and Homberger's tests with 200 customers. To determine the impact of each of these components, we defined several algorithm variants in which adaptive techniques for setting N, N_c, and the selection scheme, are replaced by the baseline approaches. The investigated AMA-VRPTW variants are summarized in Table 9. Each test was repeated 5 times, and the best results were averaged across subclasses. The minimum total travel distance T, along with the average convergence time τ_c and generation g_c, and the average time of processing a single generation τ_A^g, is given in Table 10 (the best T is indicated in boldface). As previously, $K = K_B$ for each test.

Table 10 Comparison of the results (the minimum total travel distance T, the average convergence time τ_c (in s), the average convergence generation g_c, and the average time of a single generation τ_A^g (in s)) obtained using different algorithm variants

Variant ↓	C1	C2	R1	R2	RC1	RC2
MA_1						
T	2720.07	1837.44	3721.14	2969.83	3432.39	2575.97
τ_c	111.59	102.56	277.82	322.29	301.88	315.53
g_c	20.05	6.77	16.82	6.34	24.44	7.52
τ_A^g	3.46	12.43	14.65	58.58	12.83	46.57
MA_2						
T	2720.04	1836.72	3714.25	2957.70	3419.78	2570.76
τ_c	141.01	144.87	277.72	318.26	302.16	319.72
g_c	14.30	5.57	15.65	5.82	23.06	7.10
τ_A^g	7.12	22.10	16.48	64.63	13.60	51.47
MA_3						
T	2718.62	1832.08	3680.48	2937.46	3340.50	2549.92
τ_c	129.26	92.56	267.71	275.71	297.73	280.03
g_c	23.64	10.46	35.54	17.75	46.23	19.30
τ_A^g	3.98	10.15	7.34	17.17	6.62	15.82
MA_4						
T	2718.56	1833.37	3681.78	2938.51	3355.82	2554.39
τ_c	127.36	87.87	274.11	272.78	297.65	293.83
g_c	23.73	9.85	37.71	17.64	48.65	17.96
τ_A^g	2.99	9.08	6.67	17.08	6.32	17.66
MA_5						
T	2719.64	1832.59	3649.20	2937.43	3272.49	2546.71
τ_c	66.64	81.75	205.27	194.64	261.24	216.58
g_c	49.94	28.70	108.02	42.56	176.44	55.22
τ_A^g	1.95	3.96	2.19	5.14	1.56	4.25
MA_6						
T	2720.10	1833.65	3661.09	2939.35	3294.61	2555.52
τ_c	99.62	90.10	242.04	229.13	253.29	242.80
g_c	130.66	60.22	274.40	110.36	325.05	125.63
τ_A^g	0.81	1.74	0.94	2.28	0.85	2.11
MA_7						
T	2718.87	1831.95	3647.68	2936.61	3270.71	2545.80
τ_c	62.86	63.07	195.50	211.73	240.76	206.80
g_c	46.72	22.86	116.75	44.42	168.66	55.92
τ_A^g	1.94	3.94	2.02	5.24	1.56	4.16
MA_8						
T	2718.56	1831.91	3637.91	2934.32	3243.53	2542.16
τ_c	69.99	67.03	210.46	154.68	256.71	209.76
g_c	41.98	22.39	104.76	43.50	162.62	56.06
τ_A^g	3.00	6.23	2.53	5.40	1.75	4.61
AMA						
T	**2718.41**	**1831.59**	**3627.30**	**2930.06**	**3226.78**	**2537.72**
τ_c	81.77	75.60	244.17	200.96	261.85	239.36
g_c	62.93	30.12	134.94	57.91	181.15	62.50
τ_A^g	2.14	4.29	2.13	4.39	2.18	4.81

Table 11 The level of statistical significance obtained using the two-tailed Wilcoxon test for each pair of the MA variants: $p \leq x$, where x is the value given in the table, p denotes the p value, and **no** indicates that $p > 0.05$.

