Computers & Operations Research 37 (2010) 724–737

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/cor

A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows

Yuichi Nagata[a,*], Olli Bräysy[b], Wout Dullaert[c,d]

[a]Interdisciplinary Graduate School of Science and Engineering, Tokyo Institute of Technology, 4259 Nagatsuta Midori-ku Yokohama, Kanagawa 226-8502, Japan
[b]Agora Innoroad Laboratory, Agora Center, P.O. Box 35, FI-40014 University of Jyväskylä, Finland
[c]Institute of Transport and Maritime Management Antwerp, University of Antwerp, Keizerstraat 64, 2000 Antwerp, Belgium
[d]Antwerp Maritime Academy, Noordkasteel Oost 6, 2030 Antwerp, Belgium

ARTICLE INFO

Available online 3 July 2009

Keywords:
Vehicle routing
Time windows
Memetic algorithm
Penalty function

ABSTRACT

In this paper, we present an effective memetic algorithm for the vehicle routing problem with time windows (VRPTW). The paper builds upon an existing edge assembly crossover (EAX) developed for the capacitated VRP. The adjustments of the EAX operator and the introduction of a novel penalty function to eliminate violations of the time window constraint as well as the capacity constraint from offspring solutions generated by the EAX operator have proven essential to the heuristic's performance. Experimental results on Solomon's and Gehring and Homberger benchmarks demonstrate that our algorithm outperforms previous approaches and is able to improve 184 best-known solutions out of 356 instances.

1. Introduction

The vehicle routing problem with time windows (VRPTW) is one of the most important and widely studied NP-hard combinatorial optimization problems in the operations research literature. The VRPTW can be described as the problem of determining a least cost routing plan to deliver goods from a single depot to a set of geographically scattered customers. The routing plan must be designed in such a way that each customer is visited exactly once by a single vehicle departing from and returning to the depot, the total of goods loaded in any vehicle must not exceed a vehicle capacity (capacity constraint), and each customer must be serviced with a given time interval (time window constraint). The latter constraint defines the VRPTW as a variant of the standard or capacitated VRP (CVRP). A common objective of the VRPTW is to minimize total travel distance, but a hierarchical objective of minimizing the number of routes (primary objective) and the total travel distance (secondary objective) is also frequently used.

The VRPTW provides the mathematical basis of many distribution management problems encountered in practice, ranging from route design for parcel deliveries in various industries, cash deliveries to banks and ATM terminals, school bus routing and so on, to more tactical and strategic decision support tools in which routing decisions are combined with fleet composition, inventory or warehouse location etc. Apart from its practical importance, the standard VRPTW is a challenging optimization problem of significant academic value as it is often used as a benchmark problem when new solution approaches are suggested for less-studied or novel VRPTW variants. Given its practical and academic importance, intensive research efforts have been directed towards the development of both exact and heuristic algorithms for the VRPTW and its variants. For an extensive review, the reader is referred to Cordeau et al. [1], Bräysy and Gendreau [2,3], and Golden et al. [4]. Most of exact and heuristic algorithms have been applied to the well-known benchmark problems of Solomon (100 customers) [10] and of Gehring and Homberger (200–1000 customers) [11] to compare their performance. Here we refer only to the recent exact and heuristic algorithms that have shown the best performance on these benchmarks according to the literature.

The current state of the art exact algorithms are proposed by Chabrier [7], Irnich and Villeneuve [8], Jepsen et al. [5,6] and Kallehauge et al. [9]. For exact approaches, the single objective of minimizing the total travel distance is considered by tradition. Up to the present date, 45 instances out of 56 in Solomon's benchmarks have been solved to optimality [5]. However, it sometimes needs more than 5 h to solve non-structured instances whereas some instances having a certain structure and/or very tight time windows can be solved within a minute. For larger instances in the Gehring and Homberger benchmarks consisting 300 instances, only seven

* Corresponding author at: Interdisciplinary Graduate School of Science and Engineering, Tokyo Institute of Technology, 4259 Nagatsuta Midori-ku Yokohama, Kanagawa 226-8502, Japan. Tel.: +81 45 924 5677; fax: +81 45 924 5442.
E-mail address: nagata@fe.dis.titech.ac.jp (Y. Nagata).

0305-0548/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2009.06.022

200-customer instances, one 400-customer instance and one 1000-customer instance have been solved to optimality [9]. Here, one should note that these instances have a specific structure and/or very tight time windows and thus instances of this size can only be solved under certain conditions.

The current state-of-the-art heuristics for the VRPTW consist of evolution strategies [12,13], large neighborhood searches [14–16], iterated local searches [17,18], and multi-start local searches [17,19]. Here one should note that the hierarchical objective is considered in these methods and that most of heuristic methods consider this objective by tradition. Given the hierarchical objective, the state-of-the-art heuristics are all based on a two-stage approach [32,14] where the number of routes is minimized in the first stage and the total travel distance is then minimized in the second stage. This two-stage approach allows us to independently develop algorithms for the route minimization and for the distance minimization. Therefore, developing powerful algorithms either for route minimization or for distance minimization is also justified because they can be used as a basis for new metaheuristic approaches based on the two-stage approach.

In this paper, we develop a penalty-based memetic algorithm for the VRPTW by extending the edge assembly memetic algorithm (EAMA) of Nagata [20] for the capacitated VRP (CVRP). As such, the suggested solution approach belongs to the class of memetic algorithms (MAs) [21]. MA is a population-based heuristic search approach that combines evolutionary algorithm (EA) for the global, more distant search (exploration), with local search algorithm to organize a more intensive local search (exploitation). For this reason, MAs are often referred to as hybrid genetic algorithms or genetic local searches. The suggested EAMA is based on the two-stage approach. An initial population of solutions, each consisting of the same number of routes, is generated with a sophisticated route-minimization procedure developed by Nagata and Bräysy [22]. A subsequent procedure of the EAMA is then applied for minimizing total travel distance for the determined number of routes.

Main contributions of this paper are (i) the adaptation of the EAX to the VRPTW and (ii) the development of a novel penalty function for the time window violation. In the earlier research [20] the edge assembly crossover (EAX) [23–25], for the traveling salesman problem (TSP), was adapted to the CVRP. In this paper, we further adapt the EAX to the VRPTW by incorporating features to handle the time window constraint. In MAs, it is frequently very hard to generate feasible offspring solutions, i.e. solutions satisfying both the capacity and time window constraints, immediately after invoking the crossover and mutation operators. Therefore, temporarily allowing infeasible intermediate solutions during the search process in MAs makes sense. In fact, it is a well-known concept in metaheuristic search and often penalty functions are introduced to direct the search towards feasible solutions, see e.g. Gendreau et al. [26] and Toth and Vigo [27] for the literature on the CVRP and Badeau et al. [28], Berger and Barkoui [29], Bouthillier et al. [30], Ibaraki et al. [17,18] for the literature on the VRPTW. However, for n customers in the affected routes, it takes $O(n)$ time to compute the penalty of the time window violation defined in the previous works when traditional neighborhood operators are applied. In this paper, we suggest a novel penalty function, which can be computed in $O(1)$ time for most traditional neighborhood operators, to efficiently handle the time window violation as well as the capacity violation.

The new EAMA is tested on the benchmark problems of Solomon and of Gehring and Homberger. Computational results show that the proposed algorithm is robust and highly competitive. It improved 184 best-known solutions out of 356 benchmark instances in reasonable computation time. The remainder of this paper is arranged as follows. First, the problem definition and notations are described in Section 2. Then, the problem solving methodology is described starting with an overview of the EAMA followed by a detailed explanation of the main building blocks in Section 3. The computational analysis of the EAMA is presented in Section 4 and conclusions are provided in Section 5.

2. Problem definition and notations

The VRPTW is defined on a complete directed graph $G=(V,E)$ with a set of vertices $V=\{0,1,\ldots,N\}$ and a set of edges $E=\{(v,w)|v,w\in V\ (v\neq w)\}$. Node 0 represents the depot and the set of nodes $\{1,\ldots,N\}$ represents the customers. With each node $v\ (\in V)$ are associated a non-negative demand q_v (with $q_0=0$), non-negative service time s_v (with $s_0=0$) and a time window $[e_v,l_v]$. Each edge (v,w) has the non-negative travel distance d_{vw} and travel time c_{vw}. The capacity of the identical vehicles is given by Q.

Given a route r, let $\langle v_0,v_1,\ldots,v_n,v_{n+1}\rangle$ be a sequence of the customers in this route where v_0 and v_{n+1} represent the depot and n refers to the number of customers in this route. The route satisfies the capacity constraint if $\sum_{i=1}^{n}q_{v_i}\leq Q$. The earliest departure time at the depot, a_{v_0}, the earliest start time of service at a customer v_i, $a_{v_i}\ (i=1,\ldots,n)$, and the earliest arrival time to the depot, $a_{v_{n+1}}$, are defined recursively as follows:

$$a_{v_0}=e_0,$$
$$a_{v_i}=\max\{a_{v_{i-1}}+s_{v_{i-1}}+c_{v_{i-1}v_i},e_{v_i}\}\quad(i=1,\ldots,n+1). \tag{1}$$

The route satisfies the time window constraint if $a_{v_i}\leq l_{v_i}$ (i=0,…, n+1). A route is called feasible if both of the capacity and time window constraints are satisfied. The travel distance of the route is defined by $t(r)=\sum_{i=0}^{n}d_{v_iv_{i+1}}$.

A feasible solution σ is defined as a set of m feasible routes such that each customer is visited exactly once by a single route. The total travel distance of σ is defined as $F(\sigma)=\sum_{r=1}^{m}t(r)$. For most heuristic algorithms the objective of the VRPTW consists of finding a feasible solution σ that minimizes the number of routes m (primary objective) and, in case of ties, minimizes total distance traveled $F(\sigma)$ (secondary objective).

An edge (v,w) for which $e_v+s_v+c_{vw}>l_w$ is referred to as an (time window) infeasible edge in this paper.

3. Problem solving methodology

The Edge Assembly Memetic Algorithm (EAMA) developed in this paper is an extension of the EAMA proposed by Nagata [31] for the CVRP. The main feature of our approach is that the edge assembly crossover (EAX) operator [23–25] (described in Section 3.3) first generates offspring solutions which may violate the capacity and/or time window constraints. A subsequent local search-based repair procedure (described in Section 3.4.2) then tries to restore the feasibility of the temporarily infeasible solutions where the solutions are improved with a generalized cost function (described in Section 3.2) consisting of the total travel distance and penalty functions imposed on the constraint violations. Moreover, a simple local search (described in Section 3.4.3) attempts to further improve the feasible solutions found according to a standard MA procedure. The subsequent subsections discuss the main building blocks of the EAMA in detail, starting with an overview of the EAMA.

