CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

A Tabu Search for the Time-Dependent Multi-Zone Multi-Trip Vehicle Routing Problem with Time Windows

Phuong Nguyen Khanh
Teodor Gabriel Crainic
Michel Toulouse

August 2012

CIRRELT-2012-44

Bureaux de Montréal :
Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

A Tabu Search for the Time-Dependent Multi-Zone Multi-Trip Vehicle Routing Problem with Time Windows

Phuong Nguyen Khanh[1,2], Teodor Gabriel Crainic[1,3,*], Michel Toulouse[1,4]

[1] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)

[2] Department of Computer Science and Operations Research, Université de Montréal, P.O. Box 6128, Station Centre-Ville, Montréal, Canada H3C 3J7

[3] Department of Management and Technology, Université du Québec à Montréal, P.O. Box 8888, Station Centre-Ville, Montréal, Canada H3C 3P8

[4] Department of Computer Science, Oklahoma State University, 700 North Greenwood Avenue, North Hall 328, Tulsa, OK 74106-0700, USA

Abstract. We propose a tabu search meta-heuristic for the Time dependent Multi-zone Multi-trip Vehicle Routing Problem with Time Windows. Two types of neighborhoods, corresponding to two decision sets of the problem, together with a strategy controlling the selection of the neighborhood type for particular phases of the search, provide the means to set up and combine exploration and exploitation capabilities for the search. A diversication strategy, guided by an elite solution set and a frequency-based memory, is also used to drive the search to potentially unexplored good regions and, hopefully, enhance the solution quality. Extensive numerical experiments and comparisons with the literature show that the proposed tabu search yields very high quality solutions, improving those currently published.

Keywords. Vehicle routing, time windows, multi-zone, multi-trip, synchronization, meta-heuristics, tabu search, multiple neighborhoods, elite set, frequency-based memory.

Acknowledgements. Partial funding for this project has been provided by the Natural Sciences and Engineering Research Council of Canada (NSERC), through its Industrial Research Chair and Discovery Grant programs, by the Université de Montréal, and by the EMME/2-STAN Royalty Research Funds (Dr. Heinz Spiess). We also gratefully acknowledge the support of Fonds de recherche du Québec - Nature et technologies (FRQNT) through their infrastructure grants and of Calcul Québec and Compute Canada through access to their high-performance computing infrastructure.

---

* Corresponding author: TeodorGabriel.Crainic@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2012

1 Introduction

The basic Vehicle Routing Problem with Time Windows (VRPTW) aims to design least cost routes from a single depot to a set of geographically distributed customers, while satisfying time window constraints at customers and the capacity of vehicles. In this paper, we consider the *Time-dependent Multi-zone Multi-trip Vehicle Routing Problem with Time Windows* (*TMZT-VRPTW*), which is an extension of the VRPTW involving both designing and assigning routes to vehicles within time synchronization restrictions.

The TMZT-VRPTW is encountered in several settings, in particular in the context of planning the operations of two-tiered City Logistics systems. In such systems, incoming loads are first sorted and consolidated at a first-tier facility, an intermodal platform or urban distribution center located on the outskirts of the city, moved to a second-tier facility, satellite or *supply point*, by a fleet of first-tier vehicles, where they are transferred to smaller-capacity vehicles for final delivery to customers situated within the controlled city area. Activities take place over a time horizon several hours long, vehicles, second-tier ones in particular, performing multiple tours before returning to the depots. Most importantly, the location and limited capacity of most supply points, together with the need for efficient operations and on-time delivery to customers, induce the need for transdock load-transfer activities and the synchronization of the operations of first and second-tier vehicles, very short waiting times being allowed for vehicles at supply points. Planning determines the customer demands to service out of each supply point at each time period. When the set is non empty, it is called thereafter the supply-point zone. A second-tier vehicle then arrives at a supply point at an appointed time, meets the first-tier vehicles bringing in the demands for the zone, and loads the planned freight. It then performs a tour servicing its designated customer demands within the zone. Once the last customer is serviced, the vehicle moves either directly to a supply point for its next tour, the preferred move, or to a waiting station (when available) to wait for its next appointment, or to the depot to end the current activity period.

The TMZT-VRPTW addresses this routing problem, where a fleet of homogeneous vehicles operating out of a single depot, perform multi-tour routes to service customers located in multiple (space, time) zones. The time-dependent customer demands are available at designated supply points within hard time windows, and must be delivered at customers within soft time windows. Waiting stations may be used by the vehicles to wait for the next appointment. The goal is to determine the set of vehicle routes providing timely customer service and scheduled arrival at supply points for loading freight, minimizing the total cost made up of the (variable) costs of operating vehicles and the (fixed) costs of using them.

To our knowledge, Crainic et al. (2009) were the first (and only) to propose a method to address the TMZT-VRPTW. The authors proposed a decomposition approach in which customer-zone routing out of each supply point subproblems are solved indepen-

dently. The created vehicle tours are then put together into multi-tour routes by solving a minimum cost network flow problem. Yet, as routing decisions affect the supply point assignment decisions and vice versa, these two decision levels are intertwined and should not be solved separately. This paper aims to address this issue and investigate an alternative approach that addresses the two decisions simultaneously, in a comprehensive way.

We introduce the first tabu search for the TMZT-VRPTW in which multiple neighborhoods are used to improve both the routing and the assignment of routes to vehicles. These neighborhoods belong to two types, one for vehicle-to-supply-point assignments and the other for customer service (routing), corresponding to the two decision levels of the problem, where vehicles are assigned to supply points (several supply points may be assigned to the same vehicle) at the high level, while routes are created by assigning customers to vehicles at the low level. Vehicle-to-supply-point assignment moves perturb significantly the solution and thus favor exploration of the search space, while routing moves applied to each vehicle tour exploit good assignments. Hence, dynamically adjusting their utilization during the search provides the proposed algorithm with a powerful combination of exploration and exploitation capability.

The proposed algorithm starts by freely exploring the search space made up of feasible and unfeasible solutions. As the search advances, one lowers the probability of selecting vehicle-to-supply-point assignment neighborhoods, thus limiting the size of the search region and giving routing moves more time to fully optimize routes. Of course, customer time windows and synchronization restrictions at supply points constrain these decisions and moves. A diversification strategy guided by an elite set and a frequency-based memory is called upon when the search begins to stagnate. Creating new working solutions from the elite set helps to capitalize on the best solution attributes obtained so far. On the other hand, employing a frequency-based memory to perturb new working solutions provides a certain level of diversity to the search.

The main contributions of the papers are, 1) a new formulation for the TMZT-VRPTW, which is the source to define neighborhoods in the proposed tabu search; 2) the neighborhood structure and the dynamic strategy used to control the selection of neighborhoods; 3) a new tabu search meta-heuristic outperforming the available method (Crainic et al., 2012) with new best-known solutions on all instances and an improvement in the solution quality by 4.38% on average.

The remainder of the paper is organized as follows. Section 2 contains the detailed problem description. The formulation is then given in Section 3. Section 4 reviews the literature. The details of the proposed methodology are described in Section 5. Computational results are then reported and analyzed in Section 6, while conclusions and future works are considered in Section 7.

2 Problem Description

Synchronization at supply points requires that vehicles arrive at supply points at appointed times. Consequently, a direct move that gets the vehicle to a supply point sooner than the appointed time is forbidden. In this case, the vehicle may go to a location, which we call *waiting station*, and wait there in order to get to its next supply point just before the appointed time. Otherwise, if there is no waiting station available, the vehicle goes to the main depot to finish its task. The system may provide physical waiting stations, and then vehicles select the best waiting stations to park, if necessary. On the other hand, waiting stations could be associated to supply points (when physically feasible), which means allowing vehicles to wait at supply points and, possibly, incur a penalty cost. We adopt the former definition in this paper.

The TMZT-VRPTW can then be described as follows. There is a main depot g, a set of waiting stations $w \in \mathcal{W}$, and a set of supply points $s \in \mathcal{S}$. We assume that there is a limited allowable waiting time, defined by δ, available at each supply point. Hence, the vehicle can arrive at a supply point a bit sooner than the appointed period. Each supply point $s \in \mathcal{S}$ has an opening time window $[t(s) - \delta, t(s)]$, specifying the earliest and latest times the vehicle may to be available at s, respectively, and a loading time $\delta(s)$, which is the time required to load freight to service a set of customers D_s. Thus, customers are divided into multiple zones, each zone belonging to one supply point. Each customer $d \in D_s$ has a demand q_d, a service time $\delta(d)$, a time window $[e_d, l_d]$, where e_d is the earliest time service may begin and l_d is the latest time.

The TMZT-VRPTW can be seen as the problem of determining a set of routes and an assignment of each route to one vehicle, such that each vehicle can perform several routes sequentially. The objective is to minimize the total cost, which is comprised of routing cost and fixed cost on the use of vehicles, while the following conditions are satisfied:

1. Every vehicle starts and ends its route sequence at the main depot g;
2. Every vehicle required to service customers in D_s must reach its supply point $s \in \mathcal{S}$ within its opening time window; These are hard time windows, i.e., a vehicle must not arrive to s sooner than $(t(s) - \delta)$ and no later than $t(s)$; in the former case, the vehicle has to wait at a waiting station $w \in \mathcal{W}$ before moving to s; Once at s, the vehicle starts loading goods at time $t(s)$ and continues loading for a time $\delta(s)$, after which it leaves s to service customers in D_s. After performing a route within customer zone D_s, the vehicle may move to another customer zone for the next trip or go to the main depot g to complete its task.
3. Every customer $d \in \cup_{s \in \mathcal{S}} D_s$ is visited by exactly one route with a total load not exceeding Q, and is serviced within its time window; these are soft, i.e., a vehicle may arrive before e_d and wait to begin service.