	MA_1	MA_2	MA_3	MA_4	MA_5	MA_6	MA_7	MA_8	**AMA**
MA_1	–	**no**	0.01	0.01	0.01	0.01	0.01	0.01	0.01
MA_2		–	0.01	0.01	0.01	0.01	0.01	0.01	0.01
MA_3			–	0.01	**no**	0.01	**no**	**no**	0.05
MA_4				–	0.05	0.05	0.01	0.01	0.01
MA_5					–	0.01	**no**	0.01	0.01
MA_6						–	0.01	0.01	0.01
MA_7							–	0.01	0.01
MA_8								–	0.05

Each adaptive component applied separately in the MA significantly improves the results (see variants MA_2, MA_3 and MA_5, compared with the baseline MA_1 in Table 10). The adaptive number of children N_c and the population size N affect the convergence speed of the original MA by utilizing the incremental exploration of the solution space. Combining these two variants into MA_8 helps further decrease the execution time and obtain higher-quality results. Also, the adaptive selection scheme (AE^2) offers better exploitation behavior when combined with the mentioned variants (MA_4, MA_6 and MA_7). Due to a small value of N_c ($N_c = 1$) which was set a priori, small populations were not sufficiently exploited during the search (MA_6 resulted in larger T's). It confirms the necessity of setting N_c on the fly to handle the current search state efficiently. Finally, AMA-VRPTW in which all the discussed adaptive techniques are applied resulted in the best travel distances for each subclass. It proves a good stability of AMA-VRPTW, and shows that it delivers best asymptotic results within the assumed execution time.

The algorithm variants with constant $N = 100$ and $N_c = 20$ require significantly larger amount of time to process a single generation (see τ_A^g for MA_1 and MA_2). Since the adaptive selection is more exploitative, MA_2 handles a single generation slower, but allows for an intensive search resulting in better solutions (see Table 10). Incorporating the adaptive techniques for N_c and N significantly decreases the computation time. It is easy to see that generating a single child for each pair of parents (MA_6) is very fast, but the quality of solutions obtained by this algorithm variant drastically drops. Finally, the proposed AMA-VRPTW not only can deliver very high-quality solutions, but also it is very efficient in terms of the computation time (e.g., it is more than 13.3× faster compared with the static MA, in which the parameters are fixed during the optimization—see AMA-VRPTW and MA_1 for R2).

To verify the null hypothesis saying that "two variants of the MA (with adaptive and static parameters) lead to the same quality of final solutions", we performed the two-tailed Wilcoxon test for each pair of the algorithm variants (see Table 11).[3] It is easy to see that it can be rejected with a high probability for AMA-VRPTW compared with other MA variants. This means that the increase of the solutions quality is statistically significant for the proposed adaptive MA. It is worth noting that the difference between MA_3 (the MA with an adaptive scheme for determining the number of children), and some other adaptive MAs (MA_7 and MA_8) is not necessarily statistically significant. However, the latter variants outperformed MA_3 in terms of the execution time, and converged to high-quality solutions much faster (see Table 8 for more details). Also, the population size does not need to be specified for the adaptive algorithms prior to the optimization. This was a significant drawback of the MA_3 variant leading to a very time-consuming tuning.

6 Conclusions and future work

In this paper, a new adaptive memetic algorithm (AMA-VRPTW) for solving the VRPTW has been proposed. AMA-VRPTW adaptively adjusts its various parameters, including the population size, the number of children generated for each pair of parents during the recombination process, and the selection scheme, according to the current state of the optimization. The problem of determining proper algorithm parameters before the execution is very difficult in practice, and requires a large computational effort to validate each set of parameters. This is a significant drawback of other state-of-the-art algorithms. A noteworthy feature of AMA-VRPTW is its capability of balancing the exploration and exploitation of the search space. In AMA-VRPTW, the exploration of N individuals is followed by their intensive exploitation.

