Computers & Operations Research ■ (■■■■) ■■■–■■■

Contents lists available at ScienceDirect

Computers & Operations Research

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

A hybrid variable neighborhood tabu search heuristic for the vehicle routing problem with multiple time windows

Slim Belhaiza [a,*], Pierre Hansen [b], Gilbert Laporte [c]

[a] Department of Mathematics and Statistics, KFUPM, Dhahran 31261, Saudi Arabia
[b] GERAD, HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montréal, Canada H3T 2A7
[c] CIRRELT and HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montréal, Canada H3T 2A7

ARTICLE INFO

Keywords:
Variable neighborhood search
Tabu search
Vehicle routing problem
Multiple time windows

ABSTRACT

This paper presents a new hybrid variable neighborhood-tabu search heuristic for the Vehicle Routing Problem with Multiple Time windows. It also proposes a minimum backward time slack algorithm applicable to a multiple time windows environment. This algorithm records the minimum waiting time and the minimum delay during route generation and adjusts the arrival and departure times backward. The implementation of the proposed heuristic is compared to an ant colony heuristic on benchmark instances involving multiple time windows. Computational results on newly generated instances are provided.

1. Introduction

The purpose of this paper is to present a new variable neighborhood-tabu search heuristic for the Vehicle Routing Problem with Multiple Time Windows (VRPMTW). The VRPMTW arises, for example, in the delivery operations of furniture and electronic retailers, where customers are sometimes offered a choice of delivery periods. The problem also occurs in long-haul transportation (see e.g., Rancourt et al. [15]). The VRPMTW has received relatively little attention in the operations research literature. To our knowledge, it has only been investigated by Favaretto et al. [7] who considered variants with single or multiple visits. These authors have proposed an ant colony heuristic for the problem. In contrast, several publications are available on the related Traveling Salesman Problem [12,13], the Team Orienteering Problem with Multiple Time Windows [21,19], and the Multi-Visits Multi-interdependent Time Windows VRP [6]. In this paper we describe a hybridized variable neighborhood-tabu search heuristic for the VRPMTW. The proposed heuristic embeds the concept of adaptive memory (Rochat and Taillard [16]), sometimes used in tabu search (see also Bozkaya et al. [3]), into the standard variable neighborhood search framework. The remainder of this paper is organized as follows. The VRPMTW is formulated in Section 2. The mathematical model is presented in Section 3. The heuristic is described in Section 4. The route minimization algorithm is presented in Section 5 followed by computational results in Section 6, and by conclusions in Section 7.

2. Problem description

The VRPMTW is defined on a directed graph $G=(V,A)$, where V is the vertex set and A is the arc set. The vertex set is partitioned into $V=\{N,D\}$, where $N=\{1,\ldots,n\}$ is a set of customers and D is a set of depots. We consider a set R of vehicles and we denote by m the number of vehicles. We denote by Q_k the capacity of vehicle k, where $k\in R$. Every customer $i\in N$ has a non-negative demand q_i, a non-negative service time s_i, a set $W_i=\{[l_i^p,u_i^p], p=1,\ldots,p_i\}$ of p_i time windows. The travel time associated with arc $(i,j)\in A$ is denoted by t_{ij}. We also denote by D_k, the maximum duration of the route of vehicle k. Even if some of these features, such as multi-depots and maximum duration, are exogenous to the VRPMTW context, we think that providing a larger view of the problem can be helpful.

The aim of the VRPMTW is to design a set of m vehicle routes of least total travel time or of least duration such that

(1) every customer is served once by one vehicle;
(2) the service of every customer starts within one of its time windows; if the vehicle arrives before the beginning of the time window, it must then wait;
(3) the total demand of a route does not exceed the vehicle capacity;
(4) the total duration of a route assigned to a vehicle k cannot exceed a preset upper bound D_k.

3. Mathematical programming formulation

The VRPMTW can be formulated as a 0–1 mixed integer linear program, along the lines of Favaretto et al. [7]. We define in Table 1

* Corresponding author. Tel.: +966 38601054.
E-mail addresses: slim.belhaiza@gerad.ca, slimb@kfupm.edu.sa (S. Belhaiza), pierre.hansen@gerad.ca (P. Hansen), gilbert.laporte@cirrelt.ca (G. Laporte).

0305-0548/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2013.08.010

Table 1
Definition of the variables and parameters.

Variable	Type	Description
x_{ij}^k	Binary	Equal to 1 if and only if arc (i, j) is traversed by vehicle k
y_{ij}^k	Real	Equal to the flow carried on arc (i, j)
r^k	Binary	Equal to 1 if and only if vehicle k is used
v_i^p	Binary	Equal to 1 if and only if customer i is served within its time window p
w_i^k	Real	Waiting time of vehicle k at customer i
z_i^k	Binary	Equal to 1 if and only if customer i is assigned to vehicle k
q_d^k	Real	Total demand loaded in vehicle k at depot d
d_k	Real	Route duration of vehicle k
a_i^k	Real	Arrival time of vehicle k at customer i
b_i^k	Real	Departure time of vehicle k from customer i
Parameter	**Type**	**Description**
t_{ij}	Real	Travel time associated with the arc (i, j)
q_i	Real	Demand associated with vertex customer i
l_i^p	Real	Time window p lower bound at customer i
u_i^p	Real	Time window p upper bound at customer i
Q_k	Real	Capacity of vehicle k
D_k	Real	Maximum duration of the route of vehicle k
s_i	Real	Service time at customer i
e_d^k	Binary	Equal to 1 if and only if vehicle k starts and ends at depot d
B	Binary	Equal to 1 if and only if total duration is to be minimized
F^k	Real	Fixed cost in time units of using vehicle k
M	Real	Arbitrary large constant

the variables and the parameters used in our formulation.

$$\text{minimize} \quad \sum_{k\in R}\sum_{i\neq j} t_{ij}x_{ij}^k + B\sum_{k\in R}\sum_{i\in N} w_i^k + \sum_{k\in R} F^k r^k$$

$$\text{subject to} \quad \sum_{k\in T_i} z_i^k = 1, \quad i\in V, \tag{1}$$

$$\sum_{j\in V} x_{ji}^k = \sum_{j\in V} x_{ij}^k, \quad i\in V \text{ and } k\in R, \tag{2}$$

$$2x_{ij}^k \le z_i^k + z_j^k, \quad i, j\in V \text{ and } k\in R, \tag{3}$$

$$\sum_{k\in T_i\cap T_j}\sum_{j\in V} x_{ij}^k \le 1, \quad i\in V, \tag{4}$$

$$\sum_{k\in T_i\cap T_j}\sum_{j\in V} x_{ji}^k \le 1, \quad i\in V, \tag{5}$$

$$y_{ij}^k \le Q_k x_{ij}^k, \quad i, j\in V \text{ and } k\in R, \tag{6}$$

$$\sum_{j\in V} y_{dj}^k - \sum_{i\in V} y_{id}^k = q_d^k e_d^k, \quad d\in D \text{ and } k\in R, \tag{7}$$

$$\sum_{j\in V} y_{ji}^k - \sum_{j\in V} y_{ij}^k \ge q_i z_i^k, \quad i\in N \text{ and } k\in R, \tag{8}$$

$$b_d^k \ge l_d - M(1-z_d^k), \quad d\in D \text{ and } k\in R, \tag{9}$$

$$a_d^k \le u_d + M(1-z_d^k), \quad d\in D \text{ and } k\in R, \tag{10}$$

$$a_d^k - b_d^k \le D_k + M(1-z_d^k), \quad d\in D \text{ and } k\in R, \tag{11}$$

$$b_i^k \ge a_i^k + w_i^k + s_i - M(1-z_i^k), \quad i\in N \text{ and } k\in R, \tag{12}$$

$$a_j^k \ge b_i^k + c_{ij} - M(1-x_{ij}^k), \quad i, j\in V \text{ and } k\in R, \tag{13}$$

$$a_j^k \le b_i^k + c_{ij} + M(1-x_{ij}^k), \quad i, j\in V \text{ and } k\in R, \tag{14}$$

$$a_i^k + w_i^k \ge l_i^p - M(1-z_i^k) - M(1-v_i^p), \quad i\in N,\ p\in W_i \text{ and } k\in R, \tag{15}$$

$$a_i^k + w_i^k \le u_i^p + M(1-z_i^k) + M(1-v_i^p), \quad i\in N,\ p\in W_i \text{ and } k\in R, \tag{16}$$

$$\sum_{p=1}^{p_i} v_i^p = 1, \quad i\in N\cap D, \tag{17}$$

$$r^k \ge z_i^k, \quad i\in V \text{ and } k\in R, \tag{18}$$

$$y_{ij}^k,\ w_i^k,\ q_d^k,\ d_k,\ a_i^k,\ b_i^k \ge 0, \tag{19}$$

$$r^k,\ x_{ij}^k,\ v_i^p,\ z_i^k \text{ binary}. \tag{20}$$

The objective is to minimize the total travel time, plus the total waiting time multiplied by the binary parameter B, plus the sum of the fixed costs of the vehicles used. In order to convert the real cost of using a vehicle to the same units used in the other parts of the objective, the fixed cost F is expressed in time units.

Constraints (1) state that each customer is assigned to exactly one vehicle. Constraints (2) state that each route of a vehicle k starts and ends at the depot, and the number of arcs leaving a customer i is equal to the number of arcs entering it. Constraints (3) mean that any arc (i, j) can be traversed by vehicle k only if z_i^k and z_j^k are both equal to 1. Constraints (4) and (5) force every customer i to be visited by one vehicle. Constraints (6) assert that the flow on arc (i, j) is upper-bounded by the capacity Q_k of the vehicle k traversing this arc. Constraints (7) ensure that the demand of the customers visited by vehicle k is satisfied, while constraints (8) mean that the demand of each customer assigned to the route of a vehicle k is satisfied. Constraints (9) ensure that the departure time of vehicle k from the depot d does not exceed l_d, and constraints (10) force the arrival time of vehicle k at the depot d to be less than or equal to u_d. Finally, constraints (11) mean that the total duration of each route cannot exceed the maximum duration D_k.