3.1. An overview of the EAMA

Fig. 1 gives an overview of the EAMA where N_{pop} and N_{ch} are the user-defined parameters. Our EAMA is based on a two-stage approach where the number of routes and the travel distance are independently minimized. In the first stage (lines 1–4), the minimum number of route m is first determined by the route minimization (RM) heuristic for the VRPTW by Nagata and Bräysy [22] (line 1)

```
Procedure EAMA (N_pop,N_ch)
begin
// Route minimization phase
1 : m := DETERMINE_M();                                   // RMheuristic
2 : for i :=1to N_pop do
3 :   σ_i := GENERATE_INITIAL_SOLUTIONS (m); // RMheuristic
4 : end for
// Distance minimization phase
5 : repeat
6 :   Let r(i)∈{1,...,N_pop} be a random permutation;
7 :   for i:= 1 to N_pop do
8 :     p_A :=σ_r(i); p_B := σ_r(i+1); (Note: r(N_pop+1) = r(1))
9 :     σ_best :=p_A;
10:     for j: = 1 to N_ch do
11:       σ := EAX (p_A,p_B);                             // crossover
12:       σ := Repair (σ);                                // repairprocedure
13:       σ := Local_Search(σ);                           // localsearch
14:       if F(σ) < F(σ_best ) then σ_best :=σ;
15:     end for
16:     σ_r(i) := σ_best ;
17:   end for
18: until termination condition is satisfied
19: return the best individual in the population;
end
```

Fig. 1. The penalty-based memetic algorithm.

described in Section 3.5. Here, m is minimized by executing the RM heuristic within a given time period. Then, a population of N_{pop} feasible solutions, each consisting of m routes, are created by the RM heuristic (line 3). During the second stage, the main part of the EAMA is used to minimize the total travel distance.

The main loop of the EAMA (lines 5–18) minimizes total travel distance. The procedures on lines 6–17 correspond to a single generation. For each generation, each individual solution is selected once both as parent p_A and as parent p_B (line 8) in random order determined at the beginning of this generation (line 6). For each pair of parents, p_A and p_B, the EAX crossover operator generates N_{ch} offspring solutions (line 11). If offspring solutions violate the capacity and/or time window constraints, the repair procedure is invoked to try to repair these constraint violations (line 12). If feasible solutions are obtained, they are further optimized to reduce the total travel distance by means of local search (line 13). If the best feasible offspring has a smaller total distance than p_A, it replaces the individual selected as p_A (lines 9, 14 and 16) rather than the poorest or one of the poorest solutions in the population. This replacement strategy is motivated by the fact that offspring solutions generated by the EAX followed by the repair and local search procedures tend to be similar to p_A and that we want to maintain diversity in the population.

We use two types of the EAX crossover based on *single* and *block* strategies (see Section 3.3). For each run of the EAMA, the single strategy is used until the best solution in the population cannot be improved for successive g_{max} generations. After that, only the block strategy is used until the best solution in the population cannot be improved for successive g_{max} generations. After that, the run is terminated (line 18).

3.2. Generalized cost function

To develop the EAMA for the VRPTW, possible violations of both capacity and time window constraints need to be addressed. Let σ be an infeasible solution that violates the capacity and/or time window constraints. The generalized cost function of solution σ, $F_g(\sigma)$, is defined in Eq. (2); it consists of the total travel distance $F(\sigma)$ and the penalty terms $P_c(\sigma)$ and $P_{tw}(\sigma)$ for the violations of the capacity and time window constraint, multiplied by the penalty coefficients α and β, respectively

$$F_g(\sigma) = F(\sigma) + \alpha \cdot P_c(\sigma) + \beta \cdot P_{tw}(\sigma). \tag{2}$$

$P_c(\sigma)$ is straightforwardly defined in Eq. (3) as the sum of total demand excess in all routes [26,27]. For each route r, $cust(r)$ refers to a set of customers in the route. The change in $P_c(\sigma)$ due to a local search move can be computed in $O(1)$ time when the traditional neighborhood operators such as 2-opt*, Or-Exchange, Relocation, and Exchange are applied [34]

$$P_c(\sigma) = \sum_{r=1}^{m} \max\left\{ \sum_{v \in cust(r)} q_v - Q, 0 \right\}. \tag{3}$$

As for the $P_{tw}(\sigma)$ penalty term, variants of the time window penalty for the VRP with soft time windows are employed in [28–30,17,18]. Given a route $\langle v_0, v_1, \ldots, v_n, v_{n+1} \rangle$, the penalty for the soft time windows in this route is defined as $\sum_{i=0}^{n+1} \max\{a_{v_i} - l_{v_i}, 0\}$ (see Eq. (1)). Fig. 2(a) illustrates the time schedule of a route defined by Eq. (1) where the sum of the lengths of the left arrows corresponds to the time window penalty of this route. The sum of the penalties in all routes defines the time window penalty of a solution. However, it takes $O(n)$ time to compute the change in the penalty when the traditional neighborhood operators are applied. In this paper, we suggest a new definition for the time window penalty structure illustrated in Fig. 2(b) whose change can be computed in $O(1)$ time for most traditional neighborhood operators. Moreover, this penalty measures the amount of the time window violation more appropriately as described below.

Given a route r, the extended earliest departure time at the depot, $\tilde{a}_{v_0}$, the extended earliest start time of service at a customer v_i, $\tilde{a}_{v_i}$ ($i = 1, \ldots, n$), and the extended earliest arrival time to the depot, $\tilde{a}_{v_{n+1}}$, are defined recursively in Eq. (4), and the suggested time window penalty of the route, denoted as $TW(r)$, is defined in Eq. (5)

$$\begin{aligned} &\tilde{a}_{v_0} = e_0, (\tilde{a}'_{v_0} = e_0), \\ &\tilde{a}'_{v_i} = \tilde{a}_{v_{i-1}} + s_{v_{i-1}} + c_{v_{i-1}v_i} \quad (i = 1, \ldots, n+1), \\ &\begin{cases} \tilde{a}_{v_i} = \max\{\tilde{a}'_{v_i}, e_{v_i}\} & \text{if } \tilde{a}'_{v_i} \le l_{v_i} \\ \tilde{a}_{v_i} = l_{v_i} & \text{if } \tilde{a}'_{v_i} > l_{v_i} \end{cases} \quad (i = 1, \ldots, n+1). \end{aligned} \tag{4}$$

$$TW(r) = \sum_{i=0}^{n+1} \max\{\tilde{a}'_{v_i} - l_{v_i}, 0\}. \tag{5}$$

Note that $\tilde{a}_{v_i}$ is equal to a_{v_i} (see Eq. (1)) if the route is feasible with respect to the time window constraint. For $i = 1, \ldots, n+1$, $\tilde{a}'_{v_i}$ refers to the extended earliest arrival time of the vehicle at node v_i and the time window constraint is violated at node v_i in case $l_{v_i} < \tilde{a}'_{v_i}$. In this case, we assume that the vehicle can travel back in time (by some sort of time machine) to l_{v_i} to start the service of customer v_i (arrive at the depot v_{n+1} if $i = n+1$) without delay, but at the expense of paying a penalty ($\tilde{a}'_{v_i} - l_{v_i}$). Therefore, in this case, $\tilde{a}_{v_i}$ is set to l_{v_i}. The total time window violation in the route r, $TW(r)$, is defined by the sum of the penalties that the vehicle must pay in the route to service all customers and to arrive at the depot without delay. Fig. 2(b) illustrates the time schedule of a route defined by Eq. (4) where the left arrows represent the penalties. Finally, $P_{tw}(\sigma)$ is defined as $P_{tw}(\sigma) = \sum_{r=1}^{m} TW(r)$.

Consider the partial path $\langle v_2, v_3, v_4 \rangle$ in Fig. 2. This partial path can be indeed considered to be time window feasible because the services of customers v_3 and v_4 are not delayed if the vehicle arrives at customer v_2 no later than time l_{v_2}. Contrary to previous penalty structure, our novel penalty structure does not impose penalties on this path after penalizing the time window violation at customer v_2.

As such, the new definition of the time window violation has two advantages: (i) a change in the $P_{tw}(\sigma)$ can be computed in $O(1)$ time for most traditional neighborhood operators (see Proposition 1 in

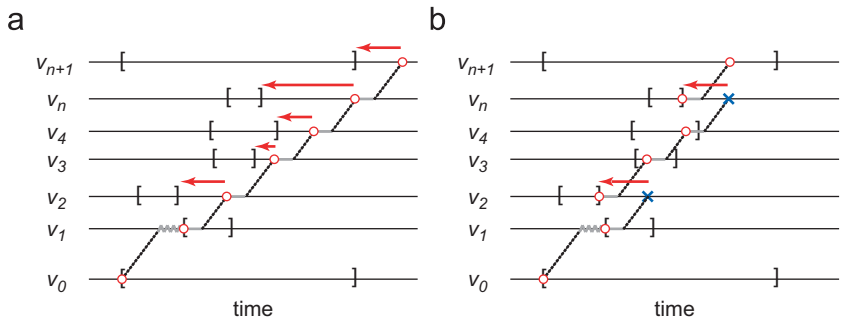

Fig. 2. Time tables of a route computed by (a) Eq. (1) and (b) Eq. (4). The brackets represent the time windows. The while circles represent (a) a_v and (b) $\tilde{a}_v$. The time window penalties are represented by the arrows.

the Appendix), and (ii) it appropriately evaluates good partial paths with respect to the time window constraint.

3.3. Edge assembly crossover

The edge assembly crossover (EAX) was originally proposed for the traveling salesman problem by Nagata and Kobayashi [23–25] and was further adapted to the CVRP by Nagata [20]. Here, the EAX is adapted to the VRPTW by incorporating features to deal with the time window constraint. More precisely, we define the EAX on the directed graph whereas the EAX for the CVRP is defined on the undirected graph. The EAX crossover operator, denoted by EAX(p_A, p_B), consists of five steps as illustrated in Fig. 3. In the procedure, p_A and p_B refer to parent solutions, both having the same number of routes m. If more than two offspring solutions are generated, Steps 3–5 are repeated.

In *Step* 1, a graph G_{AB} is defined by $G_{AB}=(V, E_A \cup E_B \backslash E_A \cap E_B)$ where E_A and E_B are defined as sets of edges consisting of parents p_A and p_B, respectively.

In *Step* 2, all edges are divided into so-called *AB-cycles* on G_{AB}. An *AB*-cycle is defined as a cycle on G_{AB} such that edges from p_A and p_B are linked alternately in the opposite orientation. AB-cycles are formed by randomly selecting starting node on G_{AB} and tracing in turn edges belonging to p_A in the forward direction and p_B in the reverse direction until an AB-cycle is found on the traced path. During the tracing process, the edge to be traced next is uniquely determined in customer nodes and is randomly selected in the depot among the several edges adjacent to the depot. Each time an *AB*-cycle is found, edges included in the *AB*-cycle are eliminated from G_{AB}. The generation of *AB*-cycles is continued until all edges in G_{AB} are eliminated. As an example, consider Fig. 3 in which all edges are divided into seven AB-cycles.

In *Step* 3, so-called *E-sets* are constructed by combining AB-cycles. For the selection of the *AB*-cycles, we employ the following two strategies.

> *Single strategy*: A single *AB*-cycle is randomly selected and defined as the *E*-set (e.g., *E*-set (a) in Fig. 3).
> *Block strategy*: First, a single *AB*-cycle is randomly selected as a *center AB-cycle*. In addition, the *AB*-cycles that share at least one customer node with the center *AB*-cycle and contain less nodes than the center *AB*-cycle are selected (e.g., *E*-set (b) is constructed by selecting *AB*-cycle 2 as the center *AB*-cycle).