The extensive experimental study conducted for two standard benchmark sets proves the high convergence capabilities of AMA-VRPTW, and shows that it not only offers high-quality results but also executes very fast. Since AMA-

[3] Here, we analyze the best results (out of 5 runs) obtained for all 200-customer tests.

VRPTW converges to high-quality solutions extremely fast, it can be applied to commercial (real-time) applications in which travel costs are dynamic—they are updated according to the traffic information. We performed the sensitivity analysis, and demonstrated how various algorithm components influence the final results and the optimization process. The two-tailed Wilcoxon test showed the statistical significance of the results obtained using the considered variants of AMA-VRPTW with certain adaptive components switched off and on.

Our ongoing research is focused on incorporating the proposed adaptive algorithm into our parallel framework (Nalepa and Blocho 2014). We aim at conducting the experiments for full Gehring and Homberger's benchmark set (i.e., for each number of customers) using the adaptive parallel MA. Also, we work on new adaptive co-operation schemes of parallel processes to guide the search more efficiently. Finally, we plan to enhance local refinement procedures applied for optimizing the already-found solutions during the education process to improve the performance of AMA-VRPTW for instances with tight time windows. Another direction of our future work encompasses designing a self-adaptive MA which will evolve its parameters with time. Finally, we plan to apply AMA-VRPTW to other complex vehicle routing problems, especially the pickup and delivery problem with time windows.

Acknowledgments This research was supported by the National Science Centre under research Grant No. DEC-2013/09/N/ST6/03461. The work was performed using the infrastructure supported by the POIG.02.03.01-24-099/13 grant: "GeCONiI—Upper Silesian Center for Computational Science and Engineering". Also, we thank Academic Computer Centre in Gdańsk (CI TASK), where the computations of our project were carried out.

References

Abdallah KS, Jang J (2014) An exact solution for vehicle routing problems with semi-hard resource constraints. Comput Ind Eng 76(0):366–377. doi:10.1016/j.cie.2014.08.011

Baldacci R, Mingozzi A, Roberti R (2011) New route relaxation and pricing strategies for the vehicle routing problem. Oper Res 59(5):1269–1283

Baldacci R, Mingozzi A, Roberti R (2012) Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. Eur J Oper Res 218(1):1–6

Balseiro S, Loiseau I, Ramonet J (2011) An ant colony algorithm hybridized with insertion heuristics for the time dependent vehicle routing problem with time windows. Comput Oper Res 38(6):954–966

Banos R, Ortega J, Gil C, Márquez AL, de Toro F (2013) A hybrid meta-heuristic for multi-objective vehicle routing problems with time windows. Comput Ind Eng 65(2):286–296

Bard JF, Kontoravdis G, Yu G (2002) A branch-and-cut procedure for the vehicle routing problem with time windows. Transp Sci 36(2):250–269

Bektas T (2006) The multiple traveling salesman problem: an overview of formulations and solution procedures. Omega 34(3):209–219

Bent R, Van Hentenryck P (2004) A two-stage hybrid local search for the vehicle routing problem with time windows. Transp Sci 38(4):515–530

Blocho M (2013) A parallel memetic algorithm for solving the vehicle routing problem with time windows. PhD thesis, Silesian University of Technology, Gliwice

Blocho M, Czech Z (2012a) A parallel algorithm for minimizing the number of routes in the vehicle routing problem with time windows. In: Wyrzykowski R, Dongarra J, Karczewski K, Waniewski J (eds) Parallel Processing and applied mathematics, vol 7203. Lecture Notes in Computer ScienceSpringer, Berlin, Heidelberg, pp 255–265

Blocho M, Czech Z (2012b) A parallel EAX-based algorithm for minimizing the number of routes in the vehicle routing problem with time windows. In: High performance computing and communication 2012 IEEE 9th international conference on embedded software and systems (HPCC-ICESS), 2012 IEEE 14th International Conference on, pp 1239–1246

