

# A Multi-Stage Very Large-Scale Neighborhood Search for the Vehicle Routing Problem with Soft Time-Windows

Sébastien Mouthuy<sup>1</sup>, Yves Deville<sup>1</sup> and Pascal Van Hentenryck<sup>2</sup>

<sup>1</sup> Université catholique de Louvain Department of Computing Science and Engineering  
Place Sainte-Barbe 2, 1348 Louvain-la-Neuve, Belgium  
{Sebastien.Mouthuy,Yves.Deville}@uclouvain.be

<sup>2</sup> Brown University, Box 1910 Providence, RI 02912, USA  
pvh@cs.brown.edu

## Abstract

This paper considers the VRP Problem with Soft Time-Windows (VRPSTW), a challenging routing problem due to its combination of hard time windows and a lexicographic objective function minimizing the number of vehicles, the violations of the soft time windows, and the total travel distance. The paper presents a multi-stage, variable neighborhood search algorithm for the VRPSTW, which uses the same very large-scale neighborhood (VLSN) for each of its three steps with different objective functions. Experimental results indicate that the multi-stage VLSN algorithm improves best-known solutions on 90% and 100% of the Type 1 and Type 3 instances respectively. Equally interesting is the fact that the multi-stage algorithm decreases the number of routes in 33% of the instances and the soft time-window violations in 92% of the remaining instances.

## 1 Introduction

This paper is concerned with Very Large-Scale Neighborhood (VLSN) search, a class of local-search algorithms whose neighborhoods contain an exponential number of neighbors. By considering neighborhoods of exponential size, VLSN search often produces local optima of higher quality than polynomial-sized neighborhoods. These exponential neighborhoods are obtained by considering, as neighbors, configurations that can be reached by a set or sequence of atomic moves.

VLSN algorithms have been successfully applied to a variety of NP-Hard problems such as Vehicle Routing Problems [10, 17], Capacitated Minimum Spanning Tree [4, 5], exam timetabling [3, 1, 2], quadratic assignment problem [9], block-to-train assignment [15]. See [6] for a survey of the main applications and VLSN approaches. VLSN search algorithms behave very well on these applications because there are many constraints that are hard to satisfy. The ability to perform many moves at once enables the search algorithm to improve solutions even though the constraints are tight.

This paper studies the application of VLSNs to the Vehicle Routing Problem with Soft Time-Windows (VRPSTW), a routing problem featuring hard and soft constraints and a complex objective function. The VRPSTW aims at scheduling several customer visits among a set of vehicles. For each customer, a hard and a soft time-windows are given. Each customer has to be visited within his hard time-window, while there is a penalty cost incurred for each customer not visited within his soft time-window. When only hard time-windows are considered, VLSN algorithms performs well on this problem [17]. However, when the penalty of the soft time-windows is added, a direct implementation of the VLSN algorithm exhibited poor performance.

To remedy this limitation, this paper presents a multi-stage VLSN algorithm, which uses the same very large-scale neighborhood but with different objective functions for each of the stage: Vehicle reduction, minimization of the soft time-window violations, and travel distance minimization. Moreover, each stage uses a variable neighborhood search (VNS) [13] to consider small sequences of customers first in the VLSN. The stage objective are especially designed to optimize their respective heuristics, thus overcoming the drawbacks of a “naive” approach.

Experimental results indicate that the multi-stage VLSN algorithm improves best-known solutions on 90% and 100% of the Type 1 and Type 3 instances respectively, which are standard benchmarks

for the VRPSTW. Equally interesting is the fact that the multi-stage algorithm decreases the number of routes in 33% of the instances and the soft time-window violations in 92% of the remaining instances.

The rest of this paper is organized as follows. Section 2 specifies the VRPSTW, including some delicate aspects of the time-window management. Section 3 gives a brief overview of basic concepts and introduces VLSN abstractly. Section 4 presents the VLSN neighborhood for the VRPSTW and Section 5 presents our multi-stage VLSN algorithm. Section 6 presents the experimental results.

## 2 Problem Definition

Consider a fleet of  $K$  identical **vehicles**:  $Vehicles = \{v_1, \dots, v_K\}$ . Each vehicle has a limited capacity  $Q$ . We also have a set of  $n$  **customers**  $Customers = \{1, \dots, n\}$  and a central depot  $D$ . For ease of notations, we identify the depot as  $K$  dummy sites:  $Depots = \{n+1, \dots, n+K\}$ . The set  $Sites = Depots \cup Customers$  identifies all the sites considered in this problem. The travel cost between sites  $i$  and  $j$  is denoted  $c_{ij}$ . Each customer  $i$  has a specific demand  $q_i$ .

A **route**  $r$  is a sequence of sites  $\langle r_0, \dots, r_k \rangle$  such that  $r_1, \dots, r_k \in Customers$  and  $r_0 \in Depots$ . Such a route represents a tour that starts from the depot, visits a sequence of customers (all distinct), and returns to the depot. The customers served by a route  $r$  is denoted by  $cust(r)$  and  $|r| = |cust(r)|$ . The sites visited by a route  $r$  is denoted by  $sites(r)$ . Given a route  $r = \langle r_0, r_1, \dots, r_k \rangle$ , we let the successor visit of visit  $r_i$  be  $r_{i+1}$ . As a route represents a tour from and to the depot, the successor of visit  $r_k$  is  $r_0$ . We define the subsequent visits of  $r_i$  as all the customers visited after  $r_i$ :  $\{r_{i+1}, \dots, r_{|r|}\}$ . A route  $r$  is consistent wrt the capacity constraint if  $\sum_{i \in cust(r)} q_i \leq Q$ .