By their nature (see Step 4), single strategy is more suited for relatively local improvement and block strategy tends to better support global improvement as more (geographically close) edges are considered.

In *Step* 4, p_A is selected as a base solution. Given an *E*-set, an intermediate solution is formed from the base solution by removing *E-set*$\cap E_A$ and adding *E-set*$\cap E_B$. As a result, each intermediate solution consists of m routes originating from the depot and possibly one or more subtours (cycles not including the depot).

In *Step* 5, possible subtours are connected to existing routes, one at a time and in random order, using 2-opt* moves [35]. More precisely, for a randomly selected subtour, an edge to be removed is selected from both the subtour and one of the routes, and then two new edges are introduced to connect them so that all edges face in the same orientation in the new route. Here, all possible combinations for the edges are attempted and the move that minimizes the generalized cost function is executed. Note that the change in the generalized cost function is evaluated on the connected route (the generalized cost function is defined as only the travel distance in subtours). The subtour-connection procedure is continued until all subtours are connected.

The main difference of the suggested EAX for the VRPTW with the EAX for the CVRP by [20] is that the orientation of the edges is taken into account. Considering the definition of the *AB*-cycle (step 2) and the *E*-set (step 3), the generation of intermediate solutions (step 4) is constrained by requiring that both of the indegree (the number of incoming edges) and the outdegree (the number of outgoing edges) are one in the customer nodes and m for the depot. Therefore, the parents' directed edges are combined to generate intermediate solutions such that the inherited edges have the same orientation in each route or subtour. On the other hand, the EAX for the CVRP generates intermediate solutions without considering the orientation of edges (i.e., *AB*-cycles are formed without considering the orientation of edges). Therefore, a route or subtour possibly consists of directed edges with different orientation in intermediate solutions as illustrated in Fig. 4. In the sequel, the suggested EAX and the EAX for the CVRP are referred to as asymmetric EAX (A-EAX) and symmetric EAX (S-EAX), respectively. This feature of the A-EAX is required to define the orientation of the routes in the offspring solutions because the VRPTW is asymmetric with respect to the orientation. Although the S-EAX can be applied to the VRPTW by defining the orientation of the routes (e.g., for each route, orientation can be determined by a majority vote on the edges' orientation in the route), this feature is still important as maintaining the order of customers in a route tends to reduce time window violations in the offspring solutions. Nevertheless, the offspring solutions still often violate the time window constraint as well as the capacity constraint. These remaining constraint violations are eliminated by the repair procedure described in the next subsection.

3.4. Local improvements

The goal of the repair procedure is to eliminate the capacity and time window violations in the offspring solutions

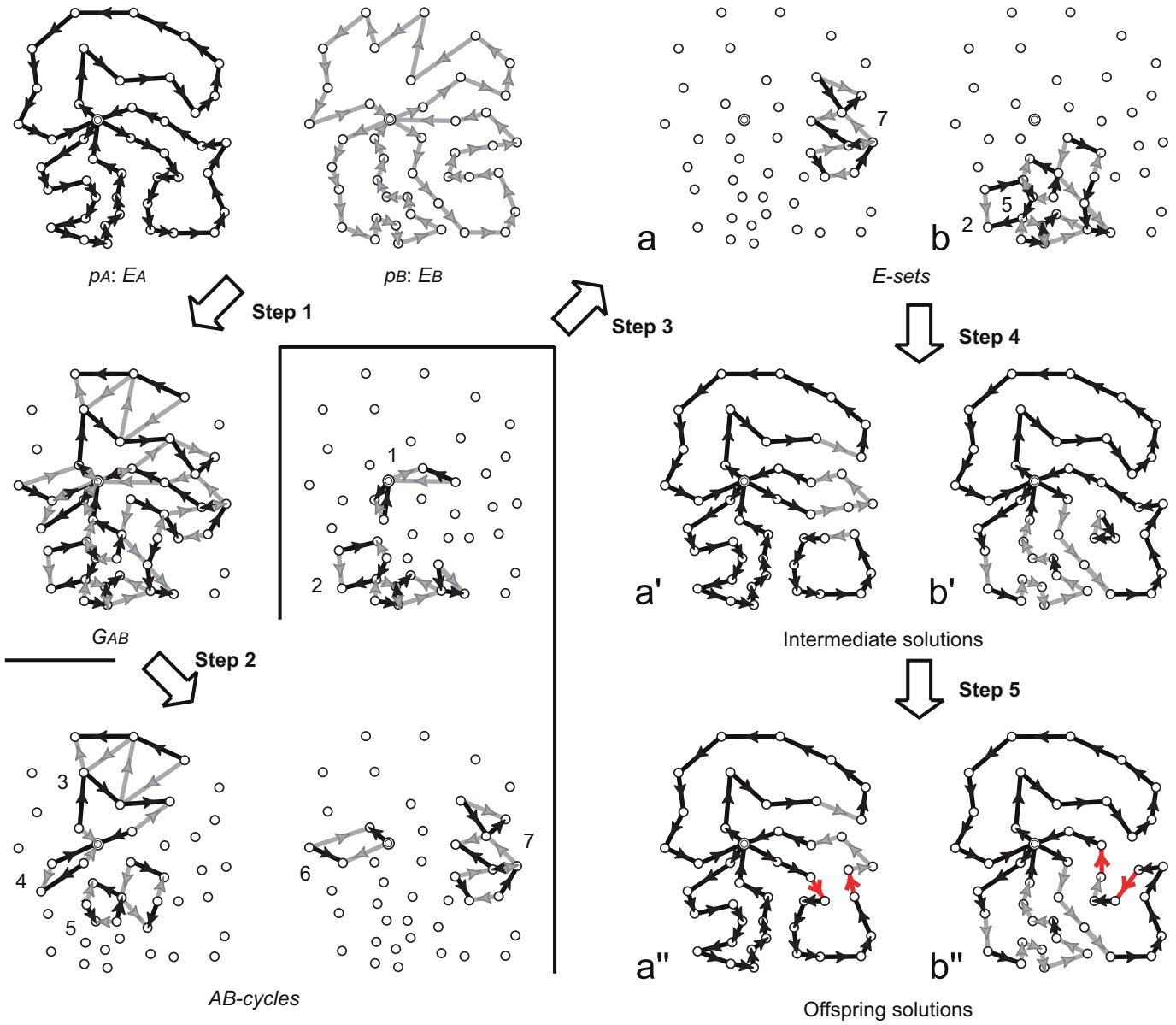

Fig. 3. Illustration of the EAX crossover steps for the VRPTW (A-EAX).

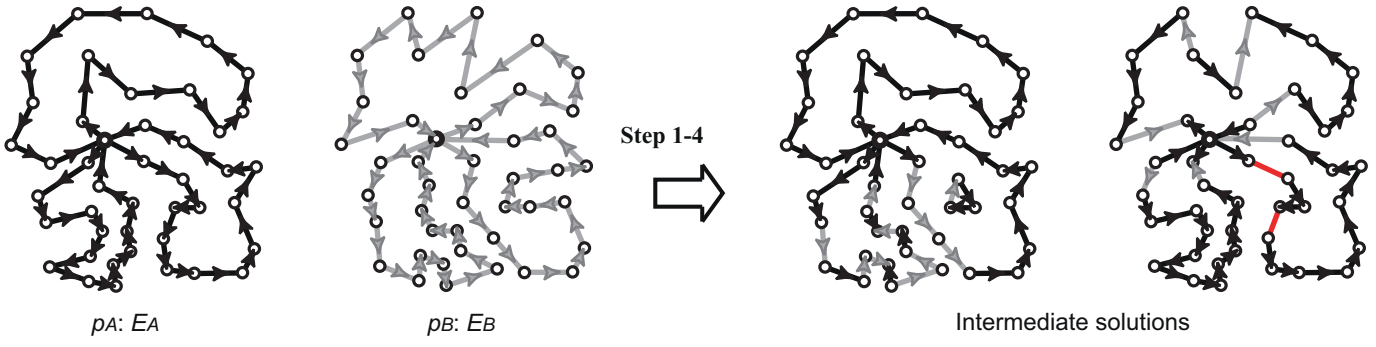

Fig. 4. Illustration of the EAX crossover for the CVRP (S-EAX).

generated by the A-EAX operator. If feasible solutions are obtained, they are further improved by the local search. The repair and local search procedures are two different hill-climbing methods based on traditional neighborhoods for the VRP. We first define the neighborhood used and then describe the details of these procedures.

Procedure REPAIR (σ)
begin
1 : **repeat**
2 : Randomly select an infeasible route r;
3 : Search $\sigma' \in \bigcup_{v \in cust(r)} N(v,\sigma)$ that minimizes $F_g(\sigma')$ subject to $\alpha \cdot \Delta P_c + \beta \cdot \Delta P_{tw} < 0$;
4 : **if** σ' exists **then** Update $\sigma := \sigma'$;
5 : **else break ;**
6 : **until** σ becomes an feasible solution
7 : **return** σ;
end

Fig. 5. Algorithm of the repair procedure.

3.4.1. Neighborhoods

The neighborhoods are based on four well-known edge-exchange operators: 2-opt* [35], Or-Exchange, Relocation, and Exchange [34], which are redefined in a different way. We define a subneighborhood, denoted by $\mathcal{N}(v,\sigma)$, as a set of moves around customer v for the current solution σ. More precisely, $\mathcal{N}(v,\sigma)$ is defined as a composite of the four neighborhoods listed below. In the definition w denotes a customer node that is among the N_{near} customers closest to v where N_{near} is a user-defined parameter. Here, w may or may not belong to the same route as v (v and w must belong to different routes in 2-opt* neighborhood). v^- (w^-) and v^+ (w^+) denote a predecessor and a successor of v (w), which may be the depot or a customer. Note that these neighborhoods are defined in such a way that an edge (v,w) is generated after each move.

2-opt*(v,σ):
- Remove (v,v^-) and (w,w^+), and add (v,w) and (v^-,w^+).
- Remove (v,v^+) and (w,w^-), and add (v,w) and (v^+,w^-).

Out-Relocate(v,σ):
- Insert v between w and w^-, and link v^- and v^+.
- Insert v between w and w^+, and link v^- and v^+.

In-Relocate(v,σ):
- Insert w between v and v^-, and link w^- and w^+.
- Insert w between v and v^+, and link w^- and w^+.

Exchange(v,σ):
- Insert v between w and $(w^-)^-$, and insert w^- between v^- and v^+.
- Insert v between w and $(w^+)^+$, and insert w^+ between v^- and v^+.

To limit the size of the subneighborhoods, the parameter N_{near} is set to 50 and the moves that introduce any infeasible edge are forbidden. Note also that In-Relocate neighborhood is not used in the repair procedure.

3.4.2. Repair procedure

Fig. 5 describes the algorithm of the repair procedure. Each iteration (lines 2–5) begins by randomly selecting an infeasible route r violating the capacity and/or time window constraints (line 2). Then, the current solution σ is improved by the move from the set of the subneighborhoods of the selected infeasible route r ($\bigcup_{v \in cust(r)} \mathcal{N}(v,\sigma)$) that minimizes the generalized cost function (line 3). Here, only moves that decreases the penalty term $(\alpha \cdot \Delta P_c(\sigma) + \beta \cdot \Delta P_{tw}(\sigma) < 0)$ are accepted because the purpose of the repair procedure is to restore the feasibility of a current solution. Without this additional condition, the feasibility will not be restored unless α and β are large enough (but this will increase the travel distance after the repair procedure). The procedure is repeated until all routes are feasible (line 6) or until no improving move is found for the selected infeasible route (line 5) (in this case, the infeasible solution is discarded). Here, one should note that In-Relocate neighborhood is not used in the repair procedure because any move from this neighborhood never decrease both of the penalty terms of the selected infeasible route.