3 Model Formulation

The TMZT-VRPTW is defined on a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, with vertex set $\mathcal{V} = \{g \cup \mathcal{S} \cup \mathcal{D} \cup \mathcal{W}\}$, where g is the main depot, $\mathcal{S}$ is the set of supply points, $\mathcal{D} = \{\cup D_s : s \in \mathcal{S}\}$ is the customer set, $\mathcal{W}$ is the set of waiting stations, and the arc set $\mathcal{A} = \{(g,s) : s \in \mathcal{S}\} \cup \{(s,d) : s \in \mathcal{S}, d \in D_s\} \cup \{(d,j) : d \in \mathcal{D}, j \in g \cup \mathcal{W}\} \cup \{(i,j) : i,j \in D_s, s \in \mathcal{S}\} \cup \{(d,s') : d \in D_s, s, s' \in \mathcal{S}, t(s) < t(s')\} \cup \{(w,s) : w \in \mathcal{W}, s \in \mathcal{S}\}$. Arcs representing direct travel from the main depot g to any customer or to waiting station, from any customer to its supply point or to supply points with opening time earlier than that of its supply point, from a supply point to any waiting station or to the main depot g are not included in $\mathcal{A}$. A routing cost (or travel time) c_{ij} is associated with each arc $(i,j) \in \mathcal{A}$. A fleet of m identical vehicles with capacity Q is based at the main depot g. Vehicles are grouped into set $\mathcal{K}$.

Let a ***route leg*** be a route that links a pair of supply points, or starts and ends at a supply point and the main depot g, respectively. Thus, there are two types of route legs and their feasibility is defined as follows:

- A ***single-supply point route leg*** l, starting at a supply point s and ending at the main depot, is feasible if it starts loading goods at supply point s at time $t(s)$ with a total load not exceeding Q, then leaves s at time $t(s) + \delta(s)$ to perform the delivery for serving a subset of customers in D_s within their time windows.
- An ***inter-supply point route leg*** l that starts and ends at a pair of supply points s and s', respectively, is feasible if it starts loading goods at supply point s at time $t(s)$ with a total load not exceeding Q, then leaves s at time $t(s) + \delta(s)$ to perform the delivery for serving a subset of customers in D_s within their time windows, and arrives to s' within the opening time window $[t(s') - \delta, t(s')]$ (the vehicle can wait at a waiting station $w \in \mathcal{W}$ before moving to s' in case the direct move from the last serviced customer in leg l to s' gets the vehicle to s' before $(t(s') - \delta)$).

A sequence of route legs, starting and ending at the main depot, assigned to a vehicle is called a ***work assignment***. For the sake of simplicity, from now on, the terms *vehicle* and *work assignment* are used interchangeably. Figure 1 illustrates a three-leg work assignment, where s_1, s_2, s_3 are supply points, g and w_1 are respectively the main depot and waiting station, $D_{s_1} = \{d_1, d_2, d_3, d_4, d_5\}$, $D_{s_2} = \{d_6, d_7, d_8, d_9\}$, and $D_{s_3} = \{d_{10}, d_{11}, d_{12}, d_{13}, d_{14}, d_{15}\}$. The dashed lines stand for the empty arrival from the depot g or from a waiting station, the empty movement from the last customer in the previous leg to the supply point of the next leg or to a waiting station, and the empty movement to the depot g once the work assignment is finished. This work assignment consists of a sequence of three legs $\{l_1, l_2, l_3\}$, where $l_1 = \{s_1, d_1, d_4, d_3, s_2\}$ and $l_2 = \{s_2, d_7, d_9, w_1, s_3\}$

are two inter-supply point route legs, while $l_3 = \{s_3, d_{11}, d_{14}, d_{12}, d_{15}, g\}$ is a single-supply point route leg.

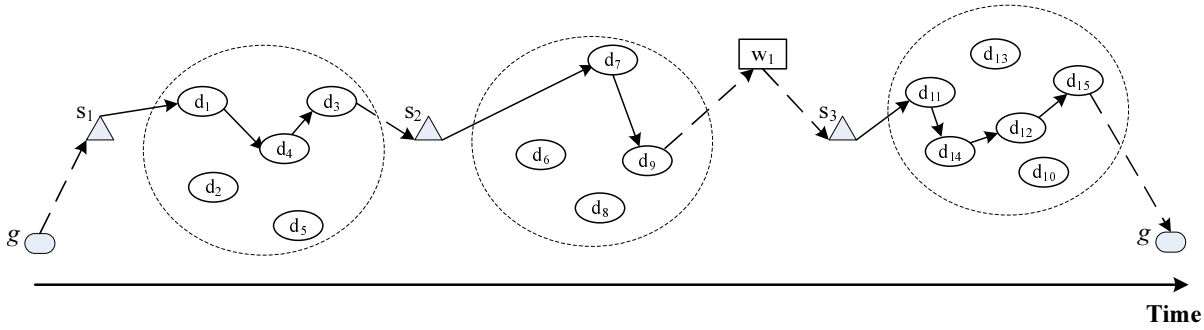


Figure 1: A three-leg work assignment illustration

Let $\mathcal{L}$ denote the set of all feasible legs satisfying the total load of vehicles, and the time windows at customers and supply points. Define the e_{dl} and f_{ls} coefficients:

$$e_{dl} = \begin{cases} 1 & \text{if customer } d \in D \text{ is on leg } l \in \mathcal{L}; \\ 0 & \text{otherwise;} \end{cases}$$

$$f_{ls} = \begin{cases} 1 & \text{if leg } l \text{ starts at supply point } s \in \mathcal{S}; \\ -1 & \text{if leg } l \text{ ends at supply point } s \in \mathcal{S}; \\ 0 & \text{otherwise;} \end{cases}$$

Binary decision variables are used in the formulation:

- $x_l^k = \begin{cases} 1 & \text{if leg } l \in \mathcal{L} \text{ is assigned to work assignment } k \in \mathcal{K}, \\ 0 & \text{otherwise} \end{cases}$
- $y_s^k = \begin{cases} 1 & \text{if work assignment } k \text{ has the first leg starting at supply point } s, \\ 0 & \text{otherwise} \end{cases}$
- $z_s^k = \begin{cases} 1 & \text{if work assignment } k \text{ has the last leg starting at supply point } s, \\ 0 & \text{otherwise} \end{cases}$

Let π_l be the total cost of route leg l, and F be the fixed cost of using a vehicle. The TMZT-VRPTW can then be formulated as

$$\text{Minimize} \sum_{k \in \mathcal{K}} \sum_{l \in \mathcal{L}} \pi_l x_l^k + \sum_{k \in \mathcal{K}} F \sum_{s \in \mathcal{S}} y_s^k \tag{1}$$

$$\text{S.t.} \sum_{k\in\mathcal{K}}\sum_{l\in\mathcal{L}} e_{dl}x_l^k = 1 \quad \forall d \in \mathcal{D}, \tag{2}$$

$$\sum_{s\in\mathcal{S}} y_s^k \leq 1 \quad \forall k \in \mathcal{K}, \tag{3}$$

$$\sum_{s\in\mathcal{S}} y_s^k = \sum_{s\in\mathcal{S}} z_s^k \quad \forall k \in \mathcal{K}, \tag{4}$$

$$\sum_{l\in\mathcal{L}} f_{ls}x_l^k = y_s^k \quad \forall s \in \mathcal{S}, k \in \mathcal{K}, \tag{5}$$

$$x_l^k \in \{0,1\} \quad \forall l \in \mathcal{L}, k \in \mathcal{K}, \tag{6}$$

$$y_s^k \in \{0,1\} \quad \forall s \in \mathcal{S}, k \in \mathcal{K}, \tag{7}$$

$$z_s^k \in \{0,1\} \quad \forall s \in \mathcal{S}, k \in \mathcal{K} \tag{8}$$

The objective function (1) minimizes the total cost made up of the costs of operating and using vehicles. Constraints (2) guarantee that each customer is visited exactly once, while constraints (3) state that at most one work assignment is assigned to every vehicle. Constraints (4) ensure that each work assignment starts and ends at the main depot. In fact, by summing over all supply points, the left-hand side counts the number of first legs assigned to each vehicle k, while the right-hand side counts the number of last legs assigned to each vehicle k. Then, from constraints (3) and (4), either the numbers of first and last legs assigned to the vehicle k are both equal to zero, or, equivalently, the vehicle k is not used ($\sum_{s\in\mathcal{S}} y_s^k = \sum_{s\in\mathcal{S}} z_s^k = 0$), or are both equal to 1, or, equivalently, the vehicle k is used ($\sum_{s\in\mathcal{S}} y_s^k = \sum_{s\in\mathcal{S}} z_s^k = 1$).

Constraints (5) ensure that when a vehicle goes to a supply point, it also leaves it, except for the starting supply point of the first leg. In fact, for any given vehicle k and supply point s, the left-hand side of the equality sums the value of f_{ls} on all legs starting or ending at supply point s which are assigned to vehicle k. Consequently, when $y_s^k = 1$, the equality (5) becomes $\sum_{l\in\mathcal{L}} f_{ls}x_l^k = 1$, which means that there must be one leg starting at supply point s assigned to vehicle k as the first leg. Constraints (6), (7), and (8) define the sets of decision variables.