A **routing**  $R$  is a collection of routes  $R = [r^{(1)}, \dots, r^{(K)}]$  assigning a consistent route  $r^{(k)}$  to each vehicle  $v_k \in Vehicles$  and such that the routes represent a partition of the set of customers. As a customer  $i$  is contained in exactly one route in a routing  $R = [r^{(1)}, \dots, r^{(K)}]$ , we define the successor  $i^+$  of visit  $i$  as the site visited right after  $i$ . A vehicle  $v_k$  is used if its route contains at least one customer ( $|cust(r^{(k)})| > 0$ ). We denote the number of used vehicles in a routing  $R$  by  $|R|$  and the total distance of  $R$  as  $dist(R) = \sum_{i \in Sites} c_{i,i^+}$ .

A **time-window** for customer  $i$  is an interval  $[e_i; l_i]$  where  $e_i$  is the earliest service time and  $l_i$  is the latest service time ( $e_i < l_i$ ). A time-window can be hard (the customer has to be served within his time-window) or soft (a unit penalty is incurred if the customer is not served within his time-window). There is a hard time-window  $[e_0; l_0]$  for the depot: It specifies the time horizon of the schedule. We are also given service duration  $d_i$  representing the duration of the visit at customer  $i$ . The feasibility and the penalty of the soft time-windows constraints of a routing  $R$  depends on the service time  $s_i$  which has to be assigned to each customer  $i$ . From the service times, we can derive the departure time of customer  $i$ :

$$\delta_i \geq 0 \quad \forall i \in Depots \quad (1)$$

$$\delta_i = s_i + d_i \quad \forall i \in Customers \quad (2)$$

Given a routing, because the vehicle must spend some time at visit  $i$  and must travel from this visit to its successor, the service times must satisfy the following constraints

$$\delta_i + c_{i,i^+} = s_{i^+} \quad \forall i \in Depots \quad (3)$$

$$\delta_i + c_{i,i^+} \leq s_{i^+} \quad \forall i \in Customers \quad (4)$$

Service times  $\{s_i | i \in Sites\}$  are said consistent if they respect constraints (1)-(4).

**Hard Time-windows** The availability of a customer  $i$  is specified by the time-window  $[e_i; l_i]$ . A vehicle can arrive at a site  $i$  before  $e_i$  but cannot wait more than  $w_{max}$  minutes before serving customer  $i$  (it should arrive at  $i$  after  $e_i - w_{max}$ ). The value of  $w_{max}$  is a parameter of the problem. For feasibility reasons, we assume that  $e_i \geq c_{0,i}$  for all  $i \in Customers$ . Given a routing  $R$ , the problem is to assign

consistent service times  $s_i$  for each customer  $i$  such that

$$e_i \leq s_i \leq l_i \quad (5)$$

$$s_i + d_i + c_{i,i+} \geq s_{i+} - w_{max} \quad (6)$$

Constraints (6) states that the vehicle does not have to wait more than  $w_{max}$  before serving a customer. A route  $r$  is consistent wrt the hard time-windows iff there exist consistent service times for all  $i \in cust(r)$  respecting constraints (5) and (6). A routing is consistent wrt the hard time-windows if all its route are consistent wrt this constraint.

Given a consistent routing  $R$ , the customers visited after customer  $i$  determine an earliest delivery time  $y_i$  for  $i$ : If the vehicle serves  $i$  before  $y_i$  then it must wait more than  $w_{max}$  minutes at one point in the route. These visits also determine a latest delivery time  $z_i$  for customer  $i$ : If the vehicle serves  $i$  later than  $z_i$ , then it is impossible to serve all the subsequent visits before their latest service times  $l_i$ . The earliest and latest delivery time are defined recursively as follows:

$$z_i = \begin{cases} l_i & \forall i \in Depots \\ \min(l_i, z_{i+} - c_{i,i+} - d_i) & \forall i \in Customers \end{cases} \quad (7)$$

$$y_i = \begin{cases} e_i & \forall i \in Depots \\ \max(e_i, y_{i+} - c_{i,i+} - d_i - w_{max}) & \forall i \in Customers \end{cases} \quad (8)$$

Notice that  $e_i \leq y_i \leq z_i \leq l_i$ . Thus serving a customer  $i$  in the interval  $[y_i, z_i]$  implies that its hard time-window is respected.

**Proposition 1.** *Given a route  $r = \langle r_0, \dots, r_K \rangle$  with  $y_i \leq z_i, \forall i \in sites(r)$ , serving a site  $r_k (0 \leq k \leq K)$  in the interval  $[y_{r_k}, z_{r_k}]$  allows to serve the subsequent customers  $i$  of  $r_k$  within their hard time-windows:*

$$\forall k = 0, \dots, K, \forall s_{r_k} \in [y_{r_k}, z_{r_k}], \exists \{s_{r_{k'}} \in [y_{r_{k'}}, z_{r_{k'}}] | k' = k + 1, \dots, K\} \text{ such that } \\ \{s_{r_k}, \dots, s_{r_K}\} \text{ are valid wrt the hard time-windows constraint}$$

**Corollary 1.** *Given a route  $r = \langle r_0, \dots, r_K \rangle$ ,*

- *$r$  is consistent wrt the hard time-windows constraint if and only if  $y_{r_0} \leq z_{r_0}$  and*
- *the following method determines consistent service times  $\{s_i | i \in cust(r)\}$  wrt the hard time-windows:*

1) *select a service time  $s_{r_1} \in [y_{r_1}, z_{r_1}]$ , and then*

2) *select service times  $s_{k+1}$  successively for  $k = 1, \dots, K - 1$  in the interval*

$$s_{r_{k+1}} \in [\max(y_{r_{k+1}}, s_{r_k} + d_{r_k} + c_{r_k, r_{k+1}}); \min(z_{r_{k+1}}, s_{r_k} + d_{r_k} + c_{r_k, r_{k+1}} + w_{max})].$$

**Soft time-windows** We are also given preferences for each customer about the time they would like to be served. These preferences are specified by soft time-windows  $[e_i^*; l_i^*]$  such that  $[e_i^*; l_i^*] \subseteq [e_i; l_i]$ . Given a routing, we define the latest preferred delivery time  $z_i^*$  as the latest time a vehicle can serve site  $i$  in order to be able to serve the subsequent visits of  $i$  within their preferred time-windows. We also define the earliest preferred delivery time  $y_i^*$ . The definitions of  $z_i^*$  and  $y_i^*$  are similar to the definitions (7) and (8).