Procedure Local_Search (σ)
begin
1 : V_{new} is initialized with a set of customers in the new routes not existing in p_A;
2 : **repeat**
3 : Randomly select $\sigma' \in \bigcup_{v \in V_{new}} N(v,\sigma)$ such that $F(\sigma') < F(\sigma)$;
4 : **if** σ' exists **then** Update $\sigma := \sigma'$;
5 : **until** no improvement move is found
6 : **return** σ ;
end

Fig. 6. Algorithm of the local search.

The change in the generalized cost $F_g(\sigma)$ (i.e., $F_g(\sigma') - F_g(\sigma)$) by a move from the subneighborhoods can be computed in $O(1)$ time with a few exceptions. In fact, the changes in the total distance $F(\sigma)$ and the penalty term $F_c(\sigma)$ can be easily computed in $O(1)$ time [34]. Moreover, the change in the penalty term $P_{tw}(\sigma)$ can also be computed in $O(1)$ time for most moves in the subneighborhoods. An efficient calculation method is described in the Appendix (Proposition 1).

In the repair procedure, the best-improvement strategy is used (lines 3, 4) to reduce the number of iterations and to avoid changing the solution structure inherited from the parents too much. However, a large amount of computation time will be spent to test all moves in a set of subneighborhoods $\bigcup_{v \in cust(r)} \mathcal{N}(v,\sigma)$ (line 3) if the number of customers in route r is large. Therefore, we suggest an additional search limitation strategy to speed up the repair procedure. In line 2 of the algorithm, routes violating the time window constraint can be preferentially selected. When a time window infeasible route is selected, we ignore a subset of the subneighborhoods $\mathcal{N}(v,\sigma)$ ($v \in cust(r)$) if it does not include a move reducing the time window penalty originating from route r. Fortunately, for a given customer node $v \in cust(r)$, we can compute in $O(1)$ time whether subneighborhood $\mathcal{N}(v,\sigma)$ should be ignored or not even though it includes a number of moves. Further details on this repair limitation strategy are provided in the Appendix (Proposition 2) and its computational implications are examined in Section 4.4.

3.4.3. Local search

If feasible solutions are generated after the repair procedure, they are improved by the local search procedure. Fig. 6 depicts the algorithm of the local search. In the procedure, feasible solutions are evaluated on their total travel distance (only feasible solutions are accepted) and are improved with the first-acceptance strategy (lines 3, 4). To reduce the computation time, we employ the search limitation strategy (line 1) by Nagata [31] that has demonstrated significant improvement in the computation time for the CVRP [20].

As shown in the procedure, the moves are restricted to the subneighborhoods of a set of customers, denoted by V_{new}, which is initialized at the beginning of the local search (line 1). Here, V_{new} is defined as a set of customers in the new routes, created after the A-EAX and the repair procedure, that do not exist in parent p_A. This strategy is motivated by the fact that the new routes are created by perturbing routes in parent p_A (other routes are inherited from parent p_A without change) and there will be more chance to find improving moves around the new routes. Note that routes other than the new routes can also be changed because customer w (see Section 3.4.1) can be selected from these routes.

3.5. Generation of the initial population

We employ the route minimization (RM) heuristic for the VRPTW [22] to minimize the number of routes and then to create the initial population of the EAMA. The RM heuristic starts with an initial solution where each customer is served by a separate route. The route-elimination procedure then repeatedly attempts to reduce the total number of routes by one at a time. This procedure begins with randomly selecting a route from the current solution for removal. Its customers are then temporarily removed from the solution and are transferred to the so-called *ejection pool* (*EP*) [19]. Then, attempts are repeated to insert the customers from the EP in the remaining existing routes avoiding violations of the capacity and time window constraints. In case that there is no insertion position for the customer selected from the EP, other customers are ejected from one of the existing routes to create a feasible insertion position for the selected customer and the *EP* is updated (i.e., the ejected customers are added to the *EP*). The ejection of customers is directed by a concept reminiscent of the guided local search [33].

At the beginning of the EAMA (see Fig. 1), number of routes (m) is minimized and determined by executing the RM heuristic once until a certain termination condition is met (line 1). The RM heuristic is stopped if (i) m is equal to the obvious lower bound $\lceil \sum_{v=1}^{N} q_v/Q \rceil$, or (ii) the execution time reaches the time limit T_1 (e.g., $T_1 = 10$ min). In addition, the RM heuristic is terminated before the condition (i) or (ii) is met if (iii) the number of customers in the EP is greater than or equal to n_u (e.g., $n_u = 5$) at the execution time of $T_1'(< T_1)$. The last condition is used to terminate the run that is unlikely to decrease the number of routes any more. Once m is determined, each independent run of the RM heuristic for generating an initial population (lines 2–4) is terminated immediately when the number of routes is equal to m or the condition (ii) is met (termination condition (iii) is not used here).

To generate the initial population, RM runs are repeated until N_{pop} solutions, each consisting of m routes, are obtained or the overall CPU time on the procedure for generating the initial population, T_{total}, is researched. If due to the time limitation (T_{total}), the number of solutions created for computationally difficult routing problems is smaller than N_{pop}, the obtained solutions are copied to generate the set of N_{pop} initial solutions (this occurred rarely during our experiments). Note that the procedures on lines 2–4 in Fig. 1 are simplistically described.

4. Computational experiments

The proposed algorithm was implemented in C++ using double precision floating point number representation and was executed on an AMD Opteron 2.4 GHz (4 GB memory) computer. Several computational experiments have been conducted to analyze the performance of the proposed EAMA. In this section, we describe the benchmark problems, the parameter settings for the experiments and present the results along with a comparative analysis.

4.1. Benchmark problems

The proposed algorithm was tested on the well-known and widely used benchmarks of Solomon [10] and of Gehring and Homberger [11]. Solomon's benchmarks consist of 56 instances with 100 customers. The instances are divided in six groups: R1, R2, RC1, RC2, C1, and C2, each containing between 8 and 12 instances. The C1 and C2 classes have customers located in clusters and in the R1 and R2 classes the customers are at random positions. The RC1 and RC2 classes contain a mix of both random and clustered customers. The C2, R2 and RC2 classes have longer scheduling horizons and larger capacities than the C1, R1 and RC1 classes, meaning that each vehicle can service a larger number of customers in the former classes. Gehring and Homberger benchmarks are extension of the Solomon's instances, consisting of five sets of 200, 400, 600, 800 and 1000 customers, with 60 instances in each set, resulting in 300 instances. For each size, the 60 instances are divided into six groups with different features, similar to the Solomon's benchmarks. We collected the best-known solutions to these benchmark sets from earlier papers [12,15,18,16,19,22] and the Sintef website.[1]

4.2. Experimental settings

The parameters of the EAMA (see Section 3.1) were set as follows: $N_{pop} = 100$, $N_{ch} = 20$ and $g_{max} = 50$. The penalty coefficients for the generalized cost function (see Section 3.2) were set as follows: $\alpha = 1.0$ and $\beta = 1.0$ (a sensitivity analysis for the penalty coefficients is presented in Section 4.4). The parameter for the neighborhood (see Section 3.4.1) was set to $N_{near} = 50$. The EAMA with this configuration was applied to each instance once.

In general, the population size N_{pop} in MAs has a great impact on both the solution quality and the computation time. Therefore, we tested a second set of parameters for the EAMA with different population sizes: $N_{pop} = 200$ (100-customer), 100 (200-customer), 50 (400-customer), 34 (600-customer), 25 (800-customer), 20 (1000-customer). Here, the population size was decreased with increasing the number of customers, i.e. $N_{pop} = \lceil 20\,000/N \rceil$, to reduce the computation time. The EAMA with this configuration was applied to each instance five times.

For the route-minimization phase (see Section 3.5), the parameters were initialized as: $T_1 = 10$ min, $T_1' = N/10$ s, $T_{total} = NN_{pop}/400$ min and $n_u = 5$. These parameters were adjusted such that good results were obtained with reasonable computation time. We do not consider effects of these parameters in this paper because the route minimization phase is not the main stream of this paper.

4.3. Computational results

The results obtained for both configurations of the EAMA are presented in Table 1 (Solomon's benchmarks) and Tables 2–6 (Gehring and Homberger benchmarks). The suggested EAMA variants, denoted by EAMA(N_{pop}), are compared with a limited set of results of recent metaheuristics that have shown the best performance according to the literature. For Solomon's benchmarks, BVH (Bent and Hentenryck [14]), HG (Homberger and Gehring [13]), RP (Pisinger and Ropke [15]), LZ (Lim and Zhang [19]) and GD (Prescott-Gagnon et al. [16]) are selected for comparison. For Gehring and Homberger benchmarks, MB (Mester and Bräysy [12]), RP (Pisinger and Ropke

[1] http://www.sintef.no/static/am/opti/projects/top/vrp/benchmarks.html. The results of R1_2_1, R1_4_1, R1_8_1 and C1_2_8 in Gehring and Homberger benchmarks are neglected because the reported solutions were infeasible resulting in an (artificial) lower number of routes. This was confirmed by the authors under consideration.

Table 1
The results for Solomon's 100-customer benchmarks.

100 customer	BVH (Best)	HG	PR (Best)	LZ	GD (Best)	EAMA(100)	EAMA(200)	
							Best	Ave.
R1	11.92	11.92	11.92	11.92	**11.92**	**11.92**	**11.92**	11.92
	1213.25	1212.73	1212.39	1213.61	**1210.34**	**1210.34**	**1210.34**	1210.34
R2	2.73	2.73	2.73	2.73	2.73	2.73	**2.73**	2.73
	966.37	955.03	957.72	961.05	955.74	952.08	**951.03**	951.71
C1	**10.0**	**10.0**	**10.0**	**10.0**	**10.0**	**10.0**	**10.0**	10.00
	828.38	**828.38**	**828.38**	**828.38**	**828.38**	**828.38**	**828.38**	828.38
C2	**3.0**	**3.0**	**3.0**	**3.0**	**3.0**	**3.0**	**3.0**	3.00
	589.86	**589.86**	**589.86**	**589.86**	**589.86**	**589.86**	**589.86**	589.86
RC1	11.5	11.5	11.5	11.5	**11.5**	11.5	**11.5**	11.50
	1384.22	1386.44	1385.78	1385.56	**1384.16**	1384.72	**1384.16**	1384.30
RC2	3.25	3.25	3.25	3.25	3.25	3.25	**3.25**	3.25
	1141.24	1123.17	1123.49	1121.82	1119.44	1119.45	**1119.24**	1119.43
CNV	405	405	405	405	405	405	**405**	405.0
CTD	57 567	57 309	57 332	57 368	57 240	57 205	**57 187**	57 197
Computer	SUN Ultra	P4-400M	P4-3.0G	P4-2.8G	OPT-2.3G	OPT-2.4G	OPT-2.4G	OPT-2.4G
CPU (min)	120×5	n/a	2.5×10	38.5	30×5	3.2	5.0×5	5.0
Runs	5	n/a	10	1	5	1	5	5

Table 2
The results for Gehring and Homberger 200-customer benchmarks.