4 Literature review

The literature on TMZT-VRPTW is limited. In the TMZT-VRPTW, customer demands are divided into a number of groups with associated supply points and time periods. More precisely, given a (supply point, period) zone, we have a known set of customer demands that need to be serviced. Taking advantage of this special structure, Crainic et al. (2009) proposed a decomposition-based heuristic approach for TMZT-VRPTW,

but no implementation was reported. The general idea is to decompose the problem by (supply point, period) zone, solve the resulting small VRPTW at each zone, and finally determine the flow of vehicles to operate the routes associated with these zones at minimum cost by solving a minimum cost network flow problem. Crainic et al. (2012) later implemented this idea, and calculated a lower bound by relaxing vehicle capacity and time window constraints at supply points and customers.

A number of VRP variants share the multi-trip setting with the TMZT-VRPTW, e.g., the Multi-trip Vehicle Routing Problem (or the Vehicle Routing Problem with Multiple Use of Vehicles; Taillard et al., 1995; Brandão and Mercer, 1998; Petch and Salhi, 2003; Salhi and Petch, 2007), the VRP with Intermediate Facilities or with inter-depot routes (Tarantilis et al., 2008; Crevier et al., 2007) and the Waste Collection VRP (Kim et al., 2006; Ombuki-Berman et al., 2007; Benjamin and Beasley, 2010). In the first variant, only one depot is used to replenish vehicles between their trips, while in the latter variants, vehicles may be replenished at intermediate depots along their trips. In addition, unlike the first two variants, each driver is assumed to take a lunch break in a period of time in the Waste Collection VRP. The challenging setting in our problem compared to these three variants is the time synchronization restrictions at supply points, which then introduces the waiting stations. Thus, each time a vehicle undergoes its trip, it has to consider the hard time window of the supply point where it will go next. The vehicle must then go to a waiting station to wait if arriving too early to the supply point.

The School Bus Routing problem (SBRP) resembles our problem setting quite closely. In general, the SBRP involves transporting students from predefined locations to their schools using a fleet of buses with varying capacity, while satisfying all timing requirements of the schools. The SBRP consists of three components: determine the bus stop locations, assign students to bus stops, route and schedule the buses. However, most of the problems described in the literature just consider some parts of the SBRP. In the multi-school setting, the SBRP shares some constraint settings with the TMZT-VRPTW, e.g., vehicle capacity, school time window, multi-trip. Yet, there are also differences in conditions and settings between the SBRP and the TMZT-VRPTW. In the SBRP, a mixed-loads setting may occur in which students from different schools can be put on the same bus at the same time, as well as maximum riding times for students on buses. These settings are not imposed in the TMZT-VRPTW. In the SBRP, only the exact earliest pick-up time for all students is considered, while there is a time window for each customer in the TMZT-VRPTW.

5 Tabu search meta-heuristic

Among the meta-heuristics proposed for the vehicle routing problem, tabu search has been shown to be very effective, providing a good compromise between solution qual-

ity and computation time. Various techniques have also been proposed to enhance the performance of tabu search to tackle complex problems with multiple constraints and characteristics. Following this trend, a tabu search with multiple neighborhoods and appropriate memory mechanisms was designed for the TMZT-VRPTW. In this section, we present our tabu search algorithm, starting with the description of its general structure, then detailing its main components. The search space and initial solution generator are presented in Sections 5.2 and 5.3, respectively. Sections 5.4 - 5.6 describe the neighborhood structures, the neighborhood-selection *Control* procedure, and the tabu status mechanism for each neighborhood. We detail in Section 5.7 the elite set management and the *Diversification* mechanism. Finally, the post optimization procedure is given in Section 5.8.

5.1 General structure

The TMZT-VRPTW schedules vehicles from the main depot to supply points in order to load freight. The vehicles then deliver freight from supply points to customers. In order to address this problem, we have to deal with two types of decision: the first assigns vehicles to supply points, and the second assigns customers to vehicles. Consequently, we define different neighborhood structures based on these two types of decisions. The *leg neighborhoods* aim to change the vehicle assignments to supply points, while *routing neighborhoods* move customers among vehicle routes.

Various studies on tabu search use multiple neighborhoods. All neighborhoods are evaluated simultaneously in a number of studies (e.g., Dell'Amico and Trubian, 1993; Gaspero and Schaerf, 2007), while neighborhoods are explored in serial fashion, one after another in either a fixed or randomized order, in a number of other ones (e.g. Xu et al., 2006; Hamiez et al., 2009). Experiments show that these approaches do not work well for the TMZT-VRPTW (see details in Annex A). The most important reason is that, in our problem, each type of neighborhood works on a particular part of the decision domain of the problem, whereas size is the main difference between neighborhoods in the literature. Moreover, the two decision domain parts are related and their explorations impact each other. Consequently, we propose an approach that not only guides the selection of the neighborhood type, but also balances the workload within each part of the decision domain. The selection of neighborhood types in the proposed tabu search is driven based on the variation of solution quality. More precisely, we use a parameter r to specify *the ratio of selecting routing neighborhoods to leg neighborhoods*, and adjust the value of this parameter during the course of the algorithm. Unfeasible solutions are allowed during the search, the amount of violation being penalized.

The general structure of the tabu search meta-heuristic (TS) we propose is presented in Algorithm 1. It starts with an initial feasible solution p, generated by a greedy method that aims to fully utilize vehicles and minimize the total cost. Then, at each iteration,

a neighborhood is selected probabilistically based on the current value of r, the selected neighborhood is searched, and the best move is chosen (Lines 7-8). This move must not be tabu, unless it improves the current best solution p_{best} (aspiration criterion). The algorithm adds the new solution to the elite set $\mathcal{E}$ when it improves p_{best}, remembers the current ratio r at which p_{best} was found (Lines 9-13), and updates $\mathcal{E}$ by removing a solution based on its value and the difference between solutions (Section 5.7).

Algorithm 1 Tabu search

```
 1: Generate an initial feasible solution p
 2: p_best ← p
 3: Elite set E ← ∅
 4: Ratio of selecting routing neighborhood to leg neighborhood r ← 1
 5: STOP ← 0
 6: repeat
 7:    A neighborhood is selected based on the value of r
 8:    Find the best solution p' in the selected neighborhood of p
 9:    if p' is better than p_best then
10:       p_best ← p'
11:       r_best ← r
12:       Add (p_best, r_best) to the elite set E; update E
13:    end if
14:    p ← p'
15:    if p_best not improved for IT_cNS iterations then
16:       if p_best not improved after C_cNS consecutive executions of Control procedure then
17:          if E = ∅ then
18:             STOP ← 1
19:          else
20:             Select randomly (p, r_p) (and remove it) from the elite set E
21:             Diversify the current solution p
22:             Set r ← r_p and reset tabu lists
23:          end if
24:       else
25:          Apply Control procedure to update the value of r
26:          p ← p_best
27:       end if
28:    end if
29: until STOP
30: p_best ← Post-optimization(p_best)
31: return p_best
```

Initially, the search freely explores the solution space by giving each neighborhood the same probability of being selected. The *Control* procedure is triggered to reduce the probability of selecting leg neighborhoods whenever the best solution is not improved for IT_{cNS} TS iterations (Line 25). This helps to limit the size of the search region, giving the routing moves more time to fully optimize routes. The search then starts from the

current best solution p_{best} (Line 26). Moreover, after C_{cNS} consecutive executions of the procedure *Control* without improvement of the current best solution p_{best}, a solution p is taken randomly and removed from the elite set $\mathcal{E}$ (Line 20), and a *Diversification* mechanism is triggered to perturb p (Line 21). The value of r is reset to the value at which the solution taken from the elite set was found, and all tabu lists are reset to empty (Line 22). The search then proceeds from the perturbed solution p. The search is stopped when the elite set $\mathcal{E}$ is empty. Finally, a post-optimization procedure is performed to potentially improve the current best solution p_{best} (Line 30).

5.2 Search space

As described in Section 3, a solution is a set of work assignments, each work assignment consisting of a sequence of route legs linking supply points. The search space is thus made up of the work assignments.

For a solution p, let $c(p)$ denote the total travel cost of the work assignments, and let $q(p), w_c(p)$, and $w_s(p)$ denote the total violation of vehicle load, customer time windows, supply point time windows, respectively. The total vehicle-load violation is computed on a route leg basis with respect to the value Q, whereas the total violation of time windows of customers is equal to $\sum_{d \in p} max\{(a_d - l_d), 0\}$, and the total violation of time windows of supply points is equal to $\sum_{s \in p} max\{(t(s) - \delta - a_s), (a_s - t(s)), 0\}$, where a_d and a_s are the arrival time at customer d and supply point s, respectively.

Due to the time synchronization restriction at supply points, the arrival time at the first customer d of each leg l starting at supply point s is always calculated as $a_d = t(s) + \delta(s) + c_{sd}$, no matter when the vehicle arrives at supply point s. This helps to prevent the propagation among legs within a work assignment of time-window unfeasibility.

Solutions are then evaluated according to the weighted fitness function $f(p) = c(p) + \alpha_1 q(p) + \alpha_2 w_c(p) + \alpha_3 w_s(p) + F * m$, where α_1, α_2, α_3 are penalty parameters adjusted dynamically during the search. The updating scheme is based on the idea of Cordeau et al. (2001). At each iteration, the value of α_1, α_2 and α_3 are modified by a factor $1 + \beta > 1$. If the current solution is feasible with respect to load constraints, the value of α_1 is divided by $1 + \beta$; otherwise it is multiplied by $1 + \beta$. The same rule applies to α_2 and α_3 with respect to time window constraints of customers and supply points, respectively. In our algorithm, we set $\alpha_1 = \alpha_2 = \alpha_3 = 1$ and $\beta = 0.5$.