Given a routing  $R$  and service times  $S = [s_i | i \in Sites]$ , the violation of the soft time-windows is defined as the number of customers who are not served within their soft time-windows:  $ViolStw(R, S) = |\{i \in Customers \mid s_i \notin [y_i^*, z_i^*]\}|$ . Notice that setting a service time  $s_i \notin [y_i^*, z_i^*]$  will induce at least one unit of penalty for subsequent visits of  $i$ .

The VRPSTW calls for a routing consistent wrt the capacity and hard time-windows constraints, and minimizing the lexicographical objective  $f(R, S) = \langle |R|, ViolStw(R, S), \sum_{i \in Sites} c_{i,i+} \rangle$ .

### 3 Preliminaries

**Combinatorial Optimization Problems** We use the following notations: let  $\mathcal{X} = [X_1, X_2, \dots, X_n]$  be a set of  $n$  variables which take their values in their domain  $D$ . We define an assignment as a function  $\sigma : \mathcal{X} \rightarrow D$  that assigns a value to each variable. We denote the set of all possible assignments as  $\Lambda$ . A constraint is a function  $\mathcal{C} : \Lambda \rightarrow \mathbb{N}$  giving the violation of a given assignment. A solution wrt constraint  $\mathcal{C}$  is an assignment  $\sigma$  such that  $\mathcal{C}(\sigma) = 0$ . An objective is a function  $f : \Lambda \rightarrow \mathbb{Z}$  giving the evaluation of the quality of a given assignment. Finally, we define a combinatorial optimization problem (COP) as the tuple  $\mathcal{P} = \langle f, \mathcal{C}, \mathcal{X}, D \rangle$ . Solving a COP requires finding a solution wrt constraint  $\mathcal{C}$  minimizing  $f$ .

**Local Moves** Local search algorithms are defined in terms of moves that modify an existing assignment. A move is thus a function  $m : \Lambda \rightarrow \Lambda$ . The set of possible moves is problem-dependent and is denoted  $\mathcal{M}$ . Computing the differentiation of a given move on the constraint and objective function is a central operation in local search. In this paper, we use  $\Delta_{\mathcal{C}}(m, \sigma)$  and  $\Delta_f(m, \sigma)$  to denote the changes induced by performing the move  $m$  on the violation of constraint  $\mathcal{C}$  and on the value of objective  $f$  given the current assignment  $\sigma$  for the decision variables. These values are defined as follows:  $\Delta_{\mathcal{C}}(m, \sigma) = \mathcal{C}(m(\sigma)) - \mathcal{C}(\sigma)$  and  $\Delta_f(m, \sigma) = f(m(\sigma)) - f(\sigma)$ .

**Very Large-Scale Neighborhoods** Very Large-Scale Neighborhood (VLSN) considers a constant set  $\mathcal{M}$  of atomic moves and, at each iteration, selects and apply a sequence  $M \subseteq \mathcal{M}$  of moves. As there is an exponential number of such sequences  $M \subseteq \mathcal{M}$ , there are exponentially many neighbors. To select a sequence of moves  $M$  respecting the constraint  $\mathcal{C}$ , it is important to view  $\mathcal{C}$  as the conjunction of two constraints  $\mathcal{C} = \mathcal{C}_1 \wedge \mathcal{C}_2$ . The constraint  $\mathcal{C}_1$  is a structural constraint, such as a partition or permutation constraint, and  $\mathcal{C}_2$  is the conjunction of all other side-constraints of the problem.  $\mathcal{C}_1$  is typically violated by simple local moves but the VLSN definition ensures that it is preserved by the selected sequence. Thus, our VLSN algorithms only consider the moves respecting  $\mathcal{C}_2$ , i.e., the moves  $M' = \{m \in \mathcal{M} | \mathcal{C}_2(m(\sigma)) = 0\}$ , and select a sequence  $M \subseteq M'$  whose impact on the current solution is such that (1) the structural constraint  $\mathcal{C}_1$  is respected, and (2) the objective function  $f$  is minimized. We denote  $M(\sigma) = m_1 \circ \dots \circ m_k(\sigma)$  where  $M = [m_1, \dots, m_k]$  is a sequence of moves in  $\mathcal{M}$ . We also define the differentiation of a sequence of moves:  $\Delta_{\mathcal{C}}(M, \sigma) = \mathcal{C}(M(\sigma)) - \mathcal{C}(\sigma)$  and  $\Delta_f(M, \sigma) = f(M(\sigma)) - f(\sigma)$ .