Blocho M, Czech ZJ (2013) A parallel memetic algorithm for the vehicle routing problem with time windows. In: Proceedings of the 2013 eighth international conference on P2P, parallel, grid, cloud and internet computing, 3PGCIC '13, pp 144–151

Bräysy O, Gendreau M (2005) Vehicle routing problem with time windows, part II: metaheuristics. Transp Sci 39(1):119–139

Chabrier A (2006) Vehicle routing problem with elementary shortest path based column generation. Comput Oper Res 33(10):2972–2990 (part Special Issue: Constraint Programming)

Chen CH, Ting CJ (2005) A hybrid ant colony system for vehicle routing problem with time windows. J East Asia Soc Transp Stud 6:2822–2836

Chiang WC, Russell R (1996) Simulated annealing metaheuristics for the vehicle routing problem with time windows. Ann Oper Res 63(1):3–27

Coltorti D, Rizzoli AE (2007) Ant colony optimization for real-world vehicle routing problems. SIGEVOlution 2(2):2–9

Cordeau JF, Desaulniers G, Desrosiers J, Solomon MM, Soumis F (2002) VRP with Time Windows. In: Toth P, Vigo D (eds) The vehicle routing problem, SIAM monographs on discrete mathematics and applications, vol 9. Philadelphia, PA, pp 157–193

Creput JC, Koukam A (2008) The memetic self-organizing map approach to the vehicle routing problem. Soft Computing 12(11):1125–1141. doi:10.1007/s00500-008-0281-4

Dantzig GB, Ramser JH (1959) The truck dispatching problem. Manag Sci 6(1):80–91

Eiben A, Hinterding R, Michalewicz Z (1999) Parameter control in evolutionary algorithms. Evol Comput IEEE Trans 3(2):124–141

El-Sherbeny NA (2010) Vehicle routing with time windows: an overview of exact, heuristic and metaheuristic methods. J King Saud Univ Sci 22(3):123–131

Feillet D, Dejax P, Gendreau M, Gueguen C (2004) An exact algorithm for the elementary shortest path problem with resource constraints: application to some vehicle routing problems. Networks 44(3):216–229

Gambardella LM, Taillard E, Agazzi G (1999) MACS-VRPTW: a multiple ant colony system for vehicle routing problems with time windows. In: New Ideas in optimization, McGraw-Hill, pp 63–76

Garey MR, Johnson DS (1990) Computers and intractability: a guide to the theory of NP-completeness. W. H. Freeman & Co., New York

Gehring H, Homberger J (1999) A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In: Proceedings of EUROGEN99-Short course on evolutionary algorithms in engineering and computer science, pp 57–64

Ghoseiri K, Ghannadpour SF (2010) Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm. Appl Soft Comput 10(4):1096–1107

Gomez C, Cruz-Reyes L, González JJ, Fraire HJ, Pazos RA, Martinez JJ (2014) Ant colony system with characterization-based heuristics for a bottled-products distribution logistics system. J Comput Appl Math B 259:965–977

Guan X, Zhang X, Han D, Zhu Y, Lv J, Su J (2014) A strategic flight conflict avoidance approach based on a memetic algorithm. Chin J Aeronaut 27(1):93–101

Ho S, Haugland D (2004) A tabu search heuristic for the vehicle routing problem with time windows and split deliveries. Comput Oper Res 31(12):1947–1964

Hong L (2012) An improved LNS algorithm for real-time vehicle routing problem with time windows. Comput Oper Res 39(2):151–163

Hosny MI, Mumford CL (2010) The single vehicle pickup and delivery problem with time windows: intelligent operators for heuristic and metaheuristic algorithms. J Heuristics 16(3):417–439

Hu W, Liang H, Peng C, Du B, Hu Q (2013) A hybrid chaos-particle swarm optimization algorithm for the vehicle routing problem with time window. Entropy 15(4):1247–1270

Ibaraki T, Imahori S, Nonobe K, Sobue K, Uno T, Yagiura M (2008) An iterated local search algorithm for the vehicle routing problem with convex time penalty functions. Discrete Appl Math 156(11):2050–2069