Constraints (12) ensure that the departure time from customer i is at least equal to the arrival time at customer i, plus the waiting time and the service time at customer i only if customer i is assigned to vehicle k. Similarly, constraints (13) and (14) mean that the arrival time at customer j is equal to the departure time from customer i, plus the cost t_{ij} of arc (i, j) only if this arc is assigned to

Please cite this article as: Belhaiza S, et al. A hybrid variable neighborhood tabu search heuristic for the vehicle routing problem with multiple time windows. Computers and Operations Research (2013), http://dx.doi.org/10.1016/j.cor.2013.08.010

vehicle k. Constraints (15) and (16) imply that the arrival time plus the waiting time of vehicle k at customer i is within the time window $[l_i^p, u_i^p]$ only if customer i is assigned to vehicle k and time window p is chosen. Constraints (17) mean that exactly one single time window is chosen for each customer i. Constraints (18) ensure that customer i is served by vehicle k only if this vehicle is used. Finally, constraints (19) and (20) state the feasibility intervals for the decision variables.

This linear program is large and can be solved to optimality, in a reasonable time, only for small instances. The use of heuristics is therefore necessary for larger instances.

4. Description of the variable neighborhood heuristic

Variable Neighborhood Search (VNS) was introduced by Mladenović and Hansen [10] as a generic local search methodology. It has since been successfully applied to a variety of contexts, including graph theory [2], packing problems [11] and location-routing [9]. The basic idea of VNS is to apply a systematic change of neighborhoods within a local search. In our implementation of VNS, we consider a number of different neighborhood structures instead of a single one, as is the case of many local search implementations. Neighborhood changes are applied during both a descent (improving) phase and an exploration phase to move out of local optima. In our implementation we allow infeasible solutions during the optimization process if these offer a good balance between feasibility and penalized costs.

Table 2 defines several control parameters used in our implementation. As for other penalties defined in this paper, these parameters are set to static values during the solution process. In Appendix B, we report the results of the parameters calibration and the best values found during our experiments. All time windows constraints, capacity constraints and maximum duration constraints can be violated if weighted penalties are added to the objective function. Since penalties weight could become very large, this is usually not a good idea. However, our computational results show that our implementation does not suffer from this.

We define $V_\varnothing$ as the set of vertices with violated time windows constraints. If $i \in V_\varnothing$, the penalty is multiplied by the minimum of the absolute values of the differences between the arrival time a_i at customer i and the bounds of each time window $p \in W_i$, raised to the power δ. For the capacity and duration constraints, the penalty multiplies the maximum between zero and the difference between the two sides of the constraint, raised to the power δ. The total cost of a solution s is then

$$f(s) = c(s) + \alpha \min_{p \in W_i,\ i \in V_\varnothing} \{|a_i - l_i^p|, |a_i - u_i^p|\}^\delta + \beta \sum_{k=1}^{m} \max\{0, (q_k - Q_k)\}^\delta + \chi \sum_{k \in R} \max\{0, (d_k - D_k)\}^\delta,$$

where $d_k = a^k - b^k$ is the route duration of vehicle k.

Our VNS implementation consists of four main phases: (i) the construction of an initial solution; (ii) a shaking phase with the choice of the neighborhood structure and exchange operators; (iii) a local search phase with different improving operators, and (iv) a move-or-not-move phase.

4.1. Construction of an initial solution

During the initialization phase, the best customer to assign to a vehicle is the one that induces the least increase in total route duration. Since this could generate a set of routes not serving all customers, the unrouted customers are inserted during the shaking phase or during the local search phase. Our experiments have shown that all customers are served in feasible routes within a reasonable number of iterations.

Table 2
Definition of the heuristic parameters.

Parameter	Definition
α	Penalty on time window constraint
β	Penalty on vehicle capacity constraint
χ	Penalty on maximum duration constraint
δ	Penalty power

Our implementation of the VNS can also be described as a sequence of two components: a *stochastic* component which consists of the randomized selection of a neighborhood structure during the shaking phase, and a *deterministic* component which consists of the application of local search operators during each iteration of the descent phase.

4.2. Shaking phase

The shaking procedure is called whenever no improvement in the current solution has been made after a preset number of iterations. We define a set of neighborhoods structures $N_\kappa(\kappa = 1, \ldots, \kappa_{max})$ from which we randomly choose one neighborhood κ, even if exploring this neighborhood would increase the current cost of the solution. To define the set of neighborhoods we must achieve a good balance between the need to perturb the current solution and the need to maintain some part of it. Another difficulty is to choose between neighborhoods leading to infeasible and feasible solutions when the number of feasible solutions is limited. Furthermore, we need to define neighborhoods of different sizes.

Hence, the length of the segments which are moved or exchanged is sometimes constrained and can be arbitrarily long. Such moves or exchanges are performed once or several times in a row depending on their complexity. If these shaking steps are not sufficient to escape from local optima, a *restart* option is needed. Hence, we consider that after a certain number Bp of iterations there is a need to recenter the local search around one of the best solutions previously found. Therefore, a list of best solutions with constant size Nb is recorded and used during the computations.

Taillard et al. [20] have proposed the cross-exchange operator with the idea of exchanging two segments from two different routes. This operator is applied to every pair of non-empty routes. It was later modified by Bräysy [4] in such a way that the orientation of each segment changes after being inserted into its new route. This defines the I-cross operator applied during the shaking phase to every pair of non-empty routes. In addition to these two neighborhoods, we also use two simple operators: the exchange of a pair of customers within a single route, and the exchange of a pair of customers between two routes (swap).

We also use a more sophisticated neighborhood structure which consists of exchanging two sequences (possibly of different lengths) of customers between two routes. One of the two sequences is randomly chosen to be inverted. This cross-operator with random inversion is applied to every pair of non-empty routes.

Our set of neighborhood structures in the VNS is not divided into parts as in Polacek et al. [14] who consider routes belonging to the same depot alongside routes belonging to different depots. Instead, we consider that all neighborhood structures have the same probability of being selected. Computational experiments performed during the development of our heuristic led to this choice.

4.3. Local search phase

The solution obtained after the shaking phase is submitted to the local search phase. In this phase we distinguish between two

Please cite this article as: Belhaiza S, et al. A hybrid variable neighborhood tabu search heuristic for the vehicle routing problem with multiple time windows. Computers and Operations Research (2013), http://dx.doi.org/10.1016/j.cor.2013.08.010

subphases: (i) a single-route improvement subphase where each single route is improved by applying customer exchange operators; (ii) a multi-route improvement subphase where inter-route exchange operators are applied to different sets of routes to improve the solution. We use a first-improvement acceptance policy during the local search.

4.3.1. Single-route improvement

During the single-route improvement subphase, different single-route exchange operators are applied to each route. To accept such insertions or moves, the cost of the route should decrease. The operators are applied in an increasing order of complexity. We first apply the relocate operator. We attempt to move one customer *up* in a route, and we then try to move one customer *down*. Moving one customer up consists of randomly choosing a customer and trying to insert it in a randomly chosen position between two customers preceding it in the route. Moving one customer down consists of randomly choosing a customer and trying to insert it in a randomly chosen position between two customers following it in the route. When no improvement is obtained by the first operator, we apply a second single-route operator which consists of exchanging the positions of two customers inside a route. Fig. 1 shows how the positions of customers i and j are exchanged in route k. When no improvement is obtained by the second operator we apply a third single-route operator which consists of inverting a sequence of customers within a route. Fig. 2 shows how the sequence of customers from i to j is inverted in route k.

4.3.2. Multi-route improvement

During the multi-route improvement phase, several inter-route exchange operators are applied to a set of routes. Here, an improvement is defined as a decrease in the current solution total cost while a set of "tabu" structures is avoided (see Section 4.4). We first try to move one customer from each route by placing it in a second route (relocate). Fig. 3 shows how customer i is moved from route k to route k' and placed before customer i'. When no improvement is obtained by the first operator, we try to exchange a pair of customers between each pair of routes (swap). Again, when no improvement is obtained by the second operator, we try to exchange a pair of segments between each pair of routes (cross). When no improvement is obtained by the third operator, we try to exchange three customers between a triplet of routes (3-node swap). Fig. 4 shows how customer i is moved from route k to route k'', how customer i' is moved from route k' to route k, and how customer i'' is moved from route k'' to route k'. If no improvement is obtained by this fourth operator, we try to exchange sequences of customers between each pair of routes (cross). Finally, when no improvement is obtained by the fifth operator we try to exchange three segments between a triplet of routes. Fig. 5 shows how the segment (i,j) is moved from route k to route k'', segment (i',j') is moved from route k' to route k and segment (i'',j'') is moved from route k'' to route k'.

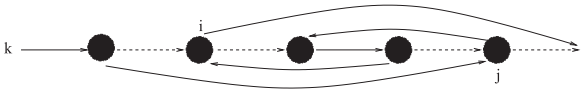

Fig. 1. 2-exchange operator.

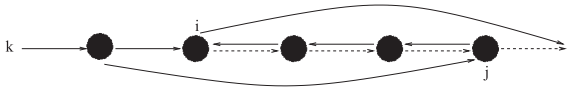

Fig. 2. Sequence invert operator.

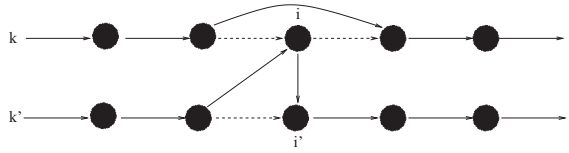

Fig. 3. Relocate operator.

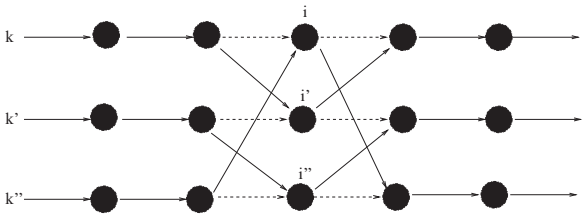

Fig. 4. 3-node swap operator.

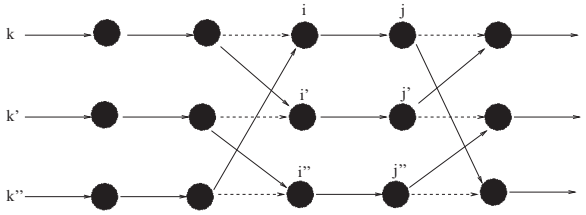

Fig. 5. 3-exchange operator.

4.4. Move-or-not-move phase

Archetti et al. [1] have used tabu search within VNS to solve the team orienteering problem. In their implementation, only feasible solutions are visited. Their tabu list is short and consists of a set of moves associated to routes. In our case, our tabu list consists of a set of solution structures. A list of tabu solution structures L_ϕ is maintained to ensure that the solution obtained does not belong to the last set of solution structures recently explored. The tabu list size is a parameter preset to a reasonable value to avoid wasting too much time or using too much memory. We define a solution structure ϕ_s as the set of attributes $B(s)$ associated with a solution s, such that $B(s)=\{(i,k):\ z_i^k=1\}$. During the local search phase, any move that generates a solution s with a structure $\phi_s \in L_\phi$ is rejected, even if it decreases the current solution cost during the multi-route improvement phase. The solution s would be accepted only if it decreases the best solution cost. The idea of maintaining a list of tabu moves or solutions is largely inspired by the tabu search metaheuristic.