**Improvement Graphs** An improvement graph is an encoding of the neighborhood where atomic moves are edges. Moreover, a color is assigned to each node such that any color-disjoint cycle in the improvement graph represents a sequence of move which, when applied, satisfies the structural constraint  $\mathcal{C}_1$ . More formally, an improvement graph is a weighted colored label graph  $G(\sigma) = (V, E, w, \eta, \varphi)$ , where  $V$  is the set of nodes,  $E$  the set of edges,  $w_{ij}$  is the weight of the edge  $(i, j)$ ,  $\varphi : V \rightarrow \mathbb{N}$  is a function assigning a color to each node and  $\eta : E \rightarrow \mathcal{M}$  is a function assigning a move to each edge. The neighborhood, introduced in [18], can then be defined as

$$VLSN_{cycle}(G) = \left\{ \eta(C)(\sigma) \mid C \text{ is a color-disjoint cycle in } G(\sigma) \right\}$$

where  $\eta(C) = \{\eta((i, j)) \mid (i, j) \in C\}$ . Given a constraint  $\mathcal{C}_1$ , an improvement graph  $G$  is cycle-consistent wrt  $\mathcal{C}_1$  if the violations of  $\mathcal{C}_1$  are not affected by applying the moves of any color-disjoint cycle, i.e.,  $\forall C \in G : \mathcal{C}_1(\eta(C)(\sigma)) = \mathcal{C}_1(\sigma)$ . A cycle-consistent improvement graph for the VRPSTW is presented in the next section. An algorithm to compute color-disjoint cycles in graphs was presented in [18]: It builds a shortest-path tree rooted at a start node  $s \in V$  and ensure, as best as possible, that all the nodes of any path in this tree are color-disjoint. In order to obtain better solutions, this algorithm is repeated for different start nodes.

## 4 A VLSN Neighborhood for the VRPSTW

The VRP with hard and soft time-windows is modeled as the problem  $\mathcal{P}_{VRP} = \langle f, \mathcal{C}_{VRP} + \mathcal{C}_2, \mathcal{X}, D \rangle$ , where  $f$  is the objective function,  $\mathcal{C}_{VRP}$  is the structural constraint (whether the variables represent a routing), and  $\mathcal{C}_2$  captures the hard constraints to be respected by all routings.

**Variables** Let  $n$  be the number of customers to visit and  $K$  the number of available vehicles. We represent the depot by  $K$  dummy visits ranging from  $n + 1$  to  $n + K$ . A routing is a collection of  $K$  variables  $\mathcal{X} = [S_1, \dots, S_K]$  representing sequences. The domain  $D$  of these variables is the set of all possible sequences on the elements 1 to  $n + K$ . The number of elements in a sequence  $S$  is denoted  $|S|$ . For all  $i$  such that  $1 \leq i < |S|$ , we denote  $S[i]$  the element at the  $i^{th}$  position in the sequence  $S$ . The values  $y_i, z_i, z_i^*$  are derived from the routing and are not decision variables. We will show that service times  $s_i$  can also be derived from the routing automatically.

**Structural Constraint** In order for an assignment  $\sigma$  to represent a consistent routing, we enforce that (a) the first visit of any vehicle  $k$  is the dummy visit  $n + k$  representing the depot, and (b) the sequences represent a partition of the visits 1 to  $n + K$ . The structural global constraint  $\mathcal{C}_{VRP}$  satisfies  $\mathcal{C}_{VRP}(\sigma) = 0$  if and only if the conditions (a) and (b) are respected. A routing is an assignment  $\sigma$  such that  $\mathcal{C}_{VRP}(\sigma) = 0$ . We denote the position of element  $i$  in sequence  $S$  by  $rank(S, i)$  and the subsequence of length  $m$  beginning with  $i$  by  $S[i; m]$ . A subsequence  $S[i; m]$  is consistent if  $i \in S$  and  $rank(S, i) + m - 1 \leq |S|$ .

**Hard constraints** The hard constraint  $\mathcal{C}_2$  for the vehicle routing problem with hard/soft time-windows is  $\mathcal{C}_2 = \mathcal{C}_{capa} + \mathcal{C}_{Htw}$  where  $\mathcal{C}_{Htw}$  and  $\mathcal{C}_{capa}$  are the hard time-windows and capacity constraints respectively. The violations of these constraints are defined respectively  $\mathcal{C}_{Htw}(\sigma) = \sum_{i=1}^{n+K} \max(0, y_i - z_i)$  and  $\mathcal{C}_{capa}(\sigma) = \sum_{k=1}^K \max(0, \sum_{i \in S_k} q_i - Q)$ .

**Objectives** The objective function for this problem is  $f_{VRP}(\sigma) = \langle card(\sigma), \mathcal{C}_{Stw}(\sigma), dist(\sigma) \rangle$  where  $card$  is the function counting the number of vehicle used,  $\mathcal{C}_{Stw}$  is the violation of the soft time-windows and  $dist$  is the function giving the total distance of a routing. We define  $card(\sigma) = |\{S_k \in \mathcal{X} \mid |\sigma(S_k)| > 1\}|$  and  $dist(\sigma) = \sum_{i=1}^{n+K} c_{i,i^+}$ . To define the violations of the soft time-windows in a routing, we must define the service time  $s_i$  of customers. Our model serves customers as soon as possible while trying to minimize the violations of the soft time-windows constraint. Corollary 1 specifies that the soonest a customer  $i$  can be served is at time  $\max(y_i, s_{i^-} + d_{i^-} + d_{i^-, i})$ . Preferably, customer  $i$  should be served at time  $y_i^*$ . However, if  $z_i^* \leq y_i^*$ , at least one subsequent customer of  $i$  would be served outside his preferred time-window (it is impossible to achieve a zero-violation for the soft time-windows constraint). So, it is desirable to serve  $i$  at time  $z_i^*$ ; the preferred time-window of customer  $i$  will be violated, but maybe the preferred time-windows of all subsequent customers of  $i$  will be satisfied. So the preferred service time would be  $\min(y_i^*, z_i^*)$ . This preferred service time and the hard time-windows leads to the following definition of service times

$$s_i = \max(y_i, s_{i^-} + d_{i^-} + c_{i^-, i}, \min(y_i^*, z_i^*)), \forall i \in Customers$$

The violation of the soft time-windows then becomes  $\mathcal{C}_{Stw}(\sigma) = \sum_{i=1}^{n+K} \left( \max(0, s_i - l_i^*, e_i^* - s_i) > 0 \right)$ .