5.3 Initial solution

We assume the supply points (customer zones) are indexed in increasing order of opening time. Thus if $t(s_1) \leq t(s_2)$ then $s_1 < s_2$ and vice versa. We then construct an initial solution by building each work assignment sequentially. Each work assignment construction consists of two phases: 1) Determine the first supply point for the current work assignment; 2) Each leg is then created sequentially by applying a greedy algorithm.

In the first phase, the supply point s with earliest opening time and unserviced customers is assigned as the initial supply point of the first leg of the current work assignment. The first leg l is then created by applying a greedy algorithm in the second phase. If the leg l ends at a supply point s', we continue applying the greedy algorithm to build the next leg of l in which s' is now used as the initial supply point. Otherwise, if the leg l ends at the main depot, it means the current work assignment cannot be used anymore, and we return to the first phase to build another work assignment. This process is repeated until all customers are serviced.

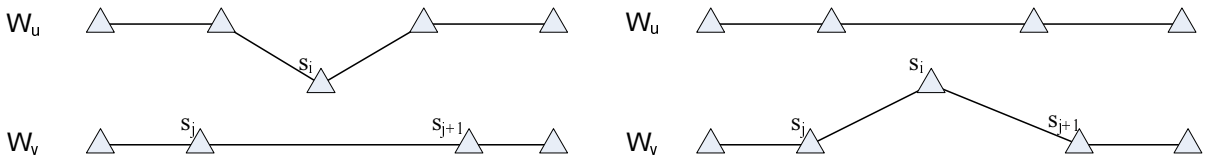
The greedy algorithm constructs each leg by attempting to minimize the cost and keep the vehicle working at full capacity as much as possible. Thus, for a given initial supply point s assigned to the leg, it finds a set of supply points $S' = \{s' \in \mathcal{S} | s'$ with unserviced customers and $t(s') > t(s)\}$. If $S' \neq \emptyset$, for each pair (s,s'), it creates an empty leg with s and s' as the initial and end supply point, respectively. It then assigns unserviced customers of customer zone s to this leg sequentially by applying the Solomon heuristic until the vehicle is full. When feasible legs exist, the one with minimum cost is selected. In the case there are no feasible legs or $S' = \emptyset$, it builds the last leg (s,g) by applying the Solomon heuristic.

5.4 Neighborhoods

Leg neighborhoods focus on repositioning legs at supply points within the time restrictions. Let W_u be the work assignment performed by vehicle u. Let s_{i-1} and s_{i+1} denote the predecessor and successor supply points, respectively, of s_i within a work assignment. The leg-move operators are:

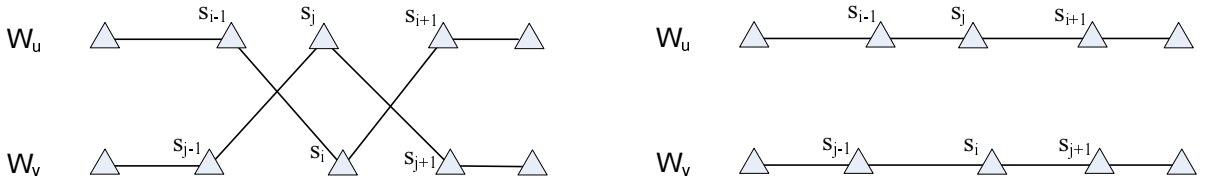
- *Relocate supply point.* Consider two work assignments W_u and W_v as illustrated in Figure 2. For supply point $s_i \in W_u$, such that $s_i \notin W_v$, and for each two successive supply points s_j, $s_{j+1} \in W_v$, if $s_j < s_i < s_{j+1}$ then move supply point s_i from work assignment W_u to W_v locating it between s_j and s_{j+1}. All customers serviced by s_i on W_u are also moved to W_v.
- *Exchange supply points.* Consider two work assignments W_u and W_v as illustrated

in Figure 3. For supply points $s_i \in W_u$ and $s_j \in W_v$ such that $s_{i-1} < s_j < s_{i+1}$ and $s_{j-1} < s_i < s_{j+1}$, swap s_i and s_j together with their customers.



(a) Work assignments before *Relocate* (b) Work assignments after *Relocate*

Figure 2: Relocate supply point



(a) Work assignments before *Exchange* (b) Work assignments after *Exchange*

Figure 3: Exchange supply points

When moving a supply point, all customers serviced by it are also moved. Therefore, the violations of load and time windows for customers are not changed by the move. The *move value* is thus defined as $\Delta f = \Delta c + F * \Delta m + \Delta w_s$. The three components of the summation are the difference in travel cost, the fixed cost of using vehicles, and the difference in violation of time windows at supply points between the value of the neighboring solution and the value of the current solution.

Routing neighborhoods try to improve routing by using different intra and inter route neighborhoods commonly used in the VRPTW literature: Relocation, Exchange and 2-opt. For each move in each neighborhood, two customers are considered.

- *Relocation move*: one of the two customers is taken from its current position and inserted after the other one.
- *Exchange move*: two customers are swapped.
- *2-opt move*: for two customers in the same leg, the edges emanating from them are removed, two edges are added, one of which connects these two customers, and the other connects their successor customers. For two customers in different legs, the work assignment segments following them are swapped preserving the order of customers succeeding them in each segment.

Moving customers could change the travel cost and the number of vehicles, as well as the level of constraint violations of load, time windows of customers, and time windows of supply points. Consequently, the value of a routing move is defined as $\Delta f = \Delta c + F * \Delta m + \Delta q + \Delta w_c + \Delta w_s$. Note that Δc, the change in the routing cost, may involve, beside a change in the routing cost between customers, a change in the routing cost from the last customer to the supply point at the end of the modified leg(s). For example, a routing move may impact on whether a vehicle has to go to a waiting station or not, therefore impacting the traveling cost from the last customer to its supply point.

5.5 Neighborhood selection and the Control procedure

The algorithm explores one neighborhood at each iteration. This neighborhood is randomly selected among the five previously defined. The main issue in this case is how to determine these selection probabilities.

Using a priori defined probabilities has several drawbacks, limiting the exploration capability of the algorithm. Fixed probabilities mean, in particular, that the algorithm would display the same behavior during the entire search. Moreover, the calibration of these probabilities would be extremely challenging and instance dependent. Indeed, too low routing-neighborhood probabilities (high leg-neighborhood probabilities) would result in an insufficient number of routing moves to adequately optimize the customer routes after the leg moves. The search may easily get stuck in the opposite case, as it needs to move to less-explored regions of the search space once a succession of routing moves have "optimize" routes.

We therefore define a dynamically adjusting mechanism for selecting neighborhoods. The goal is to enable the algorithm to freely explore the solution space in the initial moments of the search. We therefore assign the same initial probability of selection to all leg and routing neighborhoods. Later, as the number of supply points is much smaller than the number of customers in most TMM-VRPTW instances, the algorithm should perform more routing than leg moves, to ensure adequate optimization of routes. Consequently, after the initial phase, the probability of selecting leg neighborhoods becomes lower than the probability of selecting routing neighborhoods.

Let r be the neighborhood-selection parameter, and let a routing neighborhood to be selected with probability $r/(2+3r)$, and a leg neighborhood with probability $1/(2+3r)$. The equal initial probabilities are then obtained by setting $r = 1$. The *Control* procedure is then varying the value of r in the course of the algorithm to monotonically reduce (increase) the probability of selecting leg (routing) neighborhoods after each IT_{cNS} iterations without improvement of the best solution. A linear scheme $r_{k+1} = r_k + \Delta_r$ is used, where Δ_r is a user defined parameter.

5.6 Tabu lists and tabu duration

We keep a separate tabu list for each type of move. Elements of a solution generated by a move are given a tabu status as follows:

- Leg moves:
  - Relocate supply point: the position of supply point s_i just inserted into work assignment W_v cannot be changed by another relocate supply point move while it is tabu.
  - Exchange supply points: supply points s_i and s_j just swapped cannot be swapped again while they are tabu.
- Routing moves:
  - Relocation move: the position of customer i just inserted after customer j, cannot be changed by the same type of move while it is tabu.
  - Exchange move: customers i and j just swapped cannot be swapped again while they are tabu.
  - 2-opt move: a 2-opt move applied to customers i and j cannot be applied again to the same customers while tabu.

A tabu status is assigned to each tabu list element for θ iterations, where θ is randomly selected from a uniform interval. Generally, the tabu status of a move stays so for a number of iterations proportional to the number of possible moves. Consequently, we use different intervals of tabu list size for leg and routing moves. Since there are $O(m' * |\mathcal{S}|)$ possible leg moves, we set the interval of tabu list size for leg moves to $[m'*|\mathcal{S}|/a_1,\ m'*|\mathcal{S}|/a_2]$, where m' is the number of vehicles used in the initial solution, and a_1 and a_2 are user-defined parameters.

In TMZT-VRPTW, each supply point has its own customers. Therefore, the number of iterations during which a routing move within the zone of a supply point s remains tabu is only counted each time the algorithm deals with customers in that zone. The interval of tabu list size for routing moves for each supply point s with $|D_s|$ associated customers is therefore calculated as $[a_3 log_{10}(|D_s|),\ a_4 log_{10}(|D_s|)]$, where a_3 and a_4 are user defined parameters.