200 customer	MB	PR (Best)	IINSUY	LZ	GD (Best)	EAMA(100)	EAMA(100)	
							Best	Ave.
R1	18.2[1]	18.2	18.2	18.2	18.2	18.2	**18.2**	18.20
	3618.68	3631.23	3665.77	3639.60	3615.69	3615.15	**3612.36**	3614.06
R2	4.0	4.0	4.0	4.0	4.0	4.0	**4.0**	4.00
	2942.92	2949.37	2965.64	2950.09	2937.67	2930.04	**2929.41**	2930.63
C1	18.8[1]	18.9	18.9	18.9	18.9	18.9	**18.9**	18.90
	2717.21	2721.52	2732.03	2726.11	2718.77	2718.44	**2718.41**	2718.44
C2	6.0	6.0	6.0	6.0	**6.0**	6.0	6.0	6.00
	1833.57	1832.95	1834.83	1834.24	**1831.59**	1831.64	1831.64	1831.73
RC1	18.0	18.0	18.0	18.0	18.0	18.0	**18.0**	18.00
	3221.34	3212.28	3287.61	3205.51	3192.56	3182.48	**3178.68**	3181.27
RC2	4.4	4.3	4.3	4.3	4.3	4.3	**4.3**	4.30
	2519.79	2556.87	2562.56	2574.10	2559.32	2536.54	**2536.22**	2536.46
CNV	694	694	694	694	694	694	**694**	694.0
CTD	168 573	169 042	170 484	169 296	168 556	168 143	**168 067**	168 126
Computer	P4-2.0G	P4-3.0G	P4-2.8G	P4-2.8G	OPT-2.3G	OPT-2.4G	OPT-2.4G	OPT-2.4G
CPU (min)	8	7.7×10	33.3	93.2	53×5	4.7	4.1×5	4.1
Runs	1	10	1	1	5	1	5	5

See footnote 1 in Section 4.1.

[15]), IINSUY (Ibaraki et. al. [18]), LZ (Lim and Zhang [19]) and GD (Prescott-Gagnon et al. [16]) are used as a benchmark to assess the performance of our methods.

The results are presented for each problem size. For each of the problem groups (R1, R2, C1, C2, RC1 and RC2) the average number of routes and the average total travel distance are reported. At the bottom of the table, the cumulative number of vehicles (CNV) and cumulative travel distance (CTD) is reported as well. The column labeled "Computer" provides specifications on the computer used in the experiments where "SUN Ultra", "P4" and "OPT" are abbreviation for SUN Ultra 10, Pentium 4 and Opteron, respectively. The row "CPU" shows average CPU time in minutes to solve each instance (total of the route minimization and distance minimization phases) and the number of independent runs is reported in the next row. If an algorithm is executed more than once for each instance, the best reported results over the number of runs are presented for the compared algorithms and the CPU time is equal to the average CPU time per single run multiplied by the number of runs. Both best and average results are presented for the EAMA configuration with five runs. The best results both in each class and in each problem size are represented in boldface. Here, the best results are determined according to, firstly, the cumulative (average) number of routes and, secondly, the cumulative (average) travel distance. One should note that some values of the average travel distances are smaller than those of the best results because the number of routes is usually reduced at the cost of the travel distance (e.g., known minimum travel distance of instance C2_4_3 in Gehring and Homberger benchmarks are 4109.88 ($m = 11$) and 3768.63 ($m = 12$)).

As for the Solomon's 100-customer benchmark set (see Table 1), all algorithms reach the best-known CNV (i.e., 405). The EAMA(100) configuration with single run dominates most competing algorithms with respect to both CTD and CPU time. PR with single run is faster than EAMA(100) and the CPU time of HG has not been reported. The five run configuration of EAMA(200) improved best-known so-

Table 3
The results for Gehring and Homberger 400-customer benchmarks.

400 customer	MB	PR (Best)	IINSUY	LZ	GD (Best)	EAMA(100)	EAMA(50)	
							Best	Ave.
R1	36.3[1]	36.4	36.4	36.4	36.4	36.4	**36.4**	36.40
	8530.03	8540.04	8746.94	8489.53	8420.52	8413.23	**8403.24**	8420.11
R2	8.0	8.0	8.0	8.0	8.0	8.0	**8.0**	8.00
	6209.94	6241.72	6269.90	6271.57	6213.48	6149.49	**6148.57**	6156.47
C1	37.9	37.6	37.7	37.6	37.6	37.6	**37.6**	37.60
	7148.27	7290.16	7282.15	7229.04	7182.75	7179.71	**7175.72**	7185.88
C2	12.0	12.0	12.0	11.7	11.9	**11.7**	11.7	11.70
	3840.85	3844.69	3851.96	3942.93	3874.58	**3898.02**	3899.00	3906.34
RC1	36.0	36.0	36.0	36.0	36.0	36.0	**36.0**	36.00
	8066.44	8069.30	8405.32	8005.25	7940.65	7931.66	**7922.23**	7948.69
RC2	8.8	8.5	8.6	8.5	8.6	**8.4**	8.4	8.48
	5243.06	5335.09	5337.50	5431.15	5269.09	**5293.74**	5297.86	5237.31
CNV	1389	1385	1387	1382	1385	1381	**1381**	1381.8
CTD	390 386	393 210	398 938	393 695	389 011	388 658	**388 466**	388 548
Computer	P4-2.0G	P4-3.0G	P4-2.8G	P4-2.8G	OPT-2.3G	OPT-2.4G	OPT-2.4G	OPT-2.4G
CPU (min)	17	15.8 × 5	66.6	295.9	89 × 5	34.0	16.2 × 5	16.2
Runs	1	5	1	1	5	1	5	5

Table 4
The results for Gehring and Homberger 600-customer benchmarks.

600 customer	MB	PR (Best)	IINSUY	LZ	GD (Best)	EAMA(100)	EAMA(34)	
							Best	Ave.
R1	54.5	54.5	54.5	54.5	54.5	54.5	**54.5**	54.50
	18 358.68	18 888.52	19 844.39	18 381.28	18 252.13	18 194.38	**18 186.24**	18 259.25
R2	11.0	11.0	11.0	11.0	11.0	**11.0**	11.0	11.00
	12 703.52	12 619.26	12 539.78	12 847.31	12 808.59	**12 319.75**	12 330.49	12 351.88
C1	57.8	57.5	57.5	57.4	57.4	**57.4**	57.4	57.40
	14 003.09	14 065.89	14 116.97	14 103.61	14 106.03	**14 054.70**	14 067.34	14 090.20
C2	17.8	17.5	17.4	17.4	17.5	**17.4**	17.4	17.40
	7455.83	7801.30	7945.56	7725.86	7632.37	**7601.94**	7605.07	7627.50
RC1	55.0	55.0	55.0	55.0	55.0	**55.0**	55.0	55.00
	16 418.63	16 594.94	17 278.81	16 274.17	16 266.14	**16 179.39**	16 183.95	16 241.51
RC2	12.1	11.6	11.6	11.5	11.7	11.4	**11.4**	11.54
	10 677.46	10 777.12	10 791.7	10 935.91	10 990.85	10 591.87	**10 586.14**	10 518.50
CNV	2082	2071	2070	2068	2071	**2067**	2067	2068.4
CTD	796 172	807 470	825 172	802 681	800 797	**789 420**	789 592	790 888
Computer	P4-2.0G	P4-3.0G	P4-2.8G	P4-2.8G	OPT-2.3G	OPT-2.4G	OPT-2.4G	OPT-2.4G
CPU (min)	40	18.3 × 5	100.0	646.86	105 × 5	80.4	25.3 × 5	25.3
Runs	1	5	1	1	5	1	5	5

See footnote 1 in Section 4.1.

lution to R211 problem instance. As for other instances, the results are equivalent to the best-known results in 45 instances, and are very close to the best-known in eight instances.[2] Table 7 lists the values of the best result obtained by the five EAMA(200) runs for Solomon's benchmarks where the new best-known solution is indicated in boldface.

As for the Gehring and Homberger 200–1000-customer benchmark sets (see Tables 2–6), both CNV and CTD of EAMA(100) are better than those of the compared algorithms for all problem sizes. Moreover, EAMA(100) obtained better average results than the previous algorithms in all classes except for class C2 in the 200-customer benchmark set. Taking into account differences in computers,[3] the average CPU time of EAMA(100) is almost the same as or less than those of the compared algorithms in the 200-customer benchmark set. However, the average CPU time of EAMA(100) increases faster with the problem size than PR, IINSUY and GD because these algorithms terminate the runs by a pre-defined CPU time limit or a pre-defined maximum number of iterations. To reduce the computation time of the EAMA, it makes more sense to decrease population size than to stop the EAMA(100) run before the termination condition is met. Therefore, EAMA($\lceil 20\,000/N \rceil$) with five runs is compared to the other algorithms (which we denote as EAMA(20 000/N) for simplicity). The average performance of EAMA(20 000/N) outperforms the previous algorithms with respect to both CNV and CTD in all problem sizes. Moreover, the average CPU times of EAMA(20 000/N) are lower than those of the compared algorithms in all problem sizes. The best results over five runs are better than the compared algorithms in 28 out of the 30 classes.

The experimental results show that the suggested EAMA is highly capable at minimizing the total travel distance, while minimizing the number of routes depends on the RM heuristic. In particular,

[2] The differences are at most 0.5 in the travel distance and these solutions can be considered to be equivalent to the best-known solutions because the reported eight solutions were computed using numbers rounded to two decimal digits.

[3] We applied our MA to some instances on a Pentium 2.8 GHz (512 MB memory) and observed that our computer is about 1.8 times faster than a Pentium 2.8 GHz machine irrespective of problem size.

Table 5
The results for Gehring and Homberger 800-customer benchmarks.

800 customer	MB	PR (Best)	IINSUY	LZ	GD (Best)	EAMA(100)	EAMA(25)	
							Best	Ave.
R1	72.8[1]	72.8	72.8	72.8	72.8	**72.8**	72.8	72.80
	31 918.47	32 316.79	33 275.72	31 755.57	31 797.42	**31 486.74**	31 492.81	31 588.96
R2	15.0	15.0	15.0	15.0	15.0	**15.0**	15.0	15.00
	20 295.28	20 353.51	20 209.92	20 601.22	20 651.81	**19 873.04**	19 914.97	19 952.14
C1	76.2	75.6	75.7	57.4	75.4	75.3	**75.2**	75.20
	25 132.27	25 193.13	25 487.55	25 026.42	25 093.38	24 990.42	**25 151.83**	25 252.07
C2	23.7	23.7	23.4	23.4	23.5	**23.4**	23.4	23.40
	11 352.29	11 725.46	11 860.90	11 598.81	11 569.39	**11 438.52**	11 447.27	11 472.25
RC1	73.0	73.0	72.4	72.0	72.0	**72.0**	72.0	72.00
	30 731.07	29 478.30	34 621.63	31 267.84	33 170.01	**31 020.22**	31 278.28	31 663.18
RC2	15.8	15.7	15.7	15.6	15.8	**15.4**	15.4	15.48
	16 729.18	16 761.95	1666.76	16 992.79	16 852.38	**16 438.90**	16 484.31	16 480.46
CNV	2765	2758	2750	2742	2745	2739	**2738**	2738.8
CTD	1 361 586	1 358 291	1 421 225	1 372 427	1 391 344	1 352 478	**1 357 695**	1 364 091
Computer	P4-2.0G	P4-3.0G	P4-2.8G	P4-2.8G	OPT-2.3G	OPT-2.4G	OPT-2.4G	OPT-2.4G
CPU (min)	145	22.7 × 5	133.3	1269.4	129 × 5	126.8	27.6 × 5	27.6
Runs	1	5	1	1	5	1	5	5

Table 6
The results for Gehring and Homberger 1000-customer benchmarks.