**Neighborhoods** The VLSNs in this paper are defined in terms of three moves. Let  $S_1 \neq S_2$  be two sequences,  $i \in S_1, j \in S_2$  and  $S_1[i; m], S_2[j; n]$  be two consistent subsequences. The moves are defined as follows:  $insert(S_1, i, m, S_2, j)$  inserts the subsequence  $S_1[i; m]$  in  $S_2$  right after  $j$ ,  $remove(S_2, j, n)$  removes the subsequence  $S_2[j; n]$  from  $S_2$  and  $replace(S_1, i, m, S_2, j, n, r)$  inserts  $S_1[i; m]$  in  $S_2$  at position  $r$  and then removes  $S_2[j; n]$  from  $S_2$  (with  $r \leq rank(S_2, j)$  or  $r \geq rank(S_2, j) + n$ ). Moreover, our VLSNs do not consider all  $insert$  and  $replace$  moves but only those minimizing the impact on the

objective  $f$ , i.e.,  $insert(S_1, i, m, S_2, f) \equiv insert(S_1, i, m, S_2, j)$  for the position  $j$  which minimizes  $\Delta_f(insert(S_1, i, m, S_2, j), \sigma)$  and  $replace(S_1, i, m, S_2, j, n, f) \equiv replace(S_1, i, m, S_2, j, n, r)$  for the rank  $r$  that minimizes  $\Delta_f(replace(S_1, i, m, S_2, j, n, r), \sigma)$ .

We are now in position to define an improvement graph for the VRPSTW that is cycle-consistent with the structural constraint  $C_{VRP}$ . The improvement graph is parametrized by an objective  $f$  and a parameter  $L$  specifying the length of the sequences considered by the moves.

**Definition 1.** Given an assignment  $\sigma$ , the improvement graph  $IG(f, L)(\sigma) = (V = V_1 \cup V_2, E, w, \eta, \varphi)$  is defined as follows:

- $V_1 = \{(i, m) | 1 \leq i \leq n, 1 \leq m \leq L, i \in Customers, \sigma(S)[i; m] \text{ is consistent with } i \in \sigma(S), S \in \mathcal{X}\}$
- $V_2 = \mathcal{X} = \{S_1, \dots, S_K\}$
- $E = \{(a, b) \in V \times V | a \in V_1 \vee b \in V_1 \wedge a, b \text{ correspond to different routes}\},$
- $\eta(a, b) = \begin{cases} replace(S_k, i, m, S_{k'}, j, n, f) & \text{if } a = (i, m), b = (j, n) \in V_1, i \in \sigma(S_k), j \in \sigma(S_{k'}) \\ insert(S_k, i, m, S_{k'}, f) & \text{if } a = (i, m) \in V_1, b = S_{k'} \in V_2, i \in \sigma(S_k), k \neq k' \\ remove(S, i, m) & \text{if } a = S_k \in V_2, b = (i, m) \in V_1, i \in \sigma(S) \end{cases}$
- $w_{ab} = \begin{cases} \Delta_f(\eta_{ab}, \sigma) & \text{if } \Delta_{C_2}(\eta(a, b), \sigma) = 0 \\ +\infty & \text{otherwise} \end{cases}$
- $\varphi(a) = \begin{cases} k & \text{if } a \in V_1 \text{ with } a = (i, m) \text{ and } i \in \sigma(S_k) \\ k' & \text{if } a \in V_2 \text{ with } a = S_{k'} \end{cases}$

Any color-disjoint cycle in this graph corresponds to moves whose application respects the structural vehicle routing constraint.

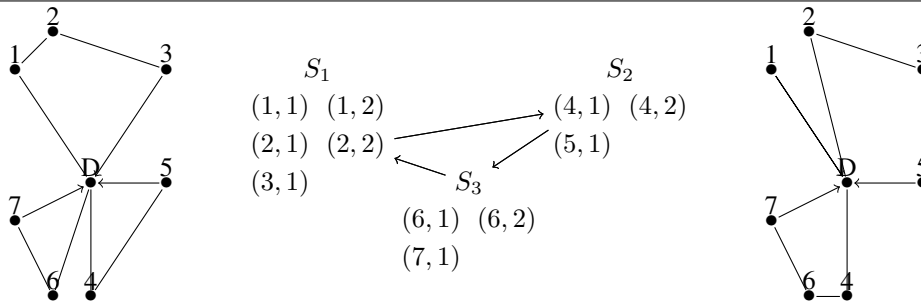
**Proposition 2.** The improvement graph  $IG(f, L)(\sigma)$  is cycle-consistent wrt the Vehicle Routing structure for any objective function  $f$  and any value of  $L$ .

An example of such improvement graph is illustrated in Example 1. The nodes represent either consistent subsequences  $S_k[i; m]$  of the current routing or entire routes. The moves *insert*, *remove* and *replace* are represented by edges.

Since the improvement graph is parametrized by  $f$  and  $L$ , it is possible to define a variety of neighborhoods of the form  $VLSN_{cycle}(IG(f, L)(\sigma))$  for specific objective function  $f$  and value of  $L$ . This feature is critical to our multi-stage algorithm for the VRPSTW.