The aspiration criterion is a condition that allows the search to perform a move even though it is forbidden by the tabu conditions. In our tabu search, the aspiration criterion allows to accept moves declared tabu if it improves the current best solution.

5.7 Diversification strategy

A diversification strategy, based on an elite set and a frequency-based memory, moves the search to potentially unexplored promising regions when it begins to stagnate. In a nutshell, diversification aims to capitalize on the best attributes obtained so far by selecting a new working solution from the elite set and perturbing it based on long-term trends.

In more details, we use the elite set as a diversified pool of high-quality solutions found during the tabu search. The elite set starts empty and is limited in size. The quality and diversity of the elite set is controlled by the insertion of new best solutions produced by the tabu search and the elimination of the existing solutions in the elite set. The elimination is based on the Hamming distance $\Delta(p_1, p_2)$ measuring the number of customer positions that differ between solutions p_1 and p_2.

The elimination of a solution from the elite set is considered each time a new best solution p_{best} is inserted. There are two cases. When the elite set is not yet full, the solution which is most similar to p_{best} is deleted, i.e., the solution p with the smallest $\Delta(p, p_{best}) \leq 0.05(n+|\mathcal{S}|)$. This aims to balance the impact on pool quality and diversity. When the elite set is full, p_{best} replaces the solution p that is the most similar to it, i.e., the one with the smallest $\Delta(p, p_{best})$.

The long-term frequency memory keeps a history of the arcs most frequently added to the current solution. Let t_{ij} be the number of times arc (i, j) has been added to the solution during the search process. The frequency of arc (i, j) is then defined as $\rho_{ij} = t_{ij}/T$, where T is the total number of iterations executed so far.

Diversification then proceeds to perturb the search that starts from the solution taken from the elite set by removing arcs with high frequency and inserting arcs with low frequency. Thus, the evaluation of neighbor solutions is biased so as to penalize the arcs most frequently added to the current solution. More precisely, a penalty $g(\bar{p}) = \bar{C}(\sum_{(i,j)\in A_a} \rho_{ij} + \sum_{(i',j')\in A_r}(1-\rho_{i'j'}))$ is added to the evaluation of the fitness $f(\bar{p})$ (Section 5.2) of a neighbor $\bar{p}$ of the current solution p, where $\bar{C}$ is the average cost of all arcs in the problem, and A_a and A_r are the sets of arcs that are added to and removed from the solution p in the move to $\bar{p}$, respectively. The diversification mechanism is executed IT_{div} iterations.

5.8 Post-optimization

The best solution obtained through the tabu search is enhanced by applying a number of well-known local search route improvement techniques. Two are intra-route operators,

the 2-opt of Lin (1965) and the Or-opt of Or (1976). The others are inter-route operators, the λ-interchange of Osman (1993), and the CROSS-exchange of Taillard et al. (1997). For the λ-interchange, we only consider the cases where $\lambda = 1$ and $\lambda = 2$ corresponding to the (1,0), (1,1), (2,0), (2,1), and (2,2)-interchange operators. A customer is re-allocated only to legs with the same initial supply point. The post-optimization procedure is executed for each customer zone separately.

The post-optimization procedure starts by applying in random order the five λ-interchange and CROSS-exchange inter-route operators. Each neighborhood is searched on all possible pairs of legs (in random order) of the same starting supply point and stopped on the first improvement. The solution is then modified and the process is repeated until no further improvement can be found. The search is then continued by locally improving each leg of the current starting supply point in turn. The intra-route 2-opt and Or-opt neighborhoods are sequentially and repeatedly applied until no more improvement is found.

6 Computational Results

The objective of the numerical experimentation is threefold. First, to study the impact of a number of major parameters and search strategies on the performance of the proposed algorithm in order to identify the most efficient ones. The second objective consists in evaluating the performance of the method through comparisons with currently published results. We finally analyze the impact of synchronization on solution quality.

TS is implemented in C++. Experiments were run on a 2.8 GHz Intel Xeon 4-core processor with 16GB of RAM. Six sets of instances generated by Crainic et al. (2012) were used throughout the experiments. These Euclidean instances are identified as A1, A2, B1, B2, C1, and C2. The numbers of customer zones for these sets are 4, 8, 16, 32, 36, and 72 respectively. The numbers of customers are 400, 1600, and 3600 for set of type A, B, and C, respectively. One waiting station is set for every 100 customers in all instances.

6.1 Algorithm design and calibration

We aimed for a general algorithmic structure avoiding instance-related parameter setting. We therefore defined settings as functions of problem size for the main parameters of the proposed algorithm, tabu tenure, neighborhood selection-control, diversification triggering, and size of the elite set.

6.1.1 Tabu tenure calibration

The intervals for the tabu list tenures for leg and routing moves were defined in Section 5.6 as $[m'^*|\mathcal{S}|/a_1, m'^*|\mathcal{S}|/a_2]$ and $[a_3 log_{10}(|D_s|), a_4 log_{10}(|D_s|)]$, respectively. Using a large interval for routing moves,[10,20], we tested different values for a_1 in the integral interval [7,9] and for a_2 in the integral interval [4,6]. We observed that too large an interval is not productive as low values cannot prevent cycling, while high ones overly restrict the search path. We therefore set a_1 and a_2 to 7 and 5, respectively.

A similar process, explored different values of a_3 in the integral interval [6,8] and a_4 in the integral interval [10,12] using the just fixed leg-move tabu tag interval. We found that the most appropriate values for a_3 and a_4 are 7 and 10, respectively.

6.1.2 Neighborhood selection control calibration

Two parameters govern the selection of neighborhoods, IT_{cNS}, the number of consecutive iterations without improvement in the best solution triggering a new execution of the *Control* procedure, and Δ_r, the adjustment factor of the neighborhood-selection ratio r.

The value of IT_{cNS} is defined as a function of the problem size, aiming to give each customer and supply point in each leg the possibility to be moved. Thus, $IT_{cNS} = e_1 * (m' * |\mathcal{S}| + n)$, where m' is the number of vehicles used in the initial solution, $|\mathcal{S}|$ and n are the numbers of supply points and customers, respectively, and e_1 is a user defined parameter. Similarly, we aimed to set the amplitude of Δ_r, and therefore the amplitude of the changes in the probability of selecting leg neighborhoods versus routing neighborhoods after each IT_{cNS} iterations, proportional to the ratio of the number of customers to the number of supply points. Thus, $\Delta_r = e_2 \log_{10}(n/|S|)$, where e_2 is a user defined parameter.

Searching for a good combination of values for e_1 and e_2 concerns balancing between exploration and exploitation. On one hand, the higher the value of IT_{cNS}, the more chances customers and supply points are to be moved between routes, thus favoring exploration. On the other hand, too high a IT_{cNS} value may waste time in useless moves. We experimented with different values of e_1 in the integral interval [1,5] and e_2 in the integral interval [1, 7]. Three runs were performed for each instance for 1 million iterations. Computational results for each combination of values (e_1, e_2) over all 60 instances are summed up in Table 1, which displays the average gaps to the previous best known solutions (BKS) of the solutions obtained by each combination.

The table indicates that (3,5) as the most appropriate combination for (e_1, e_2), improving the solution quality by 3.44% on average. We also observed that executing the

Table 1: Performance comparison between (e_1, e_2) combinations for fixed computing effort

e_1	e_2						
	1	2	3	4	5	6	7
1	-2.21%	-2.42%	-2.98%	-3.09%	-3.11%	-3.18%	-3.15%
2	-2.27%	-2.45%	-3.24%	-3.21%	-3.17%	-3.14%	-3.12%
3	-2.34%	-2.73%	-3.37%	-3.39%	-3.44%	-3.36%	-3.28%
4	-2.46%	-2.74%	-3.31%	-3.36%	-3.41%	-3.28%	-3.24%
5	-2.41%	-2.78%	-3.32%	-3.37%	-3.39%	-3.29%	-3.19%

algorithm with r greater than $50\log_{10}(n/|S|)$, yields an average improvement of the best solution of less than 0.1%, while requiring about 37.3% more time. Based on these results, we used (e_1, e_2) = (3, 5) and $r_{max} = 50\log_{10}(n/|S|)$, the maximum value of r, in the remaining experiments.

6.1.3 Neighborhood search strategy

The neighborhood exploration strategy specifies how to explore the neighborhoods. Several such strategies can be envisioned and we actually experimented with quite a number of them before selecting the one introduced in Section 5.5. For concision's sake, we placed in Annex A the description of the alternate strategies we explored and the numerical results supporting our selection.

The neighborhood-search strategy also specifies which move in the neighborhood is to be chosen at each iteration. We studied two strategies, first and best improvement, respectively. The former chooses the first neighboring solution that improves the objective function as the next starting solution, while the latter chooses the best neighbor thus requiring to evaluate all neighbors. The customers in each route are searched sequentially.

Table 2 reports comparison results between these two strategies. Corresponding average gaps to the previous BKS and average computation times are displayed in column (GAP to BKS) and (Time(min)), respectively.