1000 customer	MB	PR (Best)	IINSUY	LZ	GD (Best)	EAMA(100)	EAMA(20)	
							Best	Ave.
R1	92.1	92.2	91.9	91.9	91.9	**91.9**	91.9	91.90
	49 281.48	50 751.25	53 366.10	48 827.23	49 702.32	**48 287.98**	48 369.71	48 533.61
R2	19.0	19.0	19.0	19.0	19.0	**19.0**	19.0	19.00
	29 860.32	29 780.82	29 546.19	30 164.60	30 495.26	**28 913.40**	29 003.42	29 089.69
C1	95.1	94.6	94.5	94.4	94.3	**94.1**	94.1	94.18
	41 569.67	41 877.00	42 459.35	41 699.32	41 783.27	**41 683.29**	41 748.60	41 812.96
C2	29.7	29.7	29.4	29.3	29.5	**29.1**	29.1	29.18
	16 639.54	16 840.37	16 986.46	16 589.74	16 657.06	**16 498.61**	16 534.36	16 529.38
RC1	90.0	90.0	90.0	90.0	90.0	**90.0**	90.0	90.00
	45 396.41	46 752.15	48 275.20	44 818.54	45 574.11	**44 743.18**	44 860.60	44 983.69
RC2	18.7	18.3	18.3	18.3	18.5	**18.3**	18.3	18.30
	25 063.51	25 090.88	24 904.08	25 064.88	25 470.33	**23 939.62**	24 055.31	24 141.11
CNV	3446	3438	3431	3429	3432	**3424**	3424	3425.60
CTD	2 078 110	2 110 925	2 155 347	2 071 643	2 096 823	**2 040 661**	2 045 720	2 050 904
Computer	P4-2.0G	P4-3.0G	P4-2.8G	P4-2.8G	OPT-2.3G	OPT-2.4G	OPT-2.4G	OPT-2.4G
CPU (min)	600	26.6 × 5	188.2	1865.37	162 × 5	186.4	35.3 × 5	35.3
Runs	1	5	1	1	5	1	5	5

See footnote 1 in Section 4.1.

Table 7
The best results for Solomon's benchmarks.

N	#	R1		R2		C1		C2		RC1		RC2	
		m	Distance	m	Distance	m	Distance	m	Distance	m	Distance	m	Distance
100	1	19	1650.80	4	1252.37	10	828.94	3	591.56	14	1696.95	4	1406.94
	2	17	1486.12	3	1191.70	10	828.94	3	591.56	12	1554.75	3	1365.65
	3	13	1292.68	3	939.50	10	828.06	3	591.17	11	1261.67	3	1049.62
	4	9	1007.31	2	825.52	10	824.78	3	590.60	10	1135.48	3	798.46
	5	14	1377.11	3	994.43	10	828.94	3	588.88	13	1629.44	4	1297.65
	6	12	1252.03	3	906.14	10	828.94	3	588.49	11	1424.73	3	1146.32
	7	10	1104.66	2	890.61	10	828.94	3	588.29	11	1230.48	3	1061.14
	8	9	960.88	2	726.82	10	828.94	3	588.32	10	1139.82	3	828.14
	9	11	1194.73	3	909.16	10	828.94						
	10	10	1118.84	3	939.37								
	11	10	1096.73	2	**885.71**								
	12	9	982.14										

EAMA(20 000/N) dominates existing state-of-the-art heuristic algorithms for the VRPTW. However, we can see that EAMA(100) with single run tends to outperform the best of five EAMA(20 000/N) runs with respect to the CTDs in the 600–1000 customers benchmark sets even if the CPU times of EAMA(100) are almost same as or less than the total CPU times of the five EAMA(20 000/N) runs.

Table 8
The best results for Gehring and Homberger benchmarks.

N	#	R1		R2		C1		C2		RC1		RC2	
		m	Distance	m	Distance	m	Distance	m	Distance	m	Distance	m	Distance
200	1	20	**4784.11**	4	4483.16	20	2704.57	6	1931.44	18	3618.05	6	3099.53
	2	18	4041.71	4	3621.20	18	2917.89	6	1863.16	18	**3249.34**	5	2825.24
	3	18	3381.96	4	2880.62	18	2707.35	6	**1775.08**	18	3008.33	4	**2601.87**
	4	18	**3057.81**	4	1981.30	18	2643.31	6	1703.43	18	**2851.68**	4	**2038.56**
	5	18	4107.86	4	3366.79	20	2702.05	6	1879.31	18	**3372.88**	4	2911.46
	6	18	3583.77	4	2913.03	20	2701.04	6	1857.35	18	3328.91	4	**2873.12**
	7	18	3150.11	4	2451.14	20	2701.04	6	1849.46	18	3189.32	4	2525.83
	8	18	**2951.99**	4	1849.87	19	2775.48	6	1820.53	18	3083.93	4	2295.97
	9	18	3762.57	4	**3092.04**	18	2687.83	6	1830.05	18	**3081.13**	4	2175.04
	10	18	3301.18	4	2654.97	18	2643.55	6	**1806.58**	18	**3002.97**	4	2015.61
400	1	40	**10 396.72**	8	**9210.15**	40	7152.06	12	4116.14	36	8749.98	11	**6682.37**
	2	36	8962.13	8	7606.75	36	7695.55	12	3930.05	36	**7925.66**	9	**6197.42**
	3	36	**7836.18**	8	**5911.50**	36	7069.24	11	**4018.74**	36	**7554.95**	8	**4930.84**
	4	36	**7301.61**	8	**4241.47**	36	**6803.41**	11	**3707.38**	36	**7323.08**	8	**3631.01**
	5	36	9294.34	8	**7132.14**	40	7152.06	12	3938.69	36	**8230.39**	8	**6711.72**
	6	36	8408.02	8	**6127.60**	40	7153.45	12	3875.94	36	**8195.99**	8	**5766.61**
	7	36	**7649.48**	8	**5028.33**	39	7421.56	12	3894.16	36	**7987.79**	8	**5336.58**
	8	36	**7293.21**	8	**4015.60**	37	7364.31	12	3792.76	36	**7801.11**	8	**4795.72**
	9	36	8758.61	8	**6404.83**	36	7076.02	12	3865.65	36	**7777.79**	8	**4551.80**
	10	36	8128.01	8	**5791.70**	36	6860.63	11	**3835.85**	36	**7625.34**	8	**4285.05**
600	1	59	21 426.76	11	**18 214.90**	60	14 103.87	18	7774.16	55	17 405.31	14	**13 360.73**
	2	54	**19 014.13**	11	**14 779.26**	56	14 163.31	17	**8290.76**	55	**16 091.11**	12	**11 555.76**
	3	54	**17 191.34**	11	**11 191.01**	56	**13 778.75**	17	**7528.76**	55	**15 322.93**	11	**9447.97**
	4	54	15 957.34	11	**8032.28**	56	**13 561.11**	17	**6910.62**	55	14 891.50	11	**7092.90**
	5	54	20 018.41	11	**15 107.97**	60	14 086.23	18	7575.20	55	**16 856.18**	11	**13 000.84**
	6	54	**18 211.84**	11	**12 503.26**	60	14 089.66	18	7479.04	55	**16 778.82**	11	**11 973.95**
	7	54	**16 793.65**	11	**10 147.52**	58	14 855.32	18	**7512.22**	55	**16 322.95**	11	**10 724.85**
	8	54	15 766.32	11	**7574.39**	56	14 559.16	17	**7601.94**	55	**16 136.38**	11	**10 007.82**
	9	54	19 153.71	11	**13 377.56**	56	**13 694.94**	17	**8036.01**	55	**15 996.82**	11	**9580.87**
	10	54	**18 047.99**	11	**12 265.83**	56	**13 637.34**	17	**7274.04**	55	**15 795.75**	11	**9069.41**
800	1	80	**36 852.06**	15	**28 125.99**	80	25 191.94	24	11 662.08	72	35 467.41	18	**21 018.41**
	2	72	**32 790.69**	15	**22 795.79**	72	**27 058.99**	23	12 376.66	72	**30 095.13**	16	**18 181.14**
	3	72	**29 708.38**	15	**17 714.05**	72	**24 362.72**	23	**11 411.53**	72	**28 520.62**	15	**14 442.40**
	4	72	28 078.94	15	**13 206.10**	72	**23 848.03**	23	**10 752.87**	72	**27 311.32**	15	**11 019.70**
	5	72	34 342.50	15	**24 345.47**	80	25 171.39	24	**11 425.23**	72	32 978.66	15	**19 137.47**
	6	72	**31 341.06**	15	**20 479.46**	80	25 169.13	24	11 353.61	72	33 181.53	15	**18 146.03**
	7	72	**29 195.72**	15	**16 677.31**	78	25 906.47	24	11 379.87	72	**30 768.36**	15	**16 843.96**
	8	72	**27 960.28**	15	**12 645.63**	74	25 401.38	23	**11 309.98**	72	**30 694.45**	15	**15 799.03**
	9	72	**32 791.14**	15	**22 335.51**	72	24 726.74	23	**11 721.31**	72	**30 299.38**	15	**15 361.73**
	10	72	**31 474.59**	15	**20 401.47**	72	**24 197.40**	23	**10 990.64**	72	**29 765.69**	15	**14 439.14**
1000	1	100	**53 501.39**	19	**42 294.31**	100	42 482.61	30	16 879.24	90	47 188.44	20	**30 289.39**
	2	91	**49 951.00**	19	**33 459.32**	90	43 023.01	29	**17 126.39**	90	**44 620.38**	19	25 425.65
	3	91	**45 707.48**	19	**24 938.95**	90	**40 339.01**	29	16 322.02	90	**42 786.69**	18	**20 043.04**
	4	91	**43 262.82**	19	**17 880.11**	90	**39 568.77**	28	**15 696.68**	90	**41 892.10**	18	**15 741.56**
	5	91	**53 289.48**	19	**36 258.34**	100	42 478.21	30	16 572.60	90	**46 078.09**	18	**27 140.77**
	6	91	**48 424.28**	19	**30 073.60**	100	42 472.31	29	**17 109.56**	90	45 932.63	18	**26 877.69**
	7	91	**44 975.94**	19	**23 253.89**	98	42 844.23	30	16 428.11	90	**45 405.13**	18	**25 161.74**
	8	91	42 960.69	19	**17 509.69**	93	42 436.10	29	16 191.70	90	44 628.29	18	**23 740.40**
	9	91	51 615.61	19	**33 068.74**	90	**40 983.55**	29	**16 372.09**	90	**44 634.17**	18	**23 065.64**
	10	91	**48 979.36**	19	**30 312.50**	90	**40 205.01**	28	**16 077.57**	90	**44 265.86**	18	**21 910.33**

These facts indicate that the population size of 34 (600-customer), 25 (800-customer) and 20 (1000-customer) are too small to support the full potential of the suggested EAMA. We can also see that the EAMA performs better on minimizing CTDs in R2 and RC2 classes than in the R1 and RC1 classes when being compared with other methods. This might be because class 2 instances are more similar to the TSP than the class 1 instances (i.e., one vehicle can service a larger number of customers in the class 2 instances) and that the EAX crossover past as a TSP operator allows it to exploit this structure better.