Irnich S, Villeneuve D (2006) The shortest-path problem with resource constraints and k-cycle elimination for $k \leq 3$. INFORMS J Comput 18(3):391–406

Jin Y, Hao JK, Hamiez JP (2014) A memetic algorithm for the minimum sum coloring problem. Comput Oper Res 43:318–327

Kallehauge B (2008) Formulations and exact algorithms for the vehicle routing problem with time windows. Comput Oper Res 35(7):2307–2330

Kawulok M, Nalepa J (2012) Support vector machines training data selection using a genetic algorithm. In: Gimelfarb G, Hancock E, Imiya A, Kuijper A, Kudo M, Omachi S, Windeatt T, Yamada K (eds) Structural, Syntactic, and statistical pattern recognition, Lecture Notes in Computer Science, vol 7626, Springer, Berlin Heidelberg, pp 557–565. doi:10.1007/978-3-642-34166-3_61

Kindervater G, Savelsbergh M (1997) Vehicle routing: handling edge exchanges. In: Aarts E, Lenstra J (eds) Local Search in combinatorial optimization, Wiley, pp 337–360

Kolen AWJ, Kan AHGR, Trienekens HWJM (1987) Vehicle routing with time windows. Oper Res 35(2):266–273

Larsen J (2004) Refinements of the column generation process for the vehicle routing problem with time windows. J Syst Sci Syst Eng 13(3):326–341

Li Y, Li P, Wu B, Jiao L, Shang R (2013) Kernel clustering using a hybrid memetic algorithm. Nat Comput 12(4):605–615

Li Y, Jiao L, Li P, Wu B (2014) A hybrid memetic algorithm for global optimization. Neurocomputing 134:132–139

Lim A, Zhang X (2007) A two-stage heuristic with ejection pools and generalized ejection chains for the vehicle routing problem with time windows. INFORMS J Comput 19(3):443–457

Liu R, Xie X, Garaix T (2014) Hybridization of tabu search with feasible and infeasible local searches for periodic home health care logistics. Omega 47(0):17–32. doi:10.1016/j.omega.2014.03.003

Marinaki M, Marinakis Y (2014) An island memetic differential evolution algorithm for the feature selection problem. In: Proc. NICSO, SCI, vol 512, Springer, pp 29–42

Marinakis Y, Marinaki M (2014) A bumble bees mating optimization algorithm for the open vehicle routing problem. Swarm Evol Comput 15(0):80–94. doi:10.1016/j.swevo.2013.12.003

Masson R, Ropke S, Lehud F, Pton O (2014) A branch-and-cut-and-price approach for the pickup and delivery problem with shuttle routes. Eur J Oper Res 236(3):849–862. doi:10.1016/j.ejor.2013.08.042 (vehicle Routing and Distribution Logistics)

Nagata Y (2006) New EAX crossover for large TSP instances. In: Runarsson T, Beyer HG, Burke E, Merelo-Guervs J, Whitley L, Yao X (eds) Parallel problem solving from nature—PPSN IX, vol 4193. Lecture Notes in Computer ScienceSpringer, Berlin Heidelberg, pp 372–381

Nagata Y (2007) Edge assembly crossover for the capacitated vehicle routing problem. In: Cotta C, Hemert J (eds) Evolutionary computation in combinatorial optimization, vol 4446. Lecture Notes in Computer Science, Springer, Berlin Heidelberg, pp 142–153

Nagata Y, Bräysy O (2009) A powerful route minimization heuristic for the vehicle routing problem with time windows. Oper Res Lett 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. Comput Oper Res 37(4):724–737

Nalepa J (2014) Adaptive memetic algorithm for the vehicle routing problem with time windows. In: Proceedings of the 2014 conference companion on genetic and evolutionary computation companion, ACM, New York, GECCO Comp '14, pp 1467–1468. doi:10.1145/2598394.2602273