## 5 A Multi-Stage, Variable Neighborhood Algorithm for the VRPSTW

This section describes a multi-stage, variable neighborhood search algorithm exploiting the parametrized VLSN neighborhood to minimize the lexicographical objective  $\langle card, C_{Stw}, dist \rangle$ . Multi-stage algorithms have become very successful in VRP problems (see, for instance, [8], as they adapt the search to the various parts of the objective). The overall structure of our algorithm is presented in Figure 1. Lines 1 to 8 reduce the number of vehicles, Line 10 minimizes the violations of the soft time windows, while Line 11 minimizes the overall distance. Each stage uses the VLSN neighborhood presented earlier with a different objective function.

**Example 1** Example of Improvement Graph.

Let  $n = 7$ ,  $\sigma$  such that  $\sigma(S_1) = [8, 1, 2, 3]$ ,  $\sigma(S_2) = [9, 4, 5]$ ,  $\sigma(S_3) = [10, 6, 7]$ . The routing described by  $\sigma$  is illustrated on the left. The improvement graph  $IG(dist, 2)(\sigma) = \langle V, E, \eta, w, \varphi \rangle$  is illustrated in the middle, where  $dist$  is the function of the overall distance of a routing. For sake of clarity, only the set of nodes  $V$  and three edges are represented. Each node is colored in function of the route it is in. The edge  $((2, 2), (4, 1))$  corresponds to the move  $replace(S_1, 2, 2, S_2, 4, 1, dist)$ . Similarly  $\eta(((4, 1), S_3)) = insert(S_2, 4, 1, S_3, dist)$  and  $\eta((S_3, (2, 2))) = remove(S_1, 2, 2)$ . The three nodes of this cycle are color-disjoint and the routing obtained (illustrated on the right) respects the structural constraint  $C_{VRP}$ . Notice that visit 4 is inserted right before visit 6 because this position minimizes  $dist$ .

```

1 repeat
2   select a non-empty vehicle  $k$  do
3     while the vehicle  $k$  is not empty ( $|S_k| > 1$ ) do
4       select a visit  $i \in S_k$  with the smallest hard time windows do
5         freeze( $S_k, i$ );
6         perform a best-improvement search over  $VLSN(\langle card, rm_i, mdl_i \rangle, L_{max})$ ;
7         unFreeze( $S_k$ );
8       if the attempt of removing visit  $i$  from  $S_k$  failed ( $i \in S_k$ ) then break;
9 until the last  $T$  attempts to empty a vehicle failed;
10 perform a first-improvement search over  $VLSN(\langle card, C_{Stw} \rangle, L_{max})$ ;
11 perform a best-improvement search over  $VLSN(\langle card, \alpha C_{Stw} + dist \rangle, L_{max})$ ;

```

**Algorithm 1:** A Multi-Stage VLSN Algorithm for the VRPSTW.

**Vehicle Reduction** To reduce the number of vehicles, the algorithm selects a vehicle and moves its visits one at a time. This process is repeated until no more vehicle can be emptied. To guide this VLSN search, the algorithm uses a lexicographic function with three elements as in [8]. In particular, given a vehicle  $k$ , we define the **minimal delay** of visit  $i \in S_{k'}$  ( $k \neq k'$ ) as the minimal violation of the hard time-windows constraint incurred by the insertion of visit  $i$  into vehicle  $k$

$$mdl_{k,i}(\sigma) = \min_{j \in S_k} \Delta_{C_{Htw}}(insert(S_{k'}, i, 1, S_k, j), \sigma).$$

We also define the total minimum delay wrt visit  $i$  by  $mdl_i(\sigma) = \sum_{k=1}^K h(mdl_{k,i})$  where  $h$  is a function favoring small values<sup>1</sup>. We also use the **binary ejection** of visit  $i \in \sigma(S_k)$  that favors solutions with  $i$  being moved to a different vehicle:  $rm_i(\sigma) = i \in \sigma(S_k)$ . The resulting objective function for this step then becomes  $\langle card, rm_i, mdl_i \rangle$ . Note that because it aims at removing visit  $i$  from vehicle  $S_k$ , the algorithm *freezes* all the other visits of this vehicle. The VLSN search stops as soon as visit  $i$  has been moved to another vehicle.

**Reducing the Soft Time-Windows Violations** To reduce the soft time-windows violations, the algorithm applies a first-improvement search over the neighborhood  $VLSN(\langle card, C_{Stw} \rangle, L_{max})$ .

<sup>1</sup>Our experiments use  $h(x) = \frac{C}{x} - \frac{x^2}{C'}$ , where  $C, C'$  are two constants.

**Distance Reduction** The overall distance is minimized by a best-improvement search over the neighborhood  $VLSN(\langle card, \alpha C_{Stw} + dist \rangle, L_{max})$ . Experimental results indicated that it was best to relax the soft time windows and penalize them in the objective function, instead of enforcing their best value found in the second step.

**Variable Neighborhood Search** In each of the stage, our search algorithm is augmented with a Variable Neighborhood Search (VNS) [13]. Since the size of the neighborhood  $VLSN_{cycle}(IG(f, L)(\sigma))$  increases with increasing  $L$ , the VNS starts with  $L = 1$  and increments  $L$  when no improving cycles exists. As soon as an improving cycle is found,  $L$  is reset to 1. The search stops if no neighbor has been found with  $L \leq L_{max}$ .

**Inner-Route Optimization** Before each VLSN search, the multi-stage algorithm also performs an inner-route optimization using a *reverse* move which, given a sequence  $S$  and two positions  $i$  and  $j$ , reverses the subsequence from  $i$  to  $j$ . This inner-route optimization is performed greedily until no improvement is possible.