Computational results show that using the same elite set size (=5), best improvement gives better solutions for all sets, while first improvement has a lower computation time. One observes, however, that the difference in computation time is smaller than the difference in solution quality, indicating that the best improvement strategy yields a better algorithm. More importantly, even doubling size of the elite set (=10), which results in longer computation times, the solutions of the first improvement strategy are

Table 2: Comparative performances between neighborhood search strategies

Problem set	Best improvement		First improvement			
	Elite set size = 5		Elite set size = 5		Elite set size = 10	
	GAP to BKS	Time (min)	GAP to BKS	Time (min)	GAP to BKS	Time (min)
A1	-3.01%	18	-1.95%	16	-2.06%	24
A2	-6.22%	10	-4.42%	8	-4.57%	14
B1	-4.47%	60	-2.82%	45	-2.98%	69
B2	-4.71%	39	-3.23%	27	-3.35%	46
C1	-3.78%	165	-1.14%	115	-1.24%	176
C2	-3.72%	104	-1.89%	80	-1.91%	108
Average	-4.33%	66	-2.57%	49	-2.68%	73

still significantly worse than the solutions of the best improvement strategy.

6.1.4 Elite set calibration and diversification

We now turn to the parameters characterizing the diversification procedure and the elite set utilization, and examine their impact the performance of the algorithm.

Four variants of the algorithm were studied corresponding to the different ways to set an elite solution as the new working solution and the inclusion, or not, of the diversification phase. The first two variants simply select an elite solution p at random and re-start the algorithm from it. The *Diversification* mechanism described in Section 5.7 is applied in the last two variants to diversify from the elite solution p.

The initialization of the r parameter following the selection of p is a component common to the four variants. We studied two alternatives where r was set to either the full or half the value at which p was found, respectively (i.e., $r = r_p$ or $r = r_p/2$). The size of the elite set is relevant for the *Diversification* mechanism only. Three values were tested, 1, 5, and 10.

Similar to previous experiments, we used formulas dependent on the problem dimensions for IT_{div} and C_{cNS}, which determine for how long exploration can proceed. Thus, the number of diversification phases is set to $IT_{div} = m' * |\mathcal{S}| + n$, where m' is the number of vehicles used in the initial solution, and $|\mathcal{S}|$ and n are the numbers of supply points and customers, respectively. We also set the number of consecutive executions of the *Control* procedure without improvement of the best solution to $C_{cNS} = min(3\log_{10}(n/|S|), (r_{max} - r)/\Delta_r)$, which keeps the value of C_{cNS} sufficiently high during the course of the algorithm, even though *Control* procedure is started with different values of r (remember that $r_{max} = 50\log_{10}(n/|S|)$). Intuitively, in the begining, r is small and C_{cNS} takes the value $3\log_{10}(n/|S|)$, while when r becomes large enough, C_{cNS} takes the value $(r_{max} - r)/\Delta_r$.

Table 3 displays the performance comparison between the four variants with the three different values for the elite set size. For each variant and size of the elite set, the table shows the average gaps to the previous BKS of the average cost of the best solutions of all instances, together with the corresponding average computation time in minutes over 10 runs.

Table 3: Performance comparison between multi-start variants

Elite set size	Without diversification				With diversification			
	1st variant		2nd variant		3rd variant		4th variant	
	$r = r_p$		$r = r_p/2$		$r = r_p$		$r = r_p/2$	
	GAP to BKS	Time	GAP to BKS	Time	GAP to BKS	Time	GAP to BKS	Time
0	-2.96%	24	-	-	-	-	-	-
1	-3.18%	29	-3.03%	38	-3.93%	35	-3.97%	47
5	-3.39%	39	-3.34%	47	-4.33%	66	-4.24%	71
10	-3.53%	50	-3.61%	59	-4.35%	92	-4.25%	98

As expected, results indicate that guidance using elite solutions contributes significantly to improve the performance of the algorithm. Without using the elite set, the algorithm requires the lowest computation effort but produces solutions with the lowest average (improvement) gap to the previous BKS of -2.96%, compared to all the variants using the elite set. Comparing the two variants corresponding to the two values at which r is reset, one observes that the solution quality is not very sensitive to this value, but computing effort is increasing when the value of r is lower ($r = r_p/2$).

One observes that the third and fourth variants are significantly better in terms of finding high quality solutions. This indicates that the long-term memory and diversification mechanism added to the algorithm are important features for high performance. Moreover, setting the size of the elite set to 5 achieves a better balance between solution quality and computation time, compared to a larger size of 10. Indeed, doubling the size of the elite set improves only slightly the solution quality, 0.02%, but requires 39% more time. We therefore set the size of the elite set to 5 and reset $r = r_p$.

6.2 Comparing performance to the literature

The performance of the proposed tabu search meta-heuristic is evaluated by comparing its performance with published results on the instances provided by Crainic et al. (2012). For concision sake, only aggregated results are provided in this section. Details can be found in Annex B.

Table 4 displays the comparison between the (best) results reported by Crainic et al. (2012) and those obtained by the proposed tabu search meta-heuristic run 10 times for

Table 4: Comparative performances on Crainic et al. (2012) instances

Problem set	Crainic et al. (2012)					TS						GAP to BKS
	Best	#Vehicles	DM	MWS	Time	Avg 10	Best 10	#Vehicles	DM	MWS	Time	%
A1	18575	24	1	36	5	18043.84	18010.52	23	2	36	18	-3.04
A2	15411	19	2	42	3	14495.23	14440.79	17	5	42	10	-6.29
B1	55653	51	14	180	22	53124.00	53036.13	45	31	168	60	-4.64
B2	47396	39	20	193	10	45092.96	45081.29	34	40	177	39	-4.88
C1	117426	87	39	423	50	112965.99	112843.60	79	73	394	165	-3.90
C2	101570	64	68	434	23	97727.18	97565.44	60	108	394	104	-3.93
Average	59338.50	47.33	24	218	18.83	56908.2	56829.63	43	43.17	201.85	66	-4.38

each instance. For comparison sake, we report the best results we obtained for the 10 runs, but provide both best and average results in the detailed tables of Annex B. Table 4 gives the best results (*Best* column), the number of vehicles (*#Vehicles* column), the number of times vehicles move directly from one customer zone to another customer zone without using waiting stations (*DM* column), and the number of times waiting stations are used for moving between customer zones (*MWS* column). Average computation times in minutes are displayed in the *Time* column, while the corresponding gaps to the previous BKS are given in the last column.

TS produces high quality solutions, with an average improvement gap of -4.38% compared to the previous BKS, yielding better solutions than Crainic et al. (2012) on all instances. Moreover, the proposed tabu search meta-heuristic produces consistently good solutions, the average solution quality being very close the that of the best one. Noteworthy, as shown in Table 3, without post-optimization and using the same size of the elite set (=5), TS obtains an average gap to the previous BKS of -4.33%. Thus, the post-optimization process helped to improve solution quality 0.05% on average, requiring only a few extra seconds.

TS produces solutions that not only require less vehicles (8.87% on average), but also require less usage of waiting stations. More precisely, the TS we proposed uses waiting stations 12082 times compared to 13071 times in Crainic et al. (2012), vehicles move directly from one customer zone to another customer zone on 2582 occasions compared to 1449 occasions in Crainic et al. (2012). Thus, in our TS, 17.61% of the times vehicle move directly to another customer zone without using waiting stations, compared to 9.98% in Crainic et al. (2012). Moreover, the proposed TS provides better customer routing (i.e., traveling cost), with an average gap of -1.30% to the previous BKS. This advantage goes beyond the simple numerical performance in terms of cost to propose a distribution system that is structurally better with less vehicles traveling for idling purposes.

6.3 Synchronization at supply points

Waiting stations are introduced as locations where vehicles can wait when the direct move would get them at supply points sooner than the opening times. In this section, we analyze the impact of waiting stations on solution quality.

In all previous experiments, traveling cost and time were equivalent. In order to analyze the impact of waiting without modifying the travel costs, we explicitly introduced into the model a waiting cost measure related to the customer-to-waiting station movement generating the need to wait. Thus, the waiting cost for a (customer, supply point) pair is computed as a percentage of the total cost from the customer to the waiting station and from the latter to the supply point. We performed 6 runs corresponding to a percentage equal to 10%, 20%, 30%, 40%, 50%, and 100%.

The experiment was run on the C2 set, which includes the largest instances in terms of the numbers of customers, supply points, and waiting stations. Table 5 sums up the solution-quality variations for the six cases compared to the case without waiting costs. The table displays the solution-quality variations in terms of the total cost, traveling cost, number of vehicles, and synchronization requirement at supply points. One observes that higher waiting cost result in vehicles performing longer routes (the routing cost increases) to avoid going to waiting stations. Consequently, the number of direct moves increases, and accordingly, the number of moves using waiting stations is reduced. Moreover, it also requires more vehicles, resulting in a higher total cost.

Table 5: Impact of waiting cost on solution quality

Impact on solution quality	Increase the cost of waiting by					
	10%	20%	30%	40%	50%	100%
Total cost	2.87%	5.15%	7.46%	9.75%	11.77%	20.23%
Routing cost	3.78%	6.78%	10.04%	13.13%	15.39%	26.59%
#Vehicles	0.82%	1.47%	1.64%	2.13%	3.61%	5.91%
DM	3.91%	15.14%	23.24%	31.30%	34.19%	58.41%
MWS	-5.45%	-8.68%	-10.90%	-12.97%	-14.14%	-21.38%

7 Conclusions

We proposed a tabu search meta-heuristic for the Time-dependent Multi-zone Multi-trip Vehicle Routing Problem with Time Windows. The proposed model formulation provided the means to identify clearly the main components of the decision set of the problem. We

could thus propose a tabu search method that works on multiple neighborhoods, which are used to improve both the routing and the assignment of routes to vehicles. The selection of neighborhoods is dynamically adjusted along the search to keep the balance between exploration and exploitation. Moreover, a diversification strategy guided by an elite set and a frequency-based memory is introduced to not only provide a certain level of diversity to the search, but also help incorporate good attributes into newly created solutions.