The steps of the Hybrid Variable Neighborhood Tabu Search (HVNTS) heuristic are described in Algorithm 1. We define I, S, L, and G, respectively, as the set of insertion operators, shaking operators, single-route operators and multi-route operators to be applied to a solution s. We also represent the stopping condition by STOP and the shaking condition by SHAKE. The stopping condition refers to a maximum number of iterations, or to a maximum amount of time, or to a maximum number of iterations without improvement of the best solution.

Algorithm 1. HVNTS heuristic.

Step 1. Initialization

$s \leftarrow \varnothing$;
$s \leftarrow s_0 \leftarrow I(s)$;
$n_s \leftarrow n_{s_0}$; the number of customers served by s_0
$s^* \leftarrow s_0$; initialize the best solution
go to step 2;

Step 2. Shaking
generate $\kappa \in N_\kappa$;
$s \leftarrow S_\kappa(s)$, **go to step** 3.*a*;
Step 3. Local search
Step 3.*a*
if $c(s) \le c(s^*)$, **then** $s^* \leftarrow s$;
if $n_s < n$, **then** $s \leftarrow l(s)$, **go to step** 3.*b*;
Step 3.*b*
$s' \leftarrow L(s)$;
if $c(s') \le c(s)$, **then** $s \leftarrow s'$, **repeat step** 3.*b*;
else go to step 3.*c*;
Step 3.*c*
$s' \leftarrow G(s)$;
if $(c(s') \le c(s^*))$ or $(c(s') \le c(s)$ & $B(s') \notin L_\phi)$, **then**
$s \leftarrow s'$, **add** $B(s)$ to L_ϕ, **go to step** 3.*a*;
else if STOP = true **then go to step** 4;
else if SHAKE = true **then go to step** 2;
else repeat step 3.*c*;
Step 4;
if $c(s) \le c(s^*)$ **then** $s^* \leftarrow s$, return s^*.

5. Minimizing route duration

Once a route's order is obtained, it is often possible to delay the departure time from the depot without violating the time windows constraints of customers, which can reduce the total route duration. Some infeasible routes can also become feasible. Savelsbergh [17] has introduced the concept of forward time slack to postpone the beginning of service at a given customer. Cordeau et al. [5] have used this concept to improve their results on multi-depot and other VRPTW instances with maximum duration significantly shorter than the planning horizon given by the depot opening hours. While it computes the forward time slack at each customer after a route is generated, Salvelsbergh's procedure seems not to be extendable to the case of multiple time windows. Tricoire et al. [21] proposed an exact procedure to minimize route duration for the team orienteering problem with multiple time windows. This procedure integrates the travel times into the service times at customers and performs a preprocessing to tighten time windows and eliminate useless ones. The preprocessing evaluates the feasibility of a route with respect to the time windows constraints only. Then, the algorithm proceeds by enumerating all interesting solutions from earliest to latest and keeping track of the best one. To do so, Tricoire et al. use the concept of dominant solution, where a solution is defined as a feasible vector of departure times. Tricoire et al. prove that their algorithm returns the minimum duration solution in polynomial time under the following assumptions: time windows at customers are not allowed to overlap, and the triangular inequality holds for the travel time matrix.

5.1. Minimum backward time slack algorithm

Independently of Tricoire et al.'s algorithm, we have developed an alternative algorithm that records the minimum waiting time and the minimum delay during route generation and adjusts the arrival times backward during their computation. Appendix A details the procedures to be implemented to complete the "Minimum Backward Time Slack Algorithm". The parameters to be used by the algorithm are detailed in Table 3. The time arrival delay at customer i, denoted by θ_i, is the amount of time by which the arrival time at customer i could be delayed without violating at least one of its time window constraints. Algorithm 2 initializes the arrival times for a given route such that $a_i = a_{i-1} + s_{i-1} + t_{i-1,i}$, using $a_0 = l_0$ as the time departure from the depot. Then, for every time window p at customer i, Algorithm 3 computes a time delay window $[\lambda_i^p, \gamma_i^p]$ which represents the lower and upper bounds on the time delay that could be added to a given arrival time a_i to ensure feasibility of the time window p. Hence, $\lambda_i^p = l_i^p - a_i$ and $\gamma_i^p = u_i^p - a_i$. We assume that all time delay windows at customer i are ordered from earliest to latest λ_i^p.

In order to enumerate all possibilities, Algorithm 4 initializes $\hat{w}$ to 10 000 and all time delays to 0, and calls the *Minimum Backward Time Slack Main Procedure* GetWait() (Algorithm 5). The total waiting time found by GetWait() is compared to $\hat{w}$, the best total waiting time recorded. If an improvement is obtained, the best time delays $\hat{\theta}_i$ are updated (Algorithm 6). In the case where $\hat{w} = 0$, the enumeration is stopped since no better solution can be obtained. Finally, the arrival times are updated by adding to each a_i the corresponding best time delay obtained.

Table 4
Time windows and time delay windows.

Node	$[l_i^1, u_i^1]$	$[l_i^2, u_i^2]$	$[l_i^3, u_i^3]$	$[\lambda_i^1, \gamma_i^1]$	$[\lambda_i^2, \gamma_i^2]$	$[\lambda_i^3, \gamma_i^3]$
0	(0, 1000)			(0, 1000)		
1	(10, 20)	(40, 50)	(100, 110)	(0, 10)	(30, 40)	(90, 100)
2	(70, 80)	(120, 130)		(50, 60)	(100, 110)	
3	(60, 70)	(80, 90)	(120, 130)	(30, 40)	(50, 60)	(90, 100)
4	(90, 100)	(140, 150)		(50, 60)	(100, 110)	

Table 3
Minimum delay parameters.

Parameter	Definition
a_i	Arrival time at customer i
s_i	Service time of customer i
θ	Vector of delay of time arrivals at customers
θ_i	Delay of time arrival at customer i
$\hat{\theta}_i$	Best delay of time arrival at customer i
τ	Equal to $\lambda_i^p - \theta_i$ if $\lambda_i^p \ge \theta_i$
w	Current total waiting time on the route
$\hat{w}$	Best total waiting time on the route
$t_{i,i+1}$	Travel time from i to $i+1$
n	Number of customers in route
l_i^p	Time window p lower bound at customer i
u_i^p	Time window p upper bound at customer i
λ_i^p	Lower bound on delay to time window p of player i
γ_i^p	Upper bound on delay to time window p of player i
v_i	Equal to 1 if and only if customer i is served within one of its time windows
ρ	Maximum possible delay of the customers in the preceding sequence of a route

Please cite this article as: Belhaiza S, et al. A hybrid variable neighborhood tabu search heuristic for the vehicle routing problem with multiple time windows. Computers and Operations Research (2013), http://dx.doi.org/10.1016/j.cor.2013.08.010

Our algorithm enumerates all maximal allowable time window delays on a given route and guarantees that the total waiting time is minimized by iteratively updating the arrival times at all customers. To do so, for every customer i with p_i time windows, $(i=1,\ldots,n)$ in a route, the delay of time arrival θ_i is initialized at θ_{i-1}. While $p \le p_i$, if some waiting time $\tau=\lambda_i^p-\theta_i$ is to be generated at customer i, all preceding arrival times can be delayed by $\min\{\tau,\rho\}$. In such a situation, two cases have to be considered. In the first case where $\rho<\tau$, the modification index j would stop at

Table 5
Trace of duration minimization algorithm.

i	$\to\theta$	p	τ	$\to\rho$	$\rho<\tau$	w	$\rho\ge\tau$	$\theta\to$	$\rho\to$
1	[0, 0, 0, 0]	1	0	10 000	No	0	Yes	[0, 0, 0, 0]	10
	[0, 0, 0, 0]	2	30	10 000	No	0	Yes	[30, 0, 0, 0]	10
	[0, 0, 0, 0]	3	90	10 000	No	0	Yes	[90, 0, 0, 0]	10
2	[0, 0, 0, 0]	1	50	10	Yes	40	No	[10, 50, 0, 0]	0
	[30, 0, 0, 0]	1	20	10	Yes	10	No	[40, 50, 0, 0]	0
	[90, 0, 0, 0]	1	–	–	–	–	–	–	–
	[90, 0, 0, 0]	2	10	10	No	0	Yes	[100, 100, 0, 0]	0
3	[10, 50, 0, 0]	1	–	–	–	–	–	–	–
	[40, 50, 0, 0]	1	–	–	–	–	–	–	–
	[100, 100, 0, 0]	1	–	–	–	–	–	–	–
	[10, 50, 0, 0]	2	0	0	No	40	Yes	[10, 50, 50, 0]	0
	[40, 50, 0, 0]	2	0	0	No	10	Yes	[40, 50, 50, 0]	0
	[100, 100, 0, 0]	2	–	–	–	–	–	–	–
	[10, 50, 0, 0]	3	40	0	Yes	80	No	[10, 50, 90, 0]	0
	[40, 50, 0, 0]	3	40	0	Yes	50	No	[40, 50, 90, 0]	0
	[100, 100, 0, 0]	3	–	0	–	0	–	[100, 100, 100, 0]	0
4	[10, 50, 50, 0]	1	0	0	No	40	Yes	[10, 50, 50, 50]	0
	[40, 50, 50, 0]	1	0	0	No	10	Yes	[40, 50, 50, 50]	0
	[10, 50, 90, 0]	1	–	–	–	–	–	–	–
	[40, 50, 90, 0]	1	–	–	–	–	–	–	–
	[100, 100, 100, 0]	1	–	–	–	–	–	–	–
	[10, 50, 50, 0]	2	50	0	Yes	90	No	[10, 50, 50, 100]	0
	[40, 50, 50, 0]	2	50	0	Yes	60	No	[40, 50, 50, 100]	0
	[10, 50, 90, 0]	2	10	0	Yes	90	No	[10, 50, 90, 100]	0
	[40, 50, 90, 0]	2	10	0	Yes	60	No	[40, 50, 90, 100]	0
	[100, 100, 100, 0]	2	0	0	–	0	–	[100, 100, 100, 100]	0

Table 7
Results on the modified instances of Fisher, $B=0$, Set 1 and Set 2.