## 6 Experimental Analysis

**Benchmarks** This section reports the results of our approach and compare them to other solution methods on several well-known benchmarks. These benchmarks were proposed by [7] and are variations of the Solomon instances (with 100 customers). For these instances, a time-window  $[a_i, b_i]$  is specified for each customer  $i$ . The depot is also constrained by the time-window  $[a_0, b_0]$ . Different definitions of the hard and soft time-windows have been investigated in the litterature. They were classified into six *types* in [12]. Our algorithm handles all these different definitions. From [12], types 1 and 3 are the the most studied and we focus on these types for space reasons. To our knowledge, Types 1 and 3 have been investigated in [16, 12, 11]. **Type 1** instances has been first introduced in [16]. A vehicle can arrive early at a customer  $i$  but cannot serve him before his earliest service time ( $e_i = e_i^* = a_i$ ). However, the vehicle can serve the customer later, with a penalty linear to the delay ( $l_i^* = b_i$ ). There is no hard latest service time for the customers ( $l_i = \infty$ ) but, for the depot, the time-window  $[a_0, b_0]$  is hard. The **type 3** instances were introduced in [7]. The time-window is considered as soft ( $e_i^* = a_i$  and  $l_i^* = b_i$ ). There is also a hard time-window for each customer  $i$ , requiring that each customer has to be served within a certain percentage  $p_{max}$  of the total route duration  $D = b_0 - a_0$ :  $e_i = a_i - D \frac{p_{max}}{100}$  and  $l_i = b_i + D \frac{p_{max}}{100}$ . Moreover, a vehicle is allowed to arrive earlier than  $e_i$  at a customer  $i$  but cannot wait more than  $w_{max}$  before serving the customer. The parameter  $w_{max}$  is also expressed as a percentage of the total route duration  $D$ :  $w_{max} = D \frac{W}{100}$ . Typical values of  $p_{max}$  and  $W$  are 0, 5 or 10.

**Experimental Results** The results are depicted in Table 1 and 2. The multi-stage algorithm was implemented in COMET [14]. The experiments were run on Intel Q6600 2,4GHz. We used a time limit of 250 seconds to reduce the number of vehicles and performed 200 iterations to reduce the violations of the soft time-windows constraint. We set  $L_{max} = 2$  as our preliminary experiments revealed this value achieved the best results. We compare the best solution obtained among 10 runs of our algorithm to the best published solutions from [16, 12, 11].

For type 1 instances (Table 1), the multi-stage algorithm found the best number of vehicles for all instances and achieved to decrease the number of vehicles for 9 instances. The multi-stage algorithms also found lower time-windows violations for 18 of the 20 instances. For Type 3 instances (Table 2), the multi-stage algorithm found significantly better solutions for all instances and decreased the number of vehicles on 4 instances.

These results indicate that our multi-stage algorithm provides significant improvements over the state-of-the-art, both in vehicle reduction and in minimizing the violations of the soft-constraints. For type 1 and type 3 respectively, it produces vehicle reductions in 45% and 25% of the instances and reduces the soft time-windows violations in 90% and 100% of the instances.

Instance	Best Published			VLSN			avg time(s)
	<i>card</i>	$\mathcal{C}_{Stw}$	<i>dist</i>	<i>card</i>	$\mathcal{C}_{Stw}$	<i>dist</i>	
R101	12	56	1129	<b>11</b>	39	1214	499
R102	11	46	1059	<b>10</b>	32	1172	523
R103	10	34	1027	<b>9</b>	24	1008	599
R104	9	<b>1</b>	984	9	7	977	624
R105	11	42	1074	<b>10</b>	31	1186	573
R106	10	33	1047	10	<b>19</b>	1091	640
R107	10	0	1148	<b>9</b>	17	1013	555
R108	9	<b>0</b>	979	9	4	1004	696
R109	10	28	1001	10	<b>12</b>	1132	629
R110	9	29	1013	9	<b>18</b>	1010	527
R111	10	0	1139	<b>9</b>	18	1016	635
R112	9	17	963	9	<b>5</b>	1000	644
RC101	11	44	1255	11	<b>22</b>	1373	496
RC102	10	32	1230	10	<b>20</b>	1261	473
RC103	10	25	1155	10	<b>7</b>	1214	567
RC104	10	0	1175	<b>9</b>	12	1129	638
RC105	11	38	1220	<b>10</b>	31	1333	501
RC106	10	27	1150	10	<b>13</b>	1300	493
RC107	10	28	1123	10	<b>8</b>	1259	609
RC108	10	0	1185	<b>9</b>	14	1126	620
20 instances	55%	10%		100%	90%		

Table 1: Experimental Results for the Type 1 Instances. Bold-face indicates Improvement.

## 7 Conclusions and Perspectives

This paper considered the VRP Problem with Soft Time-Windows (VRPSTW), a challenging routing problem due to its combination of hard time windows and a lexicographic objective function minimizing the number of vehicles, the violations of the soft time windows, and the total travel distance. The paper presented a multi-stage, variable neighborhood search algorithm for the VRPSTW, which uses the same very large-scale neighborhood (VLSN) for each of its three steps with different objective functions. In particular, each stage features a (lexicographic) objective adapted to its objective: Vehicle reduction, minimization of the soft time-constraint violations, and minimization of the travel distance. In addition, the multi-stage algorithm applies a variable neighborhood search to focus first on short sequences on customers. Experimental results indicate that the multi-stage VLSN algorithm improves best-known solution on 90% and 100% of the Type 1 and Type 3 instances respectively. Equally interesting is the fact that the multi-stage algorithm decreases the number of routes in 33% of the instances and the soft time-window violations in 92% of the remaining instances.