In all the experiments for 300 instances in Gehring and Homberger benchmarks, EAMA(100) with single run improved the best-known solutions in 170 instances and matched them in 35 instances. EAMA(20 000/N) with five runs improved the best-known solutions for 169 instances and matched them in 45 instances. As a results, we found new best-known solutions to 183 instances in these experiments. The best solutions for each instance obtained by these experiments are listed in Table 8 where the new best-known solutions are represented by the boldface.

The good performance of the EAMA partly depends on the RM heuristic because the RM heuristic is very powerful at minimizing the number of routes. For each problem size, the average CPU times in minutes for the route minimization phase (for determining m) in EAMA(100), for example, are 1.9 (0.8) (100-customer), 0.7 (0.5) (200-customer), 8.1 (0.5) (400-customer), 7.5 (0.7) (600-customer), 18.5 (0.6) (800-customer) and 24.2 (0.8) (1000-customer). As a result, the numbers of routes in five (two) instances are lower (larger) than those of the best-known solutions reported in the earlier papers

Table 9
Analysis of the different penalty coefficients (400-customer): CTD (left) and average CPU time in minutes (right).

α	β					α	β				
	0.01	0.1	1.0	10.0	100.0		0.01	0.1	1.0	10.0	100.0
0.01	389 535	389 444	*389 139*	389 435	*389 367*	0.01	32.1	20.8	17.3	15.5	16.9
0.1	389 725	*389 318*	*389 219*	*389 379*	*389 339*	0.1	26.4	16.6	15.4	16.4	15.8
1.0	389 719	*389 235*	***389 012***	*389 359*	*389 204*	1.0	21.3	14.1	**11.9**	13.1	13.7
10.0	390 691	390 291	389 844	389 955	389 883	10.0	21.8	14.8	12.8	12.8	13.2
100.0	391 525	390 642	390 615	390 358	390 566	100.0	22.2	16.8	13.4	14.4	12.9

except for [22] (the number of routes is larger than those in [22] in 15 instances).

4.4. Sensitivity analysis of the penalty coefficients

The generalized cost function is one of the key components of the EAMA. In the previous subsection, the results of the EAMA with the given penalty coefficient ($\alpha=1.0$ and $\beta=1.0$) are reported. Here, we present a sensitivity analysis for the penalty coefficients α and β. Using the values of (0.01, 0.1, 1.0, 10.0, and 100.0), 25 combinations of α and β are considered.

Table 9 shows the CTD (left) and the average CPU time in minutes for the distance minimization phase (right). The combination $(\alpha,\beta)=(1.0,1.0)$ offers the lowest CTD and average CPU time (indicated by the boldface). In the left table, the italic values indicate situations in which the excess of the CTD over the best one is less than 0.1% (given the results in Table 3, 0.1% is a sufficiently small value). Therefore, one can conclude that the EAMA reaches the good CTDs for a wide range of penalty values. In the right table, the average CPU time is significantly larger (slightly larger) than the minimum value when the penalty parameters are set too small (too large). The differences in the average CPU time result from the trade-offs in computational efforts during the repair and local search procedures. It is interesting to note that increasing penalty coefficients will decrease the number of iterations during the repair procedure but will also degrade solution quality measured by the total travel distance in the resulting feasible solution. Thus, it will increase the number of iterations in the subsequent local search.

Because of the robust performance of the EAMA for the various values of the penalty coefficients (α,β), we fixed them at the roughly estimated best level ($\alpha=1.0$ and $\beta=1.0$) for 400-customer problems for all problem instances.

5. Conclusion

In this paper, we have presented a memetic algorithm for the vehicle routing problem with time windows (VRPTW). We have extended the Edge Assembly Memetic Algorithm (EAMA) that was developed for the Capacitated Vehicle Routing Problem (CVRP). In particular, the edge assembly crossover (EAX) is adapted to the VRPTW (referred to as A-EAX) by taking the time window constraint into consideration. Moreover, we develop a local search based repair procedure to restore the feasibility of the offspring solutions generated by A-EAX using a novel penalty function to handle the time window and/or capacity violation in an efficient way. The A-EAX, repair procedure and simple local search procedures are combined in accordance with the standard MA framework.

We have demonstrated that the proposed EAMA performs very well at minimizing the travel distance for the well-known 56 Solomon's and 300 Gehring and Homberger benchmark problems, improving the best-known solutions to 184 instances in reasonable computation time. Given that our results are obtained by a fairly basic framework of the MA, the components of the proposed EAMA, such as the A-EAX crossover and the generalized penalty function, appear to have potential to be embedded in other metaheuristic approaches for solving the VRPTW and its variants.

Appendix

This section describes how the time window penalty $P_{tw}(\sigma)$ can be computed efficiently. For a given route $r=\langle v_0,v_1,\ldots,v_n,v_{n+1}\rangle$, the time window penalty of the route, $TW(r)$, is defined in Eqs. (4) and (5) as described in Section 3.2. Here we additionally define the forward time window penalty slack, denoted by $TW^{\rightarrow}_{v_i}$, for each node v_i in Eq. (6). $TW^{\rightarrow}_{v_i}$ represents the partial time window penalty that the vehicle must pay on the way from the depot (v_0) to v_i. Note that after paying the penalty, the vehicle can start the service of customer v_i (arrive at the depot v_{n+1} if $i=n+1$) at time $\tilde{a}_{v_i}$

$$TW^{\rightarrow}_{v_i}=\sum_{j=0}^{i}\max\{\tilde{a}'_{v_j}-l_{v_j},0\}\quad(i=0,\ldots,n+1). \tag{6}$$

The extended latest departure time of the depot, $\tilde{z}_{v_0}$, the extended latest start time of the service at a customer v_i, $\tilde{z}_{v_i}$ $(n=1,\ldots,n)$, and the extended latest arrival time to the depot, $\tilde{z}_{v_{n+1}}$, are recursively defined in Eq. (7). In addition, we define in Eq. (8) the backward time window penalty slack, denoted by $TW^{\leftarrow}_{v_i}$, for each node v_i

$$\begin{aligned}&\tilde{z}_{v_{n+1}}=l_0\quad(\tilde{z}'_{v_{n+1}}=l_0),\\&\tilde{z}'_{v_i}=\tilde{z}_{v_{i+1}}-c_{v_iv_{i+1}}-s_{v_i}\quad(i=0,\ldots,n),\\&\begin{cases}\tilde{z}_{v_i}=\min\{\tilde{z}'_{v_i},l_{v_i}\} & \text{if } \tilde{z}'_{v_i}\geq e_{v_i}\\ \tilde{z}_{v_i}=e_{v_i} & \text{if } \tilde{z}'_{v_i}<e_{v_i}\end{cases}\quad(i=0,\ldots,n),\end{aligned} \tag{7}$$

$$TW^{\leftarrow}_{v_i}=\sum_{j=i}^{n+1}\max\{e_{v_j}-\tilde{z}'_{v_j},0\}\quad(i=0,\ldots,n+1). \tag{8}$$

Fig. 7 illustrates the definition of $\tilde{z}_{v_i}$ along with that of $\tilde{a}_{v_i}$. Here, $TW^{\leftarrow}_{v_i}$ represents the partial time window penalty that the vehicle must pay on the way from v_i to the depot (v_{n+1}) no matter how early the vehicle starts the service of customer v_i (departs from the depot v_0 if $i=0$) (i.e., $\tilde{a}_{v_i}=e_{v_i}$). If $\tilde{a}_{v_i}\leq\tilde{z}_{v_i}$, no additional penalty is required on the way from v_i to the depot. Otherwise, an additional penalty $(\tilde{a}_{v_i}-\tilde{z}_{v_i})$ is required. This fact is intuitively understood; the vehicle travels back in time from $\tilde{a}_{v_i}$ to $\tilde{z}_{v_i}$ at v_i by paying penalty $(\tilde{a}_{v_i}-\tilde{z}_{v_i})$. Eq. (9a) represents this relationship and can in fact be proven by induction (proof is omitted).

$$TW(r)=TW^{\rightarrow}_{v_i}+TW^{\leftarrow}_{v_i}+\max\{\tilde{a}_{v_i}-\tilde{z}_{v_i},0\}\quad(i=0,\ldots,n+1), \tag{9a}$$

$$TW(r)=TW^{\rightarrow}_{v_{i-1}}+TW^{\leftarrow}_{v_i}+\max\{\tilde{a}'_{v_i}-\tilde{z}_{v_i},0\}\quad(i=1,\ldots,n+1), \tag{9b}$$

$$TW(r)=TW^{\rightarrow}_{v_{i-1}}+TW^{\leftarrow}_{v_{i+1}}+\max\{\tilde{a}'_{v_i}-\tilde{z}'_{v_i},0\}\quad(i=1,\ldots,n). \tag{9c}$$

Eq. (9b) is derived from Eq. (9a) because $\max\{\tilde{a}'_{v_i}-\tilde{z}_{v_i},0\}=\max\{\tilde{a}'_{v_i}-l_{v_i},0\}+\max\{\tilde{a}_{v_i}-\tilde{z}_{v_i},0\}$ holds. Eq. (9c) is also derived from Eq. (9b) because $\max\{\tilde{a}'_{v_i}-\tilde{z}'_{v_i},0\}=\max\{e_{v_i}-\tilde{z}'_{v_i},0\}+\max\{\tilde{a}'_{v_i}-\tilde{z}_{v_i},0\}$ holds.

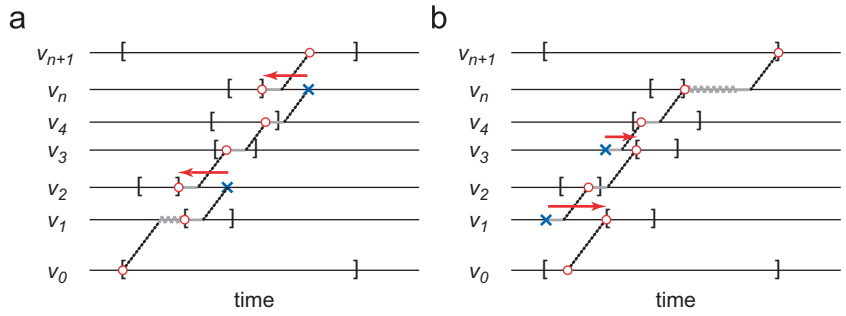

Fig. 7. The definitions of (a) $\tilde{a}_v$ and (b) $\tilde{z}_v$ (while circles). The penalties are represented by (a) the left arrows and (b) the right arrows.