Experimental results illustrated clearly the superior performance of the proposed methodology compared to the literature. It yields higher quality solutions in terms of both required number of vehicles and traveling cost. In addition, the utilization of waiting stations, resulting from the synchronization restriction at supply points, was significantly reduced. The quality of these results in terms of costs, times, and frequency of direct movements indicate that the proposed methodology could prove magisterially efficient in actual applications.

Acknowledgments

Partial funding for this project has been provided by the Natural Sciences and Engineering Council of Canada (NSERC), through its Industrial Research Chair and Discovery Grant programs, by the Université de Montréal, and by the EMME/2-STAN Royalty Research Funds (Dr. Heinz Spiess). We also gratefully acknowledge the support of Fonds de recherche du Québec through their infrastructure grants and of Calcul Québec and Compute Canada through access to their high-performance computing infrastructure.

References

A. Benjamin and J. Beasley. Metaheuristics for the waste collection vehicle routing problem with time windows, driver rest period and multiple disposal facilities. *Computers & Operations Research*, 37(12):2270–2280, 2010.

J. Brandão and A. Mercer. The multi-trip vehicle routing problem. *The Journal of the Operational Research Society*, 49(8):799–805, 1998.

J. F. Cordeau, G. Laporte, and A. Mercier. A unified tabu search heuristic for Vehicle Routing Problems with time windows. *Journal of the Operational Research Society*, 52:928–936, 2001.

T. G. Crainic, N. Ricciardi, and G. Storchi. Models for evaluating and planning city logistics systems. *Transportation Science*, 43(4):432–454, 2009.

T.G. Crainic, M. Gendreau, and Y. Gajpal. Multi-Zone Multi-Trip Vehicle Routing Problem with Time Windows. Publication CIRRELT-2012-36, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, Université de Montréal, Montréal, QC, Canada, 2012.

B. Crevier, J.-F. Cordeau, and G. Laporte. The multi-depot vehicle routing problem with inter-depot routes. *European Journal of Operational Research*, 176(2):756–773, 2007.

M. Dell'Amico and M. Trubian. Applying tabu search to the job-shop scheduling problem. *Ann. Oper. Res.*, 41(1-4):231–252, 1993.

L. D. Gaspero and A. Schaerf. A composite-neighborhood tabu search approach to the traveling tournament problem. *Journal of Heuristics*, 13(2):189–207, 2007.

J.-P. Hamiez, J. Robet, and J.-K. Hao. A tabu search algorithm with direct representation for strip packing. In *Proceedings of the 9th European Conference on Evolutionary Computation in Combinatorial Optimization*, EvoCOP '09, pages 61–72. Springer-Verlag, 2009.

B.-I. Kim, S. Kim, and S. Sahoo. Waste collection vehicle routing problem with time windows. *Computers & Operations Research*, 33(12):3624–3642, 2006.

S. Lin. Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44:2245–2269, 1965.

B. M. Ombuki-Berman, A. Runka, and F. T. Hanshar. Waste collection vehicle routing problem with time windows using multi-objective genetic algorithms. In *Proceedings of the Third IASTED International Conference on Computational Intelligence*, pages 91–97. ACTA Press, 2007.

I. Or. *Traveling Salesman-type Combinatorial Problems and their relation to the Logistics of Blood Banking*. PhD thesis, Department of Industrial Engineering and Management Science, Northwestern University, Evanston, IL, 1976.

I. H. Osman. Metastrategy simulated annealing and tabu search algorithms for the Vehicle Routing Problems. *Annals of Operations Research*, 41:421–452, 1993.

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

S. Salhi and R. Petch. A GA based heuristic for the vehicle routing problem with multiple trips. *Journal of Mathematical Modelling and Algorithms*, 6:591–613, 2007.

E. Taillard, P. Badeau, M. Gendreau, F. Guertin, and J.-Y. Potvin. A tabu search heuristic for the Vehicle Routing Problem with soft windows. *Transportation Science*, 31:170–186, 1997.

E. D. Taillard, G. Laporte, and M. Gendreau. Vehicle routing with multiple use of vehicles. *The journal of the operational research society*, 47(8):1065–1070, 1995.

C. D. Tarantilis, E. E. Zachariadis, and C. T. Kiranoudis. A hybrid guided local search for the vehicle-routing problem with intermediate replenishment facilities. *Informs Journal on Computing*, 20(1):154–168, 2008.

J. Xu, M. Sohoni, M. McCleery, and T. G. Bailey. A dynamic neighborhood based tabu search algorithm for real-world flight instructor scheduling problems. *European Journal of Operational Research*, 169(3):978–993, 2006.

A Neighborhood selection strategies

This Annex briefly presents the neighborhood selection strategies that have been studied to determine how neighborhoods in our algorithm should be combined and how they are intertwined. To select the neighborhood selection strategy that we included into the proposed tabu search (Section 5.5), we tried two variants of a greedy strategy (denoted L and M), three variants of fixing the probability of selecting leg neighborhoods (denoted N1, N2, and N3), and three variants of a two-level strategy.

The performance of all variants is compared to that of the *Control* procedure embedded into the proposed tabu search, where the selection of the neighborhood is driven dynamically.

Greedy strategy L. All five neighborhoods are evaluated at each iteration. The best move is selected among all the moves in all neighborhoods.

Greedy strategy M. One type of neighborhood, leg or routing, is first selected at each iteration based on the value of r. Then, the neighborhoods of the selected type are evaluated. The best move is finally selected.

Fixing the probability of selecting neighborhoods. A single neighborhood is evaluated at each iteration selected randomly by a fixed probability. The **Strategy N1** allocated the same probability to all neighborhood ($r = 1$). **Strategy N2** sets the probability of selecting routing neighborhoods greater than the probability of selecting leg neighborhoods ($r > 1$) to give more time to optimizing routes following a leg move. **Strategy N3** allows the search to freely explore the solution space at the beginning by allowing all neighborhoods to share the same probability of selection, while restricting the selection of leg moves (and encouraging the optimization of routes following such a move) afterwards. It thus applies Strategy N1 for the first T_1 iterations, and Strategy N2 for the last T_2 iterations.

The same number of iterations, 1 million, was performed for all strategies on each instance. In addition, we ran three times the M, N2, N3 strategies for three values of $r = \{15, 30, 45\}$. Table A6 displays the average best- solution results for all strategies and problem types. The last column (GAP to Control (%)) shows the average gap of each strategy relative to the results obtained by the *Control* neighborhood selection strategy used in the implementation studied in the main sections of the paper.

Examining the experimental results, we observe that the performance of Strategy L is the worst. The main reason is that leg moves, which are necessary to diversify the search, are not selected sufficiently often in this strategy. Indeed, one observes the better performance of the greedy Strategy M where leg moves are given a higher probability of selection. The best performance in this group of strategies is offered by Strategy N3,

Table A6: Performance comparison between neighborhood selection strategies

Strategy		Problem set						GAP to
		A1	A2	B1	B2	C1	C2	Control (%)
L		18593.07	14958.53	56089.99	48058.11	117755.10	107118.90	4.43
M	$r = 15$	18246.81	14744.71	54524.40	46218.93	116762.10	100245.40	1.40
	$r = 30$	18253.74	14740.81	54408.71	46526.30	116008.10	100474.13	1.41
	$r = 45$	18268.07	14729.11	54100.25	46295.93	117025.80	100810.93	1.43
N1		18255.39	14660.38	54109.26	46122.47	116200.50	100056.02	1.04
N2	$r = 15$	18198.15	14664.80	53927.76	46536.96	115836.10	99894.28	0.94
	$r = 30$	18198.24	14636.68	54029.69	46535.03	115298.60	99982.87	0.94
	$r = 45$	18200.22	14663.08	53865.71	46330.53	115812.40	99978.64	0.92
N3	$r = 15$	18143.10	14657.21	54364.18	46320.67	116379.40	99931.71	1.09
	$r = 30$	18158.00	14646.96	53869.52	46282.02	115768.70	99957.79	0.83
	$r = 45$	18142.19	14646.21	53741.44	46128.00	115595.40	99973.40	0.70
Control		18102.02	14549.13	53539.38	46000.00	114260.90	98503.27	

however, adjusting the type of exploration as the search progresses, from equal probabilities at the beginning of the search, to increased route-optimization neighborhoods in later stages.

Yet, fixing a priori the selection probabilities may limit the exploration capability of the search. Indeed, the *Control* procedure, implementing a selection strategy driven dynamically by variations in solution quality, outperforms all the other strategies.

Two-level strategy. It consists in running TS using one type of neighborhood until no further improvement can be found, and then switching to using the other type of neighborhood.

The algorithm starts at the "high level" applying the two leg neighborhoods in randomized order. In each case, all pairs of vehicles are searched sequentially, and the search stops either at the first improvement, or with the best improvement once all pairs have been evaluated. When both neighborhoods are completely evaluated but no improving solution has been identified, the best move is selected and becomes the current solution. "Low-level" search then starts from this solution using the three routing neighborhoods in randomized order with either the first or the best improvement criterion. Low-level search continues until no further improving solution can be found. The algorithm stops when the maximum number of iterations is reached, otherwise it is switched back to the high level.