Instance	Best R.	Avg.	Gap %	Instance	Best R.	Avg.	Gap %
F01	613.97	613.97	0.00	F29	991.54	991.54	0.00
F02	613.97	613.97	0.00	F30	991.54	991.54	0.00
F03	613.97	613.97	0.00	F31	991.54	991.54	0.00
F04	613.97	613.97	0.00	F32	991.54	991.54	0.00
F05	613.97	613.97	0.00	F33	991.54	991.54	0.00
F06	613.97	613.97	0.00	F34	991.54	991.54	0.00
F07	613.97	613.97	0.00	F35	991.54	991.54	0.00
F08	613.97	613.97	0.00	F36	991.54	991.54	0.00
F09	613.97	615.22	0.20	F37	991.54	991.54	0.00
F10	613.97	613.97	0.00	F38	991.54	991.54	0.00
F11	613.97	617.34	0.55	F39	991.54	991.54	0.00
F12	613.97	617.34	0.55	F40	991.54	991.54	0.00
F13	613.97	614.35	0.06	F41	991.54	991.54	0.00
F14	613.97	614.35	0.06	F42	991.54	991.54	0.00
F15	613.97	617.34	0.55	F43	991.54	991.54	0.00
F16	613.97	614.35	0.06	F44	991.54	991.54	0.00
F17	613.97	614.35	0.06	F45	991.54	991.54	0.00
F18	613.97	614.35	0.06	F46	991.54	991.54	0.00
F19	613.97	613.97	0.00	F47	991.54	991.54	0.00
F20	613.97	614.35	0.06	F48	991.54	991.54	0.00
F21	613.97	614.35	0.06	F49	991.54	991.54	0.00
F22	613.97	614.35	0.06	F50	991.54	991.54	0.00
F23	617.00	621.05	0.66	F51	991.54	991.54	0.00
F24	613.97	613.97	0.00	F52	991.54	991.54	0.00
F25	613.97	614.35	0.06	F53	991.54	991.54	0.00
F26	613.97	617.07	0.51	F54	991.54	991.54	0.00
F27	614.58	614.58	0.00	F55	991.54	991.54	0.00
F28	614.58	614.58	0.00	F56	991.54	991.54	0.00
Average	614.12	614.90	0.13	Average	991.54	991.54	0.00

Table 6
Results on Favaretto et al.'s instances, $B=1$, Set 1 and Set 2.

Instance	Best NP.	Best R.	Avg.	Gap %	vs FMP %	Time	Instance	Best NP.	Best R.	Avg.	Gap %	vs FMP %	Time
P01	630.303	614	614	0	−23.94	18.5	P29	1021.88	1021.88	1021.88	0	−18.41	3.5
P02	624.045	614	614	0	−18.40	23.2	P30	1052.67	1035.57	1036.07	0.05	−1.88	9.9
P03	630.992	615.71	615.71	0	−18.08	64	P31	1047.44	1047.43	1047.43	0	3.38	5.4
P04	634.062	618.35	618.35	0	−11.91	35.8	P32	1054.57	1052.28	1052.28	0	−2.73	7.8
P05	620.686	614	614	0	−31.43	64.9	P33	1044.4	1040.08	1040.08	0	−6.82	5.7
P06	622.304	614	614	0	−25.41	89.2	P34	1155.01	1060.49	1060.49	0	−12.59	10.3
P07	661.269	643.9	645.21	0	−23.93	70.2	P35	1050.15	1049.67	1049.77	0.01	−2.32	8.9
P08	654.97	643.88	646.12	0.2	−14.29	72.9	P36	1065.71	1050.02	1050.56	0	−13.18	8.7
P09	655.138	648.54	651.68	0.35	−35.38	70.3	P37	1035.1	1029.08	1029.08	0	−20.01	13.4
P10	623.092	614.45	617.46	0.48	−47.28	64.9	P38	1028.02	1017.42	1018.81	0.14	−30.43	19.8
P11	626.827	614	615.55	0.49	−31.60	65.1	P39	1152.41	1046.69	1046.69	0	2.84	13.9
P12	653.51	637.05	641.43	0.25	−38.61	81.9	P40	1203.74	1048.74	1049.05	0.03	−27.01	7.9
P13	680.978	657.38	660.03	0.69	−35.23	167.8	P41	1093.79	1061.94	1065.39	0.32	−16.58	19.3
P14	699.628	665.28	667.42	0.4	−33.03	172.2	P42	1132.35	1036.15	1036.15	0	−32.99	15.1
P15	673.589	645.96	646.39	0.32	−30.35	71.7	P43	1156.03	1053.1	1053.1	0	−11.48	14.1
P16	676.313	656.13	658.64	0.07	−39.00	148.8	P44	1191.63	1039.02	1041.39	0.23	−27.33	16.9
P17	681.825	658.78	659.12	0.38	−22.25	165.3	P45	1136.57	1092.13	1095.24	0.28	−3.10	15.6
P18	667.855	656.79	662.78	0.05	−27.70	204.3	P46	1069.93	1068.78	1068.79	0	−26.41	14.2
P19	710.188	674.68	677.35	0.4	−19.17	158.7	P47	1117.82	1049.93	1051.99	0.2	−31.72	14.1
P20	683.102	665.53	666.89	0.2	−29.07	89.8	P48	1084.88	1059.91	1061.27	0.13	−12.18	29.6
P21	660.782	638.3	642.64	0.68	−35.67	146.9	P49	1064.38	1020.46	1022.15	0.17	−9.46	23.4
P22	667.411	645.71	649.54	0.59	−31.33	124.9	P50	1010.89	998.43	998.43	0	−23.89	28.2
P23	652.672	631.72	634.16	0.39	−34.87	181.6	P51	1113.43	998.43	998.49	0.01	−8.07	24.8
P24	655.894	630.67	633.42	0.44	−39.69	137.2	P52	1130.49	1004.83	1007.2	0.24	−16.54	36.2
P25	681.019	651.72	654.17	0.38	−29.04	197.4	P53	1118.63	1029.54	1032.8	0.32	−36.37	28.4
P26	715.294	645.2	648.23	0.47	−27.51	274.9	P54	1103.91	1079.6	1084.25	0.43	−17.17	63.3
P27	765.886	667.94	670.13	0.33	−25.32	194.2	P55	1203.05	1021.08	1022.36	0.13	−11.25	70.9
P28	702.956	664.06	665.89	0.28	−13.04	133.5	P56	1218.97	1016.73	1018.94	0.22	−23.93	78.6
Average	664.74	640.99	643.01	0.31	−28.31	117.5	Average	1102.07	1040.35	1041.43	0.10	−15.63	21.7

Please cite this article as: Belhaiza S, et al. A hybrid variable neighborhood tabu search heuristic for the vehicle routing problem with multiple time windows. Computers and Operations Research (2013), http://dx.doi.org/10.1016/j.cor.2013.08.010

$i-1$ as the time delays are updated such that $\theta_j = \theta_j + \rho$. The waiting time at customer i is then reduced to $\tau = \tau - \rho$, the current total waiting time on the route is updated to $w = w + \tau$, the delay θ_i is set to λ_i^p, ρ is set to 0, and v_i is set to 1. The index p is set to $p_i + 1$ because the duration of the route would be less than any other duration obtained with a higher τ, as we will show later in the proof of Proposition 1. In the second case where $\rho \geq \tau$, the modification index j would stop at i as the time delays are updated such that $\theta_j = \theta_j + \tau$. The waiting time at customer i is reduced to 0, the current total waiting time on the route remains constant, ρ is updated to $\rho = \min\{\rho - \tau, \gamma_i^p - \theta_i\}$, and v_i is set to 1.

If $\lambda_i^p \leq \theta_i \leq \gamma_i^p$, ρ is updated to $\rho = \min\{\rho, \gamma_i^p - \theta_i\}$, and v_i is set to 1. If $\theta_i > \gamma_i^p$, then $p = p + 1$. After all customers have been processed, if $v_i = 0$ for some customer i, then the weighted time window penalty is added to w. If w improves $\hat{w}$, the best time delays and the best total waiting time are updated (Algorithm 6). A more formal proof is provided in Appendix A. In the first part of the proof, we show how the choices made by the algorithm yield the least possible route duration if $\theta_i < \lambda_i^p$ initially, and the generated waiting time at customer i is τ. In the second part of the proof, we show how the algorithm does not need to explore all possibilities if $\rho < \tau^p \leq \tau^q$, for some time window $q \neq p$. In the third part and the conclusion of the proof, we detail the simple cases where $\lambda_i^p \leq \theta_i \leq \gamma_i^p$ or $\theta_i > \gamma_i^p$ and conclude on the complexity of the algorithm. We also prove that the proposed procedure returns the route with minimum duration in $O(\prod_{i=1}^{n} p_i)$ time.

5.2. Example

Our algorithm is now illustrated on a route visiting four customers with multiple time windows. The travel time on each one of the visited arcs is 10 time units. The service time at each one of the customers is 0. Initially, the following arrival times are obtained: $a_0 = 0$, $a_1 = 10$, $a_2 = 20$, $a_3 = 30$, $a_4 = 40$. The computed time delay windows are detailed in Table 4.

The algorithm starts first with $\theta_0 = 0$, $\rho = 10\,000$, $\hat{w} = 10\,000$ and $w = 0$. With $i = 1$, we get $\theta_1 \leftarrow 0$, and for $p = 1$, we get $\lambda_1^1 \leq \theta_1 \leq \gamma_1^1$. Therefore, $\rho \leftarrow 10$ and $v_1 \leftarrow 1$. With $i = 2$, we get $\theta_2 \leftarrow 0$, and for $p = 1$, we get $\theta_2 < \lambda_2^1$. Hence, $\tau \leftarrow 50$. Since $\rho < \tau$, we update $\theta_0 \leftarrow 10$ and $\theta_1 \leftarrow 10$. We also get $w \leftarrow 40$, $\theta_2 = 50$ and $\rho \leftarrow 0$. Now we get $\lambda_2^1 \leq \theta_2 \leq \gamma_2^1$. Therefore, $\rho \leftarrow 0$ and $v_2 \leftarrow 1$. With $i = 3$, we get $\theta_3 \leftarrow 50$, and for $p = 1$, we get $\theta_3 > \gamma_3^1$. Then, for $p = 2$, we get $\lambda_3^2 \leq \theta_3 \leq \gamma_3^2$. Therefore, $\rho \leftarrow 0$ and $v_3 \leftarrow 1$. With $i = 4$, we get $\theta_4 \leftarrow 50$, and for $p = 1$, we get $\lambda_4^1 \leq \theta_4 \leq \gamma_4^1$. Therefore, $\rho \leftarrow 0$ and $v_4 \leftarrow 1$. Hence, $w = 10$ is returned and $\hat{w} \leftarrow 10$ is recorded. Table 5 summarizes the trace of algorithm on this example. A route with minimum duration equal to 50 and total waiting time $\hat{w} = 0$ is obtained. The first column "i" indicates the customer being processed. The second column "$\rightarrow\theta$" details the entries of the vector θ at the beginning of the iteration. The third column "p" indicates the index of the time window considered in the iteration. The fourth column "τ" indicates the value of τ, if it is available. The fifth column "$\rightarrow\rho$" indicates the entering value of ρ. The sixth column

Table 8
Characteristics of the new VRPMTW instances.