Nalepa J, Blocho M (2014) Co-operation in the parallel memetic algorithm. Int J Parallel Program pp 1–28. doi:10.1007/s10766-014-0343-4

Nalepa J, Czech ZJ (2012) A parallel heuristic algorithm to solve the vehicle routing problem with time windows. Studia Informatica 33(1):91–106

Nalepa J, Czech ZJ (2013) New selection schemes in a memetic algorithm for the vehicle routing problem with time windows. In: Tomassini M, Antonioni A, Daolio F, Buesser P (eds) Adaptive and natural computing algorithms, vol 7824. Lecture Notes in Computer Science, Springer, Berlin Heidelberg, pp 396–405

Nalepa J, Kawulok M (2014) A memetic algorithm to select training data for support vector machines. In: Proceedings of the 2014 Conference on genetic and evolutionary computation, ACM, New York, GECCO '14, pp 573–580. doi:10.1145/2576768.2598370

Nalepa J, Blocho M, Czech Z (2014) Co-operation schemes for the parallel memetic algorithm. In: Wyrzykowski R, Dongarra J, Karczewski K, Waniewski J (eds) Parallel processing and applied mathematics. Lecture Notes in Computer Science, Springer, Berlin Heidelberg, pp 191–201

Niu Y, Wang S, He J, Xiao J (2014) A novel membrane algorithm for capacitated vehicle routing problem. Soft Comput, pp 1–12. doi:10.1007/s00500-014-1266-0

Ombuki B, Ross BJ, Hanshar F (2006) Multi-objective genetic algorithms for vehicle routing problem with time windows. Appl Intell 24:17–30

Pang KW (2011) An adaptive parallel route construction heuristic for the vehicle routing problem with time windows constraints. Expert Syst Appl 38(9):11,939–11,946

Petch R, Salhi S (2003) A multi-phase constructive heuristic for the vehicle routing problem with multiple trips. Discrete Appl Math 133(13):69–92

Pisinger D, Ropke S (2007) A general heuristic for vehicle routing problems. Comput Oper Res 34(8):2403–2435

Potvin JY, Rousseau JM (1993) A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. Eur J Oper Res 66(3):331–340

Potvin JY, Rousseau JM (1995) An exchange heuristic for routeing problems with time windows. J Oper Res Soc 46(12):1433–1446

Prescott-Gagnon E, Desaulniers G, Rousseau LM (2009) A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows. Networks 54(4):190–204

Repoussis P, Tarantilis C, Ioannou G (2009) Arc-guided evolutionary algorithm for the vehicle routing problem with time windows. Evol Comput IEEE Trans 13(3):624–647

Righini G, Salani M (2006) Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. Discrete Optim 3(3):255–273 (graphs and Combinatorial Optimization The Cologne/Twente Workshop on Graphs and Combinatorial Optimization)

Solomon MM (1987) Algorithms for the vehicle routing and scheduling problems with time window constraints. Oper Res 35(2):254–265

Tavares L, Lopes H, Lima C (2009) Construction and improvement heuristics applied to the capacitated vehicle routing problem. In: Nature biologically inspired computing, 2009. NaBIC 2009. World Congress on, pp 690–695

Thangiah S, Nygard K, Juell P (1991) Gideon: a genetic algorithm system for vehicle routing with time windows. In: Artificial intelligence applications, proceedings., Seventh IEEE Conference on, vol i, pp 322–328. doi:10.1109/CAIA.1991.120888

Vidal T, Crainic TG, Gendreau M, Prins C (2013) A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. Comput Oper Res 40(1):475–489

Zhong Y, Pan X (2007) A hybrid optimization solution to VRPTW based on simulated annealing. In: Automation and logistics, 2007 IEEE International Conference on, pp 3113–3117

Zhu KQ (2000) A new genetic algorithm for VRPTW. In: Proceedings of the international conference on artificial intelligence, p 311264