In the variant described above, only one leg move is implemented before making the transition from the high level to the low level, no mater whether improved moves have been performed or not. It can cause to the two-level search to be trapped in local optima,

since employing only one leg move may not change significantly the solution. Thus, we define the second variant where if no improving leg move is found in a high-level TS sequence, the search will continue for a small number of leg moves before switching to the low level.

We run each variant for 1 million iterations. Table A7 reports comparison results between these variants of the two-level neighborhoods strategy. Corresponding average gaps to the solutions obtained by the tabu search with the *Control* neighborhood-selection strategy and average computation times are displayed in column (GAP to Control) and (Time (min)), respectively.

For the first variant, best improvement produces higher quality solutions with an average error gap of 1.25% compared to 2.87% for the first improvement strategy. However, first improvement requires less computation time. For the second variant, the number of leg moves (high level search) is selected randomly in the integral interval [2,5]. The result is reported in the two last columns of Table A7. One observes that this variant performs better than the two previous variants. However, given a same total number of iterations (1 million), compared to strategies N2 and N3 where neighborhoods are mixed liberally or their selections are driven by a fixed number, the two-level strategy does not perform better.

Table A7: Comparative performance of two level neighborhood-search strategies

Problem set	First improvement		Best improvement		Best improvement & random leg moves	
	GAP to Control	Time (min)	GAP to Control	Time (min)	GAP to Control	Time (min)
A1	1.96%	32	1.51%	52	1.35%	64
A2	2.27%	25	1.62%	47	1.24%	53
B1	4.95%	59	2.59%	102	2.21%	128
B2	1.93%	41	0.74%	83	0.58%	94
C1	3.01%	105	0.65%	171	0.21%	193
C2	3.12%	93	0.39%	121	0.01%	138
Average	2.87%	59.17	1.25%	96	0.93%	111.67

B Detailed Results

Tables A8, A9, and A10 display comparison results on the Crainic et al. (2012) instances. The first group of columns displays the results of Crainic et al. (2012), the best solution values (*Best* column), the number of vehicles (*#Vehicles* column), the number of times vehicles move directly from one customer zone to another without passing through waiting stations (*DM* column), and the number of times waiting stations are used for moving between customer zones (*MWS* column). The next group of columns displays the same information for the proposed tabu search, plus the average values (*Avg 10* column) and

standard deviations (*Std* column) over 10 runs, as well as the corresponding gap to the previous BKS (last column).

Table A8: Best performance comparison between our algorithm and Crainic et al. (2012) on problem sets A

Instances	Crainic et al. (2012)				TS						GAP
No	Best	#Vehicles	DM	MWS	Avg 10	Std	Best 10	#Vehicles	DM	MWS	%
A1-1	17767	19	0	44	17152.7	24.36	17118.40	18	0	44	-3.65
A1-2	18783	24	0	35	18392.85	18.29	18371.20	24	0	35	-2.19
A1-3	16114	19	0	41	15749.17	14.10	15731.10	19	0	41	-2.38
A1-4	20936	31	0	30	20798.3	17.71	20780.00	31	0	29	-0.75
A1-5	15850	18	11	34	15633.8	17.46	15613.70	18	13	32	-1.49
A1-6	17456	20	0	40	16760.83	27.52	16720.80	19	0	40	-4.21
A1-7	19169	24	0	37	18790.99	15.08	18760.10	23	0	37	-2.13
A1-8	16790	20	2	40	16598.9	23.40	16553.30	20	4	38	-1.41
A1-9	21570	32	0	31	19361.32	43.02	19318.10	26	4	31	-10.44
A1-10	21316	32	0	26	21199.57	42.77	21138.50	32	0	26	-0.83
Average	18575.1	23.9	1.3	35.8	18043.84	24.37	18010.52	23	2.1	35.3	-3.04
A2-1	16380	21	1	41	15864.90	29.61	15823.40	20	2	41	-3.40
A2-2	18433	24	2	36	17140.57	35.63	17085.00	21	10	35	-7.31
A2-3	14654	18	4	42	13785.34	37.68	13717.80	16	6	41	-6.39
A2-4	13056	13	0	49	12828.75	31.65	12774.90	13	0	48	-2.15
A2-5	14256	14	3	46	13622.56	33.07	13579.70	13	3	46	-4.74
A2-6	17480	24	7	30	15430.33	36.16	15355.10	20	17	26	-12.16
A2-7	13955	15	1	49	13325.39	31.88	13287.90	14	0	49	-4.78
A2-8	14975	18	0	46	14815.85	29.67	14765.50	18	0	46	-1.40
A2-9	14430	16	0	45	13149.59	45.10	13103.00	13	4	46	-9.20
A2-10	16490	22	6	35	14988.99	53.43	14915.60	20	8	35	-9.55
Average	15410.9	18.5	2.4	41.9	14495.23	36.39	14440.79	16.8	5	41.3	-6.29

Table A9: Best performance comparison between our algorithm and Crainic et al. (2012) on problem sets B

Instances	Crainic et al. (2012)				TS						GAP
No	Best	#Vehicles	DM	MWS	Avg 10	Std	Best 10	#Vehicles	DM	MWS	%
B1-1	71007	89	4	153	65660.43	81.26	65570.2	76	30	140	-7.66
B1-2	53419	44	22	180	50886.42	80.38	50778.1	39	42	160	-4.94
B1-3	51175	41	20	186	48328.92	33.47	48284.8	36	36	170	-5.65
B1-4	54331	42	25	176	53547.07	64.46	53439.8	42	32	168	-1.64
B1-5	54072	45	12	186	51435.01	56.81	51309.7	37	31	171	-5.11
B1-6	54593	50	16	182	51604.48	61.61	51577.5	41	45	161	-5.52
B1-7	53322	46	15	180	51843.15	67.97	51785	44	26	170	-2.88
B1-8	55423	48	5	193	53941.65	70.38	53820.5	45	22	183	-2.89
B1-9	55208	53	19	168	52555.43	71.93	52427.7	48	32	162	-5.04
B1-10	53979	48	5	195	51437.39	75.74	51368	42	14	188	-4.84
Average	55652.9	50.6	14.3	179.9	53124.00	66.40	53036.13	45.00	31.00	167.30	-4.62
B2-1	44889	32	21	193	43492.67	79.24	43318.3	27	43	175	-3.50
B2-2	50427	47	22	178	46697.50	109.51	46506.1	39	54	152	-7.78
B2-3	48941	40	25	188	46839.89	121.45	46582	36	52	168	-4.82
B2-4	45894	28	19	206	44627.32	80.86	44531.5	28	34	193	-2.97
B2-5	46523	41	17	195	44538.50	89.31	44345.1	37	29	182	-4.68
B2-6	46441	36	16	199	43574.68	91.52	45066.5	35	21	190	-2.96
B2-7	44894	34	16	199	43601.74	61.52	43450.6	31	32	183	-3.22
B2-8	44549	28	23	201	43041.03	101.17	42816.4	25	43	187	-3.89
B2-9	46801	37	21	198	44543.77	62.17	44408.8	30	58	165	-5.11
B2-10	54606	62	16	172	49972.49	96.31	49787.6	50	33	171	-8.82
Average	47396.5	38.5	19.6	192.9	45092.96	89.31	45081.29	33.80	39.90	176.60	-4.77

Table A10: Best performance comparison between our algorithm and Crainic et al. (2012) on problem sets C

Instances	Crainic et al. (2012)				TS						GAP
No	Best	#Vehicles	DM	MWS	Avg 10	Std	Best 10	#Vehicles	DM	MWS	%
C1-1	115967	86	67	394	111143.00	62.10	111196	75	114	359	-4.11
C1-2	113176	92	43	411	109077.00	98.18	109320	81	85	388	-3.41
C1-3	114773	80	62	409	109013.30	118.31	108758	68	101	374	-5.24
C1-4	114310	83	30	435	110412.90	109.96	110269	79	61	404	-3.54
C1-5	121245	100	25	423	115635.50	116.06	115335	84	55	406	-4.87
C1-6	115324	87	39	426	111761.60	180.11	111558	81	64	400	-3.27
C1-7	120443	86	31	431	117028.30	98.70	116869	83	59	401	-2.97
C1-8	115473	78	40	433	111768.80	97.67	111565	72	76	394	-3.38
C1-9	126060	89	34	427	121740.30	112.01	121607	84	68	386	-3.53
C1-10	117493	85	18	442	112079.20	120.98	111959	74	47	426	-4.71
Average	117426.4	86.6	38.9	423.1	112965.99	111.41	112843.60	78.10	73.00	393.80	-3.90
C2-1	101232	69	70	423	97566.43	101.30	97350.2	67	101	391	-3.83
C2-2	98289	53	85	433	95340.22	105.34	95262.7	48	146	372	-3.08
C2-3	106180	65	71	426	102038.30	85.31	101888	61	94	397	-4.04
C2-4	97387	74	35	450	93426.91	96.49	93217.2	64	79	410	-4.28
C2-5	101090	53	74	442	96744.10	103.29	96602.5	52	98	414	-4.44
C2-6	106847	85	54	426	101813.60	88.09	101665	75	90	392	-4.85
C2-7	98471	62	76	424	95197.20	98.25	95096.4	60	99	395	-3.43
C2-8	99948	55	73	442	96761.28	82.87	96598.2	55	123	409	-3.35
C2-9	102301	57	79	432	97949.94	92.90	97691.2	56	134	373	-4.51
C2-10	103950	63	67	437	100433.80	94.98	100283	62	108	386	-3.53
Average	101569.5	63.6	68.4	433.5	97727.18	94.88	97565.44	60.00	107.20	393.90	-3.93