Instance	$\underline{p}$	$\overline{p}$	$\underline{d}$	$\overline{d}$	$\underline{w}$	$\overline{w}$	Instance	$\underline{p}$	$\overline{p}$	$\underline{d}$	$\overline{d}$	$\underline{w}$	$\overline{w}$
pr101	1	4	50	150	30	50	pr201	2	6	50	500	50	100
pr102	2	4	70	120	20	40	pr202	2	5	50	700	50	100
pr103	3	4	50	70	20	40	pr203	1	4	50	1000	50	100
pr104	3	6	30	50	10	30	pr204	1	3	100	1000	100	200
pc101	2	6	50	200	50	100	pc201	1	3	1000	2000	400	500
pc102	2	6	50	200	100	200	pc202	1	4	1000	1500	400	500
pc103	1	5	100	300	100	200	pc203	2	4	700	1200	300	400
pc104	1	4	100	500	100	500	pc204	2	5	600	1100	200	300
prc101	1	2	200	240	100	200	prc201	1	3	300	400	50	100
prc102	1	2	160	200	100	200	prc202	1	3	400	500	50	200
prc103	1	2	160	200	70	80	prc203	1	3	400	500	100	200
prc104	1	2	140	180	50	60	prc204	1	2	200	700	100	200
rm101	5	9	10	10	10	30	rm201	5	8	50	100	50	100
rm102	5	7	10	30	10	30	rm202	3	5	50	300	50	100
rm103	4	7	10	50	10	30	rm203	2	5	50	500	50	100
rm104	3	6	10	70	10	30	rm204	2	4	50	700	50	100
rm105	2	6	10	100	10	30	rm205	1	4	50	1000	50	100
rm106	2	3	50	100	30	50	rm206	1	3	100	1000	100	200
rm107	1	3	50	150	30	50	rm207	1	3	200	1000	100	200
rm108	1	2	100	200	50	100	rm208	1	5	500	1000	100	200
cm101	5	10	10	50	50	100	cm201	5	10	100	150	50	100
cm102	5	7	10	70	50	100	cm202	5	7	100	200	50	100
cm103	3	7	10	100	50	100	cm203	3	7	100	300	50	100
cm104	3	5	10	100	50	100	cm204	3	5	100	500	50	100
cm105	2	5	50	200	50	100	cm205	2	5	200	500	100	200
cm106	2	4	50	200	100	200	cm206	2	4	200	700	100	200
cm107	1	3	100	300	100	200	cm207	1	3	200	1000	100	300
cm108	1	3	100	500	100	500	cm208	1	3	500	1000	100	500
rcm101	5	10	10	30	10	30	rcm201	5	10	100	150	50	100
rcm102	5	7	10	30	10	50	rcm202	5	7	100	200	50	100
rcm103	3	7	10	50	10	50	rcm203	3	7	100	300	50	100
rcm104	3	5	10	50	10	50	rcm204	3	5	100	500	50	100
rcm105	2	5	10	70	10	70	rcm205	2	5	200	500	100	200
rcm106	2	4	30	70	30	70	rcm206	2	4	200	700	100	200
rcm107	1	3	30	100	30	70	rcm207	1	3	200	1000	100	300
rcm108	1	3	30	100	30	100	rcm208	1	3	500	1000	100	500

Please cite this article as: Belhaiza S, et al. A hybrid variable neighborhood tabu search heuristic for the vehicle routing problem with multiple time windows. Computers and Operations Research (2013), http://dx.doi.org/10.1016/j.cor.2013.08.010

"$\rho < \tau$" indicates if the entering value of ρ is less than τ or no. The seventh column "w" indicates the current value of the total waiting time in the route. The eighth column "$\rho \geq \tau$" indicates if the entering value of ρ is greater or equal to τ. The ninth column $\theta \rightarrow$ details the entries of the vector θ at the end of the iteration. The last column "$\rho \rightarrow$" indicates the exiting value ρ.

6. Computational results

This section presents our experimental results on benchmark instances of the Vehicle Routing Problem with Multiple Time Windows (VRPMTW) [7]. We report our best and average computational results on different sets of instances. Using a C++ implementation, these experimental results were obtained under MS Windows on workstations with 3.3 GHz Intel Core i5 vPro processors, and 3.2 GB RAM.

6.1. Results on the instances of Favaretto et al. and Fisher

Two sets (Set 1 and Set 2) of VRPMTW instances were generated by Favaretto et al. [7]. These two sets respectively contain 71 and 44 customers to be served by four vehicles starting and ending their routes at the same depot. These two sets of instances were generated using two VRP instances without time windows solved to optimality by Fisher [8]. Each set of instances consists of 28 VRP instances with single or multiple time windows. Favaretto et al. used the solutions proven to be optimal by Fisher [8] to generate for each customer one time window containing the arrival time corresponding to the optimal solution.

Unfortunately, the Favaretto et al. instances contain some wrong coordinates and demands compared to the original instances used by Fisher. Therefore, we can only compare our results to those of Favaretto et al. when the total duration time is to be minimized ($B=1$). After correcting these mistakes, we compare our results to the optimal solution found by Fisher when the total travel time is to be minimized ($B=0$) with the same time windows as those used by Favaretto et al.

In their instances, Favaretto et al. considered 92.42 units of fixed cost per vehicle used for Set 1, and 68.015 units of fixed cost per vehicle used for Set 2. The value of the best lower bound solution for Set 1 instances (1–28) is 614. The value of the best lower bound solution for Set 2 instances (29–56) is 991. For Fisher's modified instances (with time windows constraints), the value of the best lower bound solution for Set 1 instances (1–28) is 613.97. It includes 241.97 as the total travel time of the routes, and 93 units of fixed cost per vehicle used. The value of the best lower bound solution for Set 2 instances (29–56) is 991.54. It includes 723.54 as the total duration of the routes, and 67 units of fixed cost per vehicle used. For these two sets of instances, the best lower bound solution is considered as the minimum possible cost of the optimal solution without time windows constraints. This means that for some instances the real optimal solution cost could be larger than its best lower bound.

The computational experiments performed on the instances of Favaretto et al. and on the modified instances of Fisher are presented in Tables 6 and 7. The column "Instance" indicates the name of the problem. The column "Best NP." indicates the best solution found by HVNTS, without the minimum backward slack time procedure, on 10 randomly generated runs (25 000 iterations). The column "Best R." indicates the best solution found by HVNTS on 10 randomly generated runs (25 000 iterations). The column "Avg." indicates the average solution found by HVNTS on the 10 runs. The column "Gap %" indicates the average percentage gap between the average and the best solution on the 10 runs. The column "vs FMP %" indicates the gap in percentage between the best solution found by HVNTS and the best solution reported by Favaretto et al. The column "vs lb %" indicates the percentage gap between the best solution found by HVNTS and the best known lower bound. Finally, the column "Time" indicates the average time, in seconds, required for each run.

On the instances of Favaretto et al., for 10 consecutive randomly generated runs our heuristic obtains an average solution equal to 643.01 on Set 1 and to 1041.43 on Set 2. The average best solution obtained by HVNTS over all runs is 640.99 on Set 1 and 1040.35 on Set 2. The HVNTS best solution value is on average −28.31% and −15.63% from the best solution reported by Favaretto et al. [7]. These results also show that the minimum backward time slack procedure helped decrease the average best solution value by 3.57% on the first set of instances and by 5.60% on the second set of instances. The large gaps between the two methods performances is due to the fact that, to the best of our knowledge, Favaretto et al. did not use any algorithm to find the minimum route duration. Favaretto et al. did not provide results for $B=0$.

On the modified instances of Fisher, for 10 consecutive randomly generated runs, our heuristic obtains an average solution equal to 614.90 on Set 1 and 991.54 on Set 2. The average best solution obtained by HVNTS over all runs is 614.12 on Set 1 and 991.54 on Set 2. The HVNTS best solution values are on average 0.13% and 0.00% superior to the best known lower bound.

6.2. Results on new VRPMTW instances

We have used Solomon's [18] VRPTW instances to generate two different types of new VRPMTW instances. For the sets of instances PR1, PR2, PC1, PC2, PRC1 and PRC2, the best known solutions are

Table 9
Results on the new VRPMTW instances I. $B=1$.

Instance	m	BKS	Best R.	Avg.	Gap %	Time	vs BKS
pr101	9	3665.95	3665.95	3671.12	0.14	43.3	0.00
pr102	9	3665.95	3666.87	3672.41	0.15	47.4	0.03
pr103	9	3665.95	3666.02	3672.13	0.17	57.3	0.00
pr104	9	3665.95	3666.46	3673.15	0.18	88.4	0.01
Average	9	3665.95	3666.3	3672.2	0.16	59.1	0.04
pr201	2	3652.07	3652.07	3671.24	0.52	90.8	0.00
pr202	2	3652.07	3652.07	3670.24	0.49	88.7	0.00
pr203	2	3652.07	3652.07	3670.21	0.50	95.5	0.00
pr204	2	3652.07	3652.07	3671.12	0.52	98.9	0.00
Average	2	3652.07	3652.07	3670.70	0.51	93.5	0.00
pc101	10	11 824.8	11 824.8	11 848.20	0.20	79.9	0.00
pc102	10	11 824.8	11 824.80	11 859.00	0.29	64.6	0.00
pc103	10	11 824.8	11 829.14	11 864.00	0.29	43.6	0.04
pc104	10	11 824.8	11 824.80	11 858.80	0.29	39.8	0.00
Average	10	11 824.8	11 825.8	11 857.5	0.27	56.9	0.01
pc201	3	11 684.28	11 684.28	11 717.40	0.28	76.2	0.00
pc202	3	11 684.28	11 684.28	11 718.20	0.29	73.6	0.00
pc203	3	11 684.28	11 684.28	11 735.50	0.44	84.6	0.00
pc204	3	11 684.28	11 699.43	11 803.40	0.89	98.8	0.13
Average	3	11 684.28	11 688.1	11 743.6	0.48	83.3	0.03
prc101	9	3795.59	3800.19	3808.42	0.22	18.7	0.12
prc102	9	3795.59	3795.59	3803.40	0.21	21.7	0.00
prc103	9	3795.59	3795.59	3813.41	0.47	33.8	0.00
prc104	9	3795.59	3795.59	3811.25	0.41	34.4	0.00
Average	9	3795.59	3796.7	3809.1	0.33	27.2	0.06
prc201	2	3656.62	3656.62	3662.12	0.25	74.2	0.00
prc202	2	3656.62	3659.90	3664.25	0.12	71.7	0.09
prc203	2	3656.62	3656.62	3660.38	0.10	90.7	0.00
prc204	2	3656.62	3656.62	3656.62	0.00	70.7	0.00
Average	2	3656.62	3657.4	3660.8	0.09	76.8	0.02
Average	5.83	6379.9	6381.1	6402.3	0.31	66.1	0.02