Thus far, we defined $\tilde{a}_v$, $\tilde{z}_v$, $TW_v^{\rightarrow}$ and $TW_v^{\leftarrow}$ in a single route. Let these values be known in all routes in a current solution, meaning that $\tilde{a}_v$, $\tilde{z}_v$, $TW_v^{\rightarrow}$ and $TW_v^{\leftarrow}$ ($v \in V$) are known. As for the depot, let $\tilde{a}_0 = e_0$, $\tilde{z}_0 = l_0$, $TW_0^{\rightarrow} = 0$ and $TW_0^{\leftarrow} = 0$. Now, Proposition 1 is useful to compute the change in $P_{tw}(\sigma)$ in constant time and can be derived from Eqs. (9b) and (9c).

Proposition 1. *Let $\tilde{a}_v$, $\tilde{z}_v$, $TW_v^{\rightarrow}$ and $TW_v^{\leftarrow}$ ($v \in V$) be known for a current solutionσ.*

If a route $\langle 0,\ldots,x,v,\ldots,0\rangle$ is generated from two partial paths $\langle 0,\ldots,x\rangle$ and $\langle v,\ldots,0\rangle$, the time window penalty of this route is computed by

$$TW_x^{\rightarrow} + TW_v^{\leftarrow} + \max\{\tilde{a}_x + s_x + c_{xv} - \tilde{z}_v, 0\}.$$

If a route $\langle 0,\ldots,x,v,y,\ldots,0\rangle$ is generated from two partial paths $\langle 0,\ldots,x\rangle$, $\langle y,\ldots,0\rangle$ and customer v, the time window penalty of this route is computed by

$$TW_x^{\rightarrow} + TW_y^{\leftarrow} + \max\{\tilde{a}_x + s_x + c_{xv} - (\tilde{z}_y - c_{vy} - s_v), 0\}.$$

Therefore, the change in $P_{tw}(\sigma)$ can be computed in constant time when a randomly selected inter-route move from 2-opt*(v,σ), Out-Relocate(v,σ) and Exchange(v,σ) is tested. However, a randomly selected intra-route move from Out-Relocate(v,σ) and Exchange(v,σ)[4] (2-opt* is not defined in this situation) cannot be computed in constant time. In these cases, the worst case computation time is $O(n)$. Every time solution σ is updated, $\tilde{a}_v$, $\tilde{z}_v$, $TW_v^{\rightarrow}$ and $TW_v^{\leftarrow}$ ($v \in$ updated routes) must be recalculated and it takes $O(n)$ time. The actual computation times for these updates are negligible because the number of updates is much lower than the number of evaluations for the moves in the repair procedure.

Proposition 2 is used to limit the search in the repair procedure (the repair limitation strategy, see Section 3.4.2) and can be also derived from Eqs. (9b) and (9c).

Proposition 2. *For a given route $r = \langle 0,\ldots,v^-,v,v^+,\ldots,0\rangle$, if $TW_{v^-}^{\rightarrow} + TW_{v^+}^{\leftarrow} = TW(r)$, any move from $\mathcal{N}(v,\sigma)$ except for intra-route exchange does not decrease the time window penalty originating from this route.*

Proposition 2 is proven as follows. If the above condition holds and customer v is removed from route r, we can see that $TW(r)$ does not change. Thus, $TW(r)$ (and $P_{tw}(\sigma)$) is never decreased by a move from Out-Relocate(v,σ). In the same way, $TW(r)$ is never decreased by an inter-route move from Exchange(v,σ) (the time window penalty of the other route may be decreased). When two new route $\langle 0,\ldots,v^-,w^+,\ldots,0\rangle$ and $\langle 0,\ldots,w,v,\ldots,0\rangle$ are generated by a move from 2-opt*(v,σ), we can see that the two partial paths $\langle 0,\ldots,v^-\rangle$ and $\langle v,\ldots,0\rangle$, which originally form route r, bring down the penalties $TW_{v^-}^{\rightarrow}$ and $TW_v^{\leftarrow}$ in the new two routes. Here, $TW_v^{\leftarrow}$ is equal to $TW_{v^+}^{\leftarrow}$ if the assumption is met because, in general, $TW_{v^-}^{\rightarrow} + TW_v^{\leftarrow} \leq TW(r)$ and $TW_v^{\leftarrow} \geq TW_{v^+}^{\leftarrow}$ hold. Only intra-route moves from Exchange(v,σ) may decrease $TW(r)$ even if the assumption is met because w^+ (or w^-) is replaced with v in the same route.

When the Repair limitation strategy is used, v is temporary removed from $cust(r)$ (see Fig. 6) if $TW_{v^-}^{\rightarrow} + TW_{v^+}^{\leftarrow} = TW(r)$ is met. As described above, an intra-route move from *Exchange*(v,σ) may decrease $TW(r)$ even if the assumption is met because w^+ (or w^-) is replaced with v in the same route. Nevertheless we can still ignore $\mathcal{N}(v,\sigma)$ because such a move will be tested in $\mathcal{N}(w^+,\sigma)$ (or $\mathcal{N}(w^-,\sigma)$).

[4] Although we applied inter-route exchange moves, these are not usually used in the literature.

References

[1] Cordeau JF, Desaulniers G, Desrosiers J, Solomon MM, Soumis F. VRP with time windows. In: Toth P, Vigo D, editors. The vehicle routing problem. SIAM monographs on discrete mathematics and applications, vol. 9. Philadelphia, PA, 2002. p. 157–93.
[2] Bräysy O, Gendreau M. Vehicle routing problem with time windows, part I: route construction and local search algorithms. Transportation Science 2005;39(1):104–18.
[3] Bräysy O, Gendreau M. Vehicle routing problem with time windows, part II: metaheuristics. Transportation Science 2005;39(1):119–39.
[4] Golden B, Raghavan S, Wasil E, editors. The vehicle routing problem, latest advances and new challenges. In: Operations research/computer science interfaces series, vol. 43. Berlin: Springer; 2008.
[5] Jepsen M, Spoorendonk S, Petersen B, Pisinger D. A non-robust branch-and-cut-and-price algorithm for the vehicle routing problem with time windows. Technical Report no. 06/03 ISSN: 0107-8283. Department of Computer Science, University of Copenhagen; 2006.
[6] Jepsen M, Spoorendonk S, Petersen B, Pisinger D. Subset-row inequalities applied to the vehicle-routing problem with time windows. INFORMS Journal on Operations Research 2008;56(2):479–511.
[7] Chabrier A. Vehicle routing problem with elementary shortest path based column generation. Computers & Operations Research 2006;33(10):2972–90.
[8] Irnich S, Villeneuve D. The shortest path problem with resource constraints and k-cycle elimination. INFORMS Journal on Computing 2006;18(3):391–406.
[9] Kallehaugea B, Larsenb J, Madsena OBG. Lagrangian duality applied to the vehicle routing problem with time windows. Computers & Operations Research 2006;33(5):1464–87.
[10] Solomon MM. Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research 1987;35(2):254–65.
[11] Gehring H, Homberger J. A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In: Proceedings of EUROGEN99, 1999. p. 57–64.
[12] Mester D, Bräysy O. Active guided evolution strategies for large scale vehicle routing problems with time windows. Computers & Operations Research 2005;32(6):1593–614.
[13] Homberger J, Gehring H. Two-phase hybrid metaheuristic for the vehicle routing problem with time windows. European Journal of Operational Research 2005;162:220–38.
[14] Bent R, Hentenryck P. A two-stage hybrid local search for the vehicle routing problem with time windows. Transportation Science 2004;38(4):515–30.
[15] Pisinger D, Ropke S. A general heuristic for vehicle routing problems. Computers & Operations Research 2007;34(8):2403–35.
[16] Prescott-Gagnon E, Desaulniers G, Rousseau L-M. A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows. Working paper, University of Montreal, Canada; 2007.

[17] Ibaraki T, Kubo M, Masuda T, Uno T, Yagiura M. Effective local search algorithms for the vehicle routing problem with general time window constraints. Transportation Science 2005;39(2):206–32.
[18] Ibaraki T, Imahori S, Nonobe K, Sobue K, Uno T, Yagiura M. An iterated local search algorithm for the vehicle routing problem with convex time penalty functions. Discrete Applied Mathematics 2008; 156(11):2050–69.
[19] Lim A, Zhang X. A two-stage heuristic with ejection pools and generalized ejection chains for the vehicle routing problem with time windows. INFORMS Journal on Computing 2007;19(3):443–57.
[20] Nagata Y. Edge assembly crossover for the capacitated vehicle routing problem. In: Proceedings of the 7th European conference on evolutionary computation in combinatorial optimization, 2007. p. 124–53.
[21] Moscato P. On evolution, search, optimization, genetic algorithms and martial arts towards memetic algorithms. C3P Report 826, California Institute of Technology; 1989.
[22] Nagata Y, Bräysy O. A powerful route minimization heuristic for the vehicle routing problem with time windows, Operations Research Letters, in press.
[23] Nagata Y, Kobayashi S. Edge assembly crossover: a high-power genetic algorithm for the traveling salesman problem. In: Proceedings of the 7th international conference on genetic algorithms, 1997. p. 450–7.
[24] Nagata Y. Fast EAX algorithm considering population diversity for traveling salesman problems. In: Proceedings of the 6th international conference on EvoCOP2006, 2006. p. 171–82.
[25] Nagata Y. New EAX crossover for large TSP instances. In: Proceedings of the 9th international conference on parallel problem solving from nature, 2006. p. 372–81.
[26] Gendreau M, Hertz A, Laporte G. A tabu search heuristic for the vehicle routing problem. Management Science 1994;40(10):1276–90.
[27] Toth P, Vigo D. The granular tabu search and its application to the vehicle routing problem. INFORMS Journal on Computing 2003;15(4):333–48.
[28] Badeau P, Guertin F, Gendreau M, Potvin J-Y, Taillard ED. A parallel tabu search heuristic for the vehicle routing problem with time windows. Transportation Research C-5 1997;5(2):109–22.
[29] Berger J, Barkoui M. A parallel hybrid genetic algorithm for the vehicle routing problem with time windows. Computers & Operations Research 2004;31(12):2037–53.
[30] Bouthillier A, Crainic T. A cooperative parallel meta-heuristic for the vehicle routing problem with time windows. Computers & Operations Research 2005;32(7):1685–708.
[31] Nagata Y, Bräysy O. Efficient local search limitation strategies for vehicle routing problems. In: Proceedings of the 8th European conference on evolutionary computation in combinatorial optimization, 2008. p. 48–60.
[32] Homberger J, Gehring H. Two evolutionary meta-heuristics for the vehicle routing problem with time windows. INFORMS Journal on Computing 1999;37(3):297–318.
[33] Voudouris C, Tsang E. Guided local search. In: Glover F, editor. Handbook of metaheuristics. Dordrecht: Kluwer; 2003. p. 185–218.
[34] Kindervater G, Savelsbergh M. Vehicle routing: handling edge exchanges. In: Aarts E, Lenstra JK, editors. Local search in combinatorial optimization. New York: Wiley; 1997. p. 337–60.
[35] Potvin J-Y, Rousseau J-M. An exchange heuristic for routing problems with time windows. Journal of the Operational Research Society 1995;46:1433–46.