This research emerged from an attempt to automate the implementation of VLSN algorithms in order to make them easier to apply and to extend. Our current focus is to study the theoretical foundations and the implementation of a VLSN framework that would simplify the design and implementation of these algorithms on complex problems such as the VRPSTW. This framework would then allow us to determine if sophisticated VLSN algorithms can also be successful on other complex routing and scheduling applications.

**Acknowledgements** The first author is supported by the belgian FNRS ('Aspirant'). This research is partially supported by the Interuniversity Attraction Poles Programme (Belgian State, Belgian Science Policy) and the FRFC project 2.4504.10 of the Belgian FNRS (National Fund for Scientific Research).

## References

- [1] Salwani Abdullah, Samad Ahmadi, Edmund Burke, and Moshe Dror. Applying ahuja-orlin's large neighborhood for constructing examination timetabling solution. In *Proceedings of the Fifth International Conference on the Practice and Theory of Automated Timetabling*, number 3616 in Lecture Notes in Computer Science, pages 413–420. Springer, 2004.

	Instance	Best Published			VLSN			
		<i>card</i>	$C_{Stw}$	<i>dist</i>	<i>card</i>	$C_{Stw}$	<i>dist</i>	<i>avgtime(s)</i>
$p_{max} = 5\%$	R101	14	32	1633	<b>13</b>	58	1582	526
	R102	12	37	1404	12	<b>20</b>	1364	610
	R103	11	7	1374	<b>10</b>	19	1184	529
	R109	11	7	1393	<b>10</b>	22	1211	521
	RC101	13	7	1778	<b>12</b>	33	1586	425
	RC102	11	42	1375	11	<b>19</b>	1556	451
	RC103	10	17	1256	10	<b>9</b>	1236	511
	RC106	11	19	1336	11	<b>7</b>	1451	424
$p_{max} = 10\%$	R101	12	69	1376	12	<b>53</b>	1349	495
	R102	10	67	1173	10	<b>62</b>	1259	554
	R103	10	24	1274	10	<b>16</b>	1170	559
	R109	10	47	1116	10	<b>20</b>	1175	522
	RC101	11	57	1322	11	<b>36</b>	1512	466
	RC102	11	26	1367	11	<b>13</b>	1443	489
	RC103	10	15	1228	10	<b>9</b>	1221	464
	RC106	10	51	1160	10	<b>30</b>	1331	540
16 instances	0%	0%		100%	100%			

Table 2: Experimental Results for the Type 3 Instances. Bold-face indicates Improvement.

- [2] Salwani Abdullah, Samad Ahmadi, Edmund Burke, and Moshe Dror. Investigating ahuja-orlin’s large neighbourhood search approach for examination timetabling. *OR Spectrum*, 29:351–372, April 2007.
- [3] Salwani Abdullah, Samad Ahmadi, Edmund Burke, Moshe Dror, and Barry McCollum. A tabu-based large neighbourhood search methodology for the capacitated examination timetabling problem. *Journal of the Operational Research Society*, 58:1494–1502, November 2007.
- [4] Ravindra K. Ahuja, James B. Orlin, and Dushyant Sharma. Multi-exchange neighborhood structures for the capacitated minimum spanning tree problem. *Mathematical Programming*, 91:71–97, October 2001.
- [5] Ravindra K. Ahuja, James B. Orlin, and Dushyant Sharma. A composite very large-scale neighborhood structure for the capacitated minimum spanning tree problem. *Operations Research Letters*, 31:185–194, May 2003.
- [6] Ravindra K. Ahuja, Özlem Ergun, James B. Orlin, and Abraham P. Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Appl. Math.*, 123(1-3):75–102, 2002.
- [7] Nagraj Balakrishnan. Simple heuristics for the vehicle routing problem with soft time windows. *The Journal of the Operational Research Society*, 44(3):279–287, March 1993.
- [8] Russell Bent and Pascal Van Hentenryck. A two-stage hybrid local search for the vehicle routing problem with time windows. *Transportation Science*, 38:515–530, 2004.
- [9] Vladimir G Deineko and Gerhard J Woeginger. A study of exponential neighborhoods for the travelling salesman problem and for the quadratic assignment problem. *Mathematical Programming*, 87:519–542, 2000.
- [10] Ozlem Ergun, James B. Orlin, and Abran Steele-Feldman. Creating very large scale neighborhoods out of smaller ones by compounding moves: A study on the vehicle routing problem. Technical Report 4393-02, MIT Sloan School of Management, oct 2002.
- [11] Miguel Andres Figliozzi. An iterative route construction and improvement algorithm for the vehicle routing problem with soft time windows. *Transportation Research Part C: Emerging Technologies*, In Press, Corrected Proof, 2009.
- [12] Zhuo Fu, Richard Eglese, and Leon Li. A unified tabu search algorithm for vehicle routing problems with soft time windows. *Journal of the Operational Research Society*, 59:663–673, May 2008.
- [13] Pierre Hansen and Nenad Mladenovic. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467, May 2001.
- [14] Pascal Van Hentenryck and Laurent Michel. *Constraint-Based Local Search*. MIT Press, October 2005.
- [15] Krishna C. Jha, Ravindra K. Ahuja, and Güvenç Şahin. New approaches for solving the block-to-train assignment problem. *Networks*, 51(1):48–62, 2008.
- [16] Éric Taillard, Philippe Badeau, Michel Gendreau, François Guertin, and Jean-Yves Potvin. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation science*, 31(2):170–186, 1997.
- [17] Paul M. Thompson and Harilaos N. Psaraftis. Cyclic transfer algorithms for multivehicle routing and scheduling problems. *Operations Research*, 41(5):935–946, sep 1993.
- [18] Paul Michael Thompson and James B. Orlin. The theory of cyclic transfers. Technical Report OR 200-89, Massachusetts Institute of Technology, Operations Research Center, 1989.