known. For the sets of instances RM1, RM2, CM1, CM2, RCM1 and RCM2, the best known solutions are not known. Every new instance contains 100 customers and belongs to one of twelve sets. Sets PR1, PR2, RM1 and RM2 were generated from Solomon's instances sets R1 and R2. In these sets, customers are randomly and uniformly distributed. Sets PC1, PC2, CM1 and CM2 were generated from Solomon's instances sets C1 and C2. In these sets, customers are clustered. Sets PRC1, PRC2, RCM1 and RCM2 were generated from Solomon's instances sets RC1 and RC2. These sets contain a mix of uniformly distributed and clustered customers. The depot has a narrow time window in instances of type 1 and a longer horizon in instances of type 2.

Every newly generated instance can be described by means of some parameters. Let $\underline{p}$ and $\overline{p}$ be the minimum and maximum

Table 10
Results on the new VRPMTW instances II.

Instance	m	$B=0$				$B=1$			
		Best R.	Avg.	Gap %	Time	Best R.	Avg.	Gap %	Time
rm101	10	2977.2	3005	0.93	66.4	4041.9	4072.9	0.77	88.1
rm102	9	2759.4	2759.4	0.00	57.8	3765.1	3765.1	0.77	98.3
rm103	9	2692.5	2710.5	0.67	47.2	3708.5	3736.5	0.76	90.2
rm104	9	2696.6	2719.2	0.84	73.9	3718.0	3722.9	0.13	80.3
rm105	9	2688.8	2711.0	0.82	84.1	3688.8	3716.9	0.76	64.6
rm106	9	2691.9	2691.9	0.00	75.2	3692.9	3743.7	1.37	41.2
rm107	9	2690.8	2714.7	0.48	43.6	3701.4	3714.1	0.34	45.8
rm108	9	2729.1	2729.1	0.00	53.2	3729.1	3729.1	0.00	43.1
Average	9.1	2740.8	2753.7	0.47	62.7	3755.7	3775.2	0.52	69
rm201	3	3711.4	3720.5	0.24	43	4808.2	4847.4	0.82	92.8
rm202	2	2698.1	2717.3	0.71	40.2	3739.0	3775.7	0.98	95.7
rm203	2	2686.1	2702	0.59	46.6	3710.3	3728.3	0.48	86.8
rm204	2	2680.5	2691.2	0.40	47.9	3691.9	3708.8	0.46	79
rm205	2	2671.0	2688.2	0.64	44.3	3689.9	3707.7	0.48	63.5
rm206	2	2686.3	2704.9	0.69	31.9	3703.4	3720.3	0.46	88.1
rm207	2	2678.2	2696.2	0.67	51.6	3701.7	3719.4	0.48	92.5
rm208	2	2673.9	2690.7	0.63	43.3	3682.8	3699.6	0.46	81.7
Average	2.1	2810.7	2826.4	0.57	43.6	3850.1	3876.9	0.58	85.0
cm101	10	3089.2	3102.4	0.43	102.3	12 320.0	12 344.4	0.20	96.2
cm102	12	3426.9	3426.9	0.00	96.8	12 492.1	12 492.1	0.00	96.8
cm103	12	3532.7	3572.7	1.13	90.7	12 641.2	12 687.7	0.37	72.5
cm104	14	4051.3	4058	0.17	69.2	13 087.8	13 117.9	0.23	62.4
cm105	11	3060.6	3077.3	0.54	66.7	12 083.4	12 144.4	0.50	61.4
cm106	10	2992.4	3020.2	0.93	65.3	12 073.9	12 133.9	0.50	59.7
cm107	11	3256.5	3292.3	1.10	37.9	12 324.2	12 364.1	0.32	39.4
cm108	10	2968.7	2973.1	0.15	31.5	11 990.4	12 012.6	0.19	31.1
Average	10.9	3297.3	3315.4	0.56	70.0	12382.2	12 412.1	0.29	64.9
cm201	5	4436.62	4452.5	0.36	91.9	13 520.1	13 591.7	0.53	96.3
cm202	6	4998.8	5024.9	0.52	90.2	14 027.3	14 060.7	0.24	96.8
cm203	5	4445.8	4484.6	0.87	94.3	13 497.2	13 512.8	0.12	86.3
cm204	5	4335.2	4372.4	0.86	91.5	13 359.8	13 413.7	0.40	87.3
cm205	4	3863.5	3883.2	0.51	99.4	12 884.1	12 963.1	0.61	86.9
cm206	4	3722	3743.2	0.57	84.1	12 767.7	12 811.2	0.39	93.3
cm207	4	3968.4	3977.8	0.24	60.1	13 009.7	13 017.6	0.06	83.3
cm208	4	3771.1	3793.2	0.59	64.9	12 788.1	12 805.2	0.13	77.2
Average	4.6	4192.7	4216.5	0.56	84.6	13 231.0	13 272.0	0.31	88.4
rcm101	10	3062.0	3086.6	0.80	76.5	4098.92	4129.7	0.75	91.9
rcm102	10	3132.2	3142.3	0.32	60.1	4222.61	4228.4	0.14	56.1
rcm103	10	3152.9	3163.8	0.35	75.1	4174.25	4185.4	0.27	52.2
rcm104	10	3119.6	3134.6	0.48	63.8	4156.26	4170.7	0.35	65.8
rcm105	10	3187.9	3210.7	0.71	64.7	4216.65	4227.0	0.25	46.1
rcm106	10	3218.9	3218.9	0.00	54.7	4219.93	4236.3	0.39	46.2
rcm107	11	3488.9	3514.0	0.72	35.4	4542.38	4560.8	0.41	24.0
rcm108	11	3592.7	3592.7	0.00	35.6	4614.49	4614.5	0.00	36.6
Average	10.3	3244.4	3257.9	0.42	58.2	4280.7	4294.1	0.32	52.4
rcm201	2	2804.0	2827.8	0.85	69.2	3783.6	3824.5	1.08	72.9
rcm202	2	2836.9	2846.8	0.35	78.9	3847.1	3847.1	0.00	78.9
rcm203	2	2721.9	2725.4	0.13	92.2	3721.9	3725.4	0.09	92.2
rcm204	2	2726.5	2743.1	0.61	75.9	3726.5	3743.1	0.44	75.9
rcm205	2	2754.5	2775.7	0.77	71.7	3754.5	3775.7	0.56	71.7
rcm206	2	2812.7	2830.6	0.64	21.5	3812.7	3830.6	0.47	21.5
rcm207	3	3749.8	3786.8	0.99	67.9	4764.2	4792.2	0.59	61.4
rcm208	2	2791.4	2817.2	0.92	21.4	3791.4	3817.2	0.68	21.4
Average	2.1	2899.7	2919.2	0.66	62.3	3900.3	3919.5	0.49	62.0
Average	6.52	3197.9	3214.8	0.54	63.6	6897.5	6922.7	0.42	70.3

Please cite this article as: Belhaiza S, et al. A hybrid variable neighborhood tabu search heuristic for the vehicle routing problem with multiple time windows. Computers and Operations Research (2013), http://dx.doi.org/10.1016/j.cor.2013.08.010

numbers of time windows for a customer, let $\underline{d}$ and $\overline{d}$ be the minimum and maximum distance between two consecutive time windows, and let $\underline{w}$ and $\overline{w}$ be the minimum and maximum width of a time window for a customer. Table 8 summarizes the characteristics of each instance.

The computational experiments performed on the new VRPMTW instances are presented in Tables 9 and 10. Column "m" indicates the number of vehicles used by the best solution. We considered that the fixed cost per vehicle used should be proportional to the capacity of the vehicle. Therefore, the fixed costs considered are equal to 200, 1000, 200, 700, 200 and 1000, respectively, for the sets of instances PR1, PR2, PC1, PC2, PRC1 and PRC2, and for the set of instances RM1, RM2, CM1, CM2, RCM1 and RCM2.

6.2.1. New instances with best known solutions

The best solutions, and most probably the optimal ones, were obtained after relaxing the time windows constraints for the sets of instances R1, R2, C1, C2, RC1 and RC2. We have used our implementations of tabu search and our HVNTS heuristic to obtain these best solutions. We then used these best solutions to generate for each customer at least one time window containing its arrival time in the best solution. These instances involve overlapping time windows. The obtained instances define our first set of instances PR1, PR2, PC1, PC2, PRC1 and PRC2. The computational experiments performed on this first set of new VRPMTW instances are presented in Table 9. Column "BKS" indicates the cost of the best known and most probably optimal solution. Column "vs BKS"' indicates the gap between the best solution found by HVNTS and the best known solution. The results in Table 9 show that the HVNTS heuristic finds the best known solution on almost every set of runs.

6.2.2. New instances without best known solutions

For the newly generated VRPMTW instances RM1, RM2, CM1, CM2, RCM1 and RCM2, the average number of time windows considered for each customer is larger than the average in the previous set of instances. These instances do not involve overlapping time windows and the number of vehicles considered is larger than the number of vehicles in the previous set. On these newly generated instances, and for 10 consecutive randomly generated runs, our heuristic obtains an average gap of 0.47%, 0.57%, 0.56%, 0.56%, 0.42% and 0.66%, respectively, for the sets of instances RM1, RM2, CM1, CM2, RCM1 and RCM2, when $B=0$. The average gap is 0.52%, 0.58%, 0.29%, 0.31%, 0.32% and 0.49%, for these sets, when $B=1$. The overall average gap is 0.54%, when $B=0$, and 0.42%, when $B=1$.

7. Conclusions

>We have presented a hybridization of the variable neighborhood search metaheuristic using a tabu search memory concept in order to solve the vehicle routing problem with multiple time windows. The paper also described a minimum backward time slack algorithm for this problem. The algorithm records the minimum waiting time and the minimum delay during route generation and adjusts the arrival and departure times backward during their computation. While the procedure proposed by Tricoire et al. [21] to minimize route duration for the team orienteering problem with multiple time windows worked under two assumptions (time windows at customers are not allowed to overlap, and the triangular inequality holds for the travel time matrix), our route minimization procedure only assumes that all time delay windows at customer i are ordered from earliest to latest λ_i^p. At this point, since Tricoire et al.'s procedure and our procedure were applied to two different problems and developed independently from each other, we cannot pretend that one of the procedures is "better" than the other. However, in our context, our procedure is applied to problems with possibly overlapping time windows and with a larger number of time windows, if compared to the context of Tricoire et al.'s procedure. Tricoire et al.'s procedure was applied to problems with at most two time windows per customer, while our procedure was applied to problems with at most ten time windows per customer.

The new algorithm outperforms the ant colony heuristic of Favaretto et al. [7] on benchmark instances involving single and multiple time windows. The HVNTS heuristic also obtains best solution values very close to the best known lower bound. The HVNTS heuristic results are stable on randomly generated consecutive runs. The average gap on Favaretto et al.'s instances is 0.31% and 0.10%. The average gap on the modified instances of Fisher is around 0.13% and 0.00%. Finally, the average gap on the newly generated VRPMTW instances is 0.54%, when $B=0$, and 0.31% and 0.42%, when $B=1$.

Acknowledgments

This work was partially supported by KFUPM Deanship of research under Grant IN101038 and by the Canadian Natural Sciences and Engineering Research Council under Grants 105574-12 and 39682-10. This support is gratefully acknowledged. Thanks are due to the referees for their valuable comments.

Appendix A

We now provide the details of the procedures to be implemented to complete the "Minimum Backward Time Slack Algorithm".

- Algorithm 2 initializes the arrival times.
- Algorithm 3 computes the time delay windows.
- Algorithm 4 sets the time arrivals.
- Algorithm 5 details the main procedure.
- Algorithm 6 records the best time delays and the best total waiting time.

Algorithm 2. IntializeArrivalTimes().

$a_0 = l_0$
for $i = 1, \ldots, n$:
  $a_i \leftarrow a_{i-1} + t_{i-1,i} + s_i$
end for
return

Algorithm 3. ComputeDelayBounds().

for $i = 1, \ldots n$:
  for $p = 1, \ldots, p_i$:
    if $(a_i \le u_i^p)$ **then**
      $\lambda_i^p \leftarrow l_i^p - a_i$
      $\gamma_i^p \leftarrow u_i^p - a_i$
    else
      $\lambda_i^p \leftarrow -1000$
      $\gamma_i^p \leftarrow -1000$
    end if
  end for
end for
return

Please cite this article as: Belhaiza S, et al. A hybrid variable neighborhood tabu search heuristic for the vehicle routing problem with multiple time windows. Computers and Operations Research (2013), http://dx.doi.org/10.1016/j.cor.2013.08.010

Algorithm 4. SetArrivalTimes().

for $i=1,\ldots,n$:
  $\theta_i \leftarrow 0$
  $\hat{\theta}_i \leftarrow 0$
end for
$\hat{w} \leftarrow 10000$
$\theta_0 \leftarrow 0$
GetWait(θ, 10000, 0, 1)
for $i=0,\ldots,n+1$:
  $a_i \leftarrow a_i + \hat{\theta}_i$
end for
return

Algorithm 5. Minimum backward time slack main procedure: GetWait(θ, ρ, w, i).

if ($\hat{w}=0$) **then**
  return;
else if ($i \le n$) **then**
  $\theta_i \leftarrow \theta_{i-1}$;
  $p \leftarrow 1$;
  while ($p \le p_i$) **do**
    $v_i \leftarrow 0$;
    if ($\theta_i < \lambda_i^p$) **then**
      $\tau \leftarrow \lambda_i^p - \theta_i$;
      if ($\rho < \tau$) **then**
        for $j=0$ to $i-1$:
          $\theta_j \leftarrow \theta_j + \rho$;
        end for
        $w \leftarrow w + \tau - \rho$;
        $\theta_i \leftarrow \lambda_i^p$;
        $\rho \leftarrow 0$;
        $v_i \leftarrow 1$;
        $p \leftarrow p_i + 1$;
        GetWait(θ, ρ, w, $i+1$);
      else if ($\rho \ge \tau$) **then**
        for $j=0$ to i :
          $\theta_j \leftarrow \theta_j + \tau$;
        end for
        $\rho \leftarrow \min\{\rho-\tau, \gamma_i^p - \theta_i\}$;
        $v_i \leftarrow 1$;
        GetWait(θ, ρ, w, $i+1$);
      end if
    else if ($\lambda_i^p \le \theta_i \le \gamma_i^p$) **then**
      $\rho \leftarrow \min\{\rho, \gamma_i^p - \theta_i\}$;
      $v_i \leftarrow 1$;
      GetWait(θ, ρ, w, $i+1$);
    else if ($\theta_i > \gamma_i^p$) **then**
      $p \leftarrow p+1$;
    end if
  end while
else if ($i > n$) **then**
  for $j=1$ to n :
    if ($v_i = 0$) **then**
      $w \leftarrow w + \alpha \min_{p \in W_i}\{|\theta_i - \lambda_i^p|, |\theta_i - \gamma_i^p|\}^{\delta}$;
    end if
  end for
  if ($w < \hat{w}$) **then**
    UpdateBestDelays();
  end if
end if
return;

Algorithm 6. UpdateBestDelays().

$\hat{\theta}_0 \leftarrow \theta_1$;
for $j=1$ to n :
  $\hat{\theta}_j \leftarrow \theta_j$;
end for
$\hat{\theta}_{n+1} \leftarrow \theta_n$;
$\hat{w} \leftarrow w$;
return

Note that if $a_i > u_i^p$, then λ_i^p and γ_i^p are both set to a large negative value (Algorithm 3). This would allow the main procedure to skip this time window. Algorithm 4 initializes θ_0 to 0 and calls the procedure detailed in Algorithm 5. The total waiting time is found by the main procedure *GetWait*() (Algorithm 5). This time is compared to $\hat{w}$, the best total waiting time recorded. If an improvement is obtained, the best time delays $\hat{\theta}_i$ are updated (Algorithm 6). In the case where $\hat{w}=0$, the enumeration is stopped since no better solution can be obtained. Finally, the arrival times are updated by adding to each a_i the corresponding best time delay obtained.

We now prove that the proposed "Minimum Backward Time Slack Algorithm" returns the optimal solution to the minimum duration problem with multiple time windows.

Proposition 1. *The Minimum Backward Time Slack Algorithm finds the optimal solution to the minimum duration problem with multiple time windows.*

Proof. The initial arrival times are computed regardless of the time windows constraints. Hence, for each customer i in the route we have

$$\theta_1 \le \ldots \le \theta_i \le \ldots \le \theta_n.$$

Thus, θ_i is initialized to its lower bound θ_{i-1}, before verifying the feasibility of the time windows at customer i. We suppose that the time delay windows at each customer are ordered from earliest to latest lower bound. Therefore, the time delay θ_i is compared to the earliest encountered time delay window lower bound λ_i^p. In the first part of the proof, we show that the choices made by the algorithm yield the least possible route duration if $\theta_i < \lambda_i^p$ initially, and the generated waiting time at customer i is τ. In the second part of the proof, we show that the algorithm does not need to explore all possibilities if $\rho < \tau^p \le \tau^q$, for some time window $q \ne p$. In the third part and the conclusion of the proof, we detail the simple cases where $\lambda_i^p \le \theta_i \le \gamma_i^p$ or $\theta_i > \gamma_i^p$ and conclude on the complexity of the algorithm.

Part I. ($\theta_i < \lambda_i^p$)

In the case where $\rho < \tau$, the time delays of the preceding customers in the route can be increased by any value $\epsilon \in [0, \rho]$, while the time delay θ_i is updated to λ_i^p. If the time delays of customers 1 to $i-1$ are increased by ϵ, the duration of the sequence of the route 1 to i is equal to $\lambda_i^p - (\theta_1 + \epsilon)$ and larger than the duration of the same sequence if $\epsilon = \rho$:

$$\lambda_i^p - (\theta_1 + \epsilon) \ge \lambda_i^p - (\theta_1 + \rho).$$

This means that the new maximum delay of the sequence 1 to i would be $\rho_\epsilon = \min\{\rho - \epsilon, \gamma_i^p - \lambda_i^p\}$ which is larger than $\rho = 0$ if $\epsilon = \rho$. Moreover, the waiting time generated would be $w_\epsilon = \tau - \epsilon$ and larger than $w = \tau - \rho$. If $\rho_\epsilon = \rho - \epsilon$, w_ϵ can be reduced by at most

Please cite this article as: Belhaiza S, et al. A hybrid variable neighborhood tabu search heuristic for the vehicle routing problem with multiple time windows. Computers and Operations Research (2013), http://dx.doi.org/10.1016/j.cor.2013.08.010

$\rho-\epsilon$. Hence, we would always have

$$w_\epsilon \geq \tau-\epsilon-(\rho_\epsilon)=\tau-\rho.$$

If $\rho_\epsilon=\gamma_i^p-\lambda_i^p$, w_ϵ can be reduced by at most $\gamma_i^p-\lambda_i^p$. Hence, we would always have

$$w_\epsilon \geq \tau-\epsilon-(\rho_\epsilon)=\tau-\epsilon-(\gamma_i^p-\lambda_i^p)\geq \tau-\epsilon-(\rho-\epsilon)=\tau-\rho.$$

Therefore, if $\epsilon \leq \rho$ is chosen, the duration of the sequence 1 to i would always remain larger than the duration of the same sequence if ρ was chosen. Hence, the choice of ρ is better than any other choice if $\rho<\tau$.

In the case where $\tau \leq \rho$, the time delays of the preceding customers in the route can be increased by any value $\epsilon \in [0,\tau]$, while the time delay θ_i is updated to λ_i^p. If the time delays of customers 1 to $i-1$ are delayed by ϵ, the duration of the sequence of the route from 1 to i is equal to $\lambda_i^p-(\theta_1+\epsilon)$ and larger than the duration of the same sequence if $\epsilon=\tau$:

$$\lambda_i^p-(\theta_1+\epsilon)\geq \lambda_i^p-(\theta_1+\tau).$$

This means that the new maximum delay of the sequence 1 to i would be $\rho_\epsilon=\min(\rho-\epsilon,\gamma_i^p-\lambda_i^p)$ and larger than $\rho=\min\{\rho-\tau,\gamma_i^p-\lambda_i^p\}$, if $\epsilon=\tau$. Moreover, the waiting time generated would be $w_\epsilon=\tau-\epsilon$ and larger than $w=0$. If $\rho_\epsilon=\rho-\epsilon$, w_ϵ can be reduced by at most $\rho-\epsilon$. Hence, we would always have

$$w_\epsilon \geq \tau-\epsilon-(\rho_\epsilon)=\tau-\rho.$$

If $\rho_\epsilon=\gamma_i^p-\lambda_i^p$, w_ϵ can be reduced by at most $\gamma_i^p-\lambda_i^p$. Hence, we would always have

$$w_\epsilon \geq \tau-\epsilon-(\rho_\epsilon)=\tau-\epsilon-(\gamma_i^p-\lambda_i^p)\geq \tau-\epsilon-(\rho-\epsilon)=\tau-\rho\geq 0.$$

Therefore, if $\epsilon \leq \rho$ is chosen, the duration of the sequence 1 to i would always remain larger than the duration of the same sequence if τ was chosen. Hence, the choice of τ is better than any other choice if $\tau \leq \rho$. We can now conclude that the choices made by the algorithm at every customer i such that $\theta_i<\lambda_i^p$ are sequentially better than all other possible choices. Therefore, at $i=n$ the algorithm finds all minimal route durations for all possible combinations of time windows assigned to customers.

Part II. $(\rho<\tau^p\leq\tau^q)$

If $\rho<\tau^p\leq\tau^q$ for some time delay window $q\neq p$, as shown in part I, the best choice for the time delays updating would make the duration of the sequence of customers 1 to i such that $\lambda_i^p-(\theta_1+\rho)\leq\lambda_i^q-(\theta_1+\rho)$, with waiting times generated at i such that $w^p=\tau^p-\rho\leq w^q=\tau^q-\rho$, respectively for time delay windows p and q. Since ρ would become equal to 0 in both cases, the minimum duration of the route if customer i was served within the time window q would not be lower than the minimum duration of the route obtained if customer i is served within the time window p. Therefore, the algorithm should skip the next time delay windows if $\rho<\tau$.

Part III. $(\lambda_i^p\leq\theta_i\leq\gamma_i^p)$ **or** $(\theta_i>\gamma_i^p)$

Now, in the case where $\lambda_i^p\leq\theta_i\leq\gamma_i^p$, ρ is updated to its maximum possible value which is $\min\{\rho,\gamma_i^p-\theta_i\}$. At this stage, customer i is served on its time window p with the least possible duration given the entering values of the vector θ and ρ. In the case where θ_i is strictly larger than the current γ_i^p, the time delay window $p+1$ is to be explored.

The "Minimum Backward Time Slack Algorithm" finds the optimal solution to the minimum duration problem with multiple time windows.

Proposition 2. *The complexity of the Minimum Backward Time Slack Algorithm is* $O(\prod_{i=1}^n p_i)$.

Proof. Since the time delay windows are ordered, the complexity of the setting of θ_i is $O(p_i)$ at each customer i with p_i time windows. Therefore, the complexity of the algorithm is $O(\prod_{i=1}^n p_i)$.

Appendix B

The following tables compile some of the results of the parameters calibration tests. Table 11 compares the results on a sample of instances for different values of the tabu list size. Table 12 compares the results on a sample of instances for different values of the penalty power.

Table 11
Tests on Tabu list size $|L_\phi|$. $\alpha=100$, $\delta=4$, 25 K iterations.

Instance	$\lvert L_\phi\rvert=0$			$\lvert L_\phi\rvert=10$			$\lvert L_\phi\rvert=50$			$\lvert L_\phi\rvert=100$		
	Best R.	Avg.	Time	Best R.	Avg.	Time	Best R.	Avg.	Time	Best R.	Avg.	Time
rm108	3759.7	3774.5	40.6	3753.6	3759.3	44.5	3761.1	3769.9	48.7	3729.1	3729.1	43.1
rm204	3705.2	3725.6	75.9	3695.6	3725.9	77.8	3696.1	3723.3	82.7	3691.9	3708.8	79.0
cm101	12 404.6	12 479.0	67	12 362.5	12 426.8	100.1	12 338.4	12 402.6	74.2	12 355.4	12 398.9	94.8
cm205	12 902.7	12 977.4	101.5	12 903.4	12 954.5	91.2	12 919.5	12 976.9	85.4	12 884.1	12 963.1	86.9
rcm103	4180.3	4186.3	65.4	4176.8	4194.6	49.5	4178.2	4189.2	75.3	4174.3	4185.4	52.2
rcm206	3823.6	3840.1	26.6	3840.9	3853.8	39.7	3834.5	3845.8	32.8	3812.7	3830.6	21.5
Average	6796.0	6830.5	62.9	6788.8	6819.2	67.1	6787.9	6817.9	66.5	6774.6	6802.6	63.2

Table 12
Tests on penalty power δ. $\alpha=100$, $|L_\phi|=100$, 25 K iterations.

Instance	$\delta=0.5$			$\delta=1$			$\delta=2$			$\delta=4$		
	Best R.	Avg.	Time	Best R.	Avg.	Time	Best R.	Avg.	Time	Best R.	Avg.	Time
rm108	3732.3	3747.2	49.2	3755.2	3766.2	47.2	3764.4	3778.2	42.6	3729.1	3729.1	43.1
rm204	3703.6	3731.4	99.5	3707.1	3723.8	84.6	3711.5	3726.9	94.5	3691.9	3708.8	79.01
cm101	12 344.0	12 352.1	100.2	12 354.0	12 372.5	97.8	12 320	12 344.4	96.2	12 355.4	12 398.9	94.8
cm205	12 911.0	12 972.2	112.7	12 904.3	12 959.3	132.4	12 902.6	12 960.6	102.6	12 884.1	12 963.1	86.9
rcm103	4178.0	4192.2	67.1	4168.6	4179.8	60.9	4180.5	4186.2	57.6	4174.3	4185.4	52.2
rcm206	3852.7	3860.2	32.7	3850.7	3858.1	34.5	3812.7	3821.4	29	3812.7	3830.6	21.5
Average	6786.9	6809.2	76.9	6790.0	6809.9	76.2	6781.9	6802.9	70.4	6774.6	6802.6	63.2

Please cite this article as: Belhaiza S, et al. A hybrid variable neighborhood tabu search heuristic for the vehicle routing problem with multiple time windows. Computers and Operations Research (2013), http://dx.doi.org/10.1016/j.cor.2013.08.010

References

[1] Archetti C, Hertz A, Speranza MG. Metaheuristics for the team orienteering problem. Journal of Heuristics 2006;13:49–76.

[2] Belhaiza S, de Abreu NM, Hansen P, Oliveira CS. Variable neighborhood search for extremal graphs XI. Bounds on algebraic connectivity. In: Avis D, Hertz A, Marcotte O, editors. Graph theory and combinatorial optimization. New York: Springer; 2005. p. 1–16.

[3] Bozkaya B, Erkut E, Laporte G. A tabu search algorithm and adaptive memory procedure for political districting. European Journal of Operational Research 2003;144:12–26.

[4] Bräysy O. Fast local searches for the vehicle routing problems with time windows. INFOR 2002;40:319–30.

[5] Cordeau J-F, Laporte G, Mercier A. Improved tabu search algorithm for the handling of route duration constraints in vehicle routing problems with time windows. Journal of the Operational Research Society 2004;55:542–6.

[6] Doerner KF, Gronalt M, Hartl RF, Kichele G, Riemann M. Exact and heuristic algorithms for the vehicle routing problem with multiple interdependent time windows. Computers & Operations Research 2008;35:3034–48.

[7] Favaretto D, Moretti E, Pellegrini P. Ant colony system for a VRP with multiple time windows and multiple visits. Journal of Interdisciplinary Mathematics 2007;10:263–84.

[8] Fisher ML. Optimal solution of vehicle routing problems using minimum K-Trees. Operations Research 1994;42:626–42.

[9] Jarboui B, Derbel H, Hanafi S, Mladenović N. Variable neighborhood search for location routing. Computers & Operations Research 2013;40:47–57.

[10] Mladenović N, Hansen P. Variable neighborhood search. Computers & Operations Research 1997;24:1097–100.

[11] Mhallah R, Alkandari A, Mladenović N. Packing unit spheres into the smallest sphere using VNS and NLP. Computers & Operations Research 2013;40:603–15.

[12] Pesant G, Gendreau M, Potvin J-Y. An exact constraint programming algorithm for the traveling salesman problem with time windows. Transportation Science 1998;32:12–29.

[13] Pesant G, Gendreau M, Potvin J-Y, Rousseau L-M. On the flexibility of constraint programming models: from single to multiple time windows for the traveling salesman problem. European Journal of Operational Research 1999;117:253–63.

[14] Polacek M, Hartl RF, Doerner KF. A variable neighborhood search for the multi depot vehicle routing problem with time windows. Journal of Heuristics 2004;10:613–27.

[15] Rancourt M-È, Cordeau J-F, Laporte G. Long-haul vehicle routing and scheduling with working hour rules. Transportation Science 2013;47:81–107.

[16] Rochat Y, Taillard É-D. Probabilistic diversification and intensification in local search for vehicle routing. Journal of Heuristics 1995;1:147–67.

[17] Savelsbergh MWP. The vehicle routing problem with time windows: minimizing route duration. ORSA Journal on Computing 1992;4:146–54.

[18] Solomon MM. Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research 1987;35:254–65.

[19] Souffriau W, Vansteenwegen P, Vanden Berghe G, van Oudheusden D. The multiconstraint team orienteering problem with multiple time windows. Transportation Science 2013;47:53–63.

[20] Taillard É-D, Badeau P, Gendreau M, Guertin F, Potvin J-Y. A tabu search heuristic for the vehicle routing problem with soft time windows. Transportation Science 1997;31:170–86.

[21] Tricoire F, Romauch M, Doerner KF, Hartl RF. Heuristics for the multi-period orienteering problem with multiple time windows. Computers & Operations Research 2010;37:351–67.

Please cite this article as: Belhaiza S, et al. A hybrid variable neighborhood tabu search heuristic for the vehicle routing problem with multiple time windows. Computers and Operations Research (2013), http://dx.doi.org/10.1016/j.cor.2013.08.010