Multi-trip pickup and delivery problem with time windows and synchronization

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

Published online: 17 March 2016

Abstract In this paper, we consider two-tiered city logistics systems accounting for both the inbound and outbound traffic, that have not been taken into account in models and algorithms for vehicle routing research. The problem under study, called the Multi-trip Pickup and Delivery Problem with Time Windows and Synchronization, has two sets of intertwined decisions: the routing decisions which determine the sequence of customers visited by each vehicle route, the scheduling decisions which plan movements of vehicles between facilities within time synchronization restrictions. We propose a tabu search algorithm integrating multiple neighborhoods targeted to the decision sets of the problem. To assess the proposed algorithm, tests have been conducted on the first benchmark instances of the problem which have up to 72 facilities and 7200 customer demands. As no previous results are available in the literature for the problem, we also evaluate the performance of the method through comparisons with published results on two simplified problems: the Multi-zone multi-trip vehicle routing problem with separate delivery and collection, and the Vehicle routing problem with backhauls. The proposed algorithm is competitive with existing exact and meta-heuristic methods for these two problems.

Keywords Multi-trip pickup and delivery problem with time windows · Synchronization · Tabu search · Multiple neighborhoods

Electronic supplementary material The online version of this article (doi:10.1007/s10479-015-2001-7) contains supplementary material, which is available to authorized users.

✉ Teodor Gabriel Crainic
TeodorGabriel.Crainic@cirrelt.net

1 Department d'Informatique et de Recherche pérationnelle, Université de Montréal, Montreal, QC, Canada

2 Centre Interuniversitaire de Recherche sur les Réseaux d'Entreprise, la Logistique et le Transport (CIRRELT), Montreal, QC, Canada

3 Department de Management et Technologie, ESG, U.Q.A.M, Montreal, QC, Canada

4 Faculty of Engineering, Vietnamese-German University, Ho Chi Minh City, Vietnam

1 Introduction

We introduce a new problem class, the *Multi-trip Pickup and Delivery Problem with Time Windows and Synchronization* (*MT-PDTWS*), which generalizes a number of Pickup and Delivery with Backhauls (P&DB) problem settings (Savelsbergh and Solomon 1995; Parragh et al. 2008a, b; Berbeglia et al. 2007, 2010; Toth and Vigo 2002).

In the MT-PDTWS setting, a homogeneous fleet of vehicles operates multi-tour routes out of a single garage to deliver and pick up loads to and at customers, respectively. To-be-delivered loads are customer specific, are available at particular terminals within specified hard time windows, and must be delivered within the time window of the respective customer. The same or different customers have loads that must be picked up, within the customer time windows, and brought to a terminal, within one of its periods of activity, belonging to the subset of terminals associated to the particular customer. A vehicle must complete a terminal-to-customer delivery sequence before starting a pickup sequence or moving to a terminal for another delivery phase. Waiting at terminals may be strictly limited (in both time and space) and, thus, synchronization of vehicle arrivals and terminal operating time windows is an important characteristic of the problem setting. The original characteristics of the MT-PDTWS, setting it apart from and generalizing most P&DB problems, therefore are (1) multi-commodity demand defined as specific, time-dependent origin-to-destination loads to be delivered or picked up; (2) the synchronization of activities at terminals; and (3) multi-tour routes.

The MT-PDTWS arises in logistics and production planning. Our initial motivation comes from planning the operations of two-tiered City Logistics systems (Crainic et al. 2009). In such systems, inbound loads are sorted and consolidated at first-tier facilities (called *external zones*) located on the outskirts of the city, moved to second-tier facilities (the *satellites*), located close to or within the City Logistics-controlled area (the *CL-area*), by vehicles of various modes. In the second tier, a fleet of vehicles of size and motorization appropriate for the CL-area performs multi-tour routes to pickup outbound demands within the CL-area and bring them to satellites. Once there, planned appropriate pairs of first-tier and second-tier vehicles transfer inbound and outbound loads to each other according to a cross-docking strategy, without intermediate storage. The first-tier vehicles then bring the outbound loads to external zones, while the second-tier vehicles deliver the inbound loads to designated customers situated within the CL-area. This integration of inbound and outbound operations is aimed to help reduce the number of empty vehicle movements of all vehicle fleets and the freight traffic in the CL-area. As satellites are used as cross-dock transshipment facilities, the synchronization of the operations of first-tier and second-tier vehicles at satellites becomes one of the most constraining aspects of the problem.

To our best knowledge, the MT-PDTWS has not been addressed in the literature before. Crainic et al. (2012) discussed the issue of combining different types of routing activities within the City Logistics planning context, but no problem definition was provided, nor any modelling or algorithmic contribution. Our goal is to formally introduce and define the MT-PDTWS, provide a mathematical formulation and propose an efficient meta-heuristic (generalizing the method proposed by Nguyen et al. 2013, for the Time-dependent Multi-zone Multitrip Vehicle Routing Problem with Time Windows).

We make the following contributions: (1) we formally define and present the first formulation for the MT-PDTWS; (2) we propose an efficient tabu search meta-heuristic to address the problem; (3) we introduce a new set of benchmark instances with up to 72 facilities and 7200 customers; (4) we analyze the performance of the proposed method, including through comparisons with methods proposed for related P&DB problems, and study the impact of two main problem characteristics, namely combining pick up and delivery operations, and synchronization.

The remainder of the paper is organized as follows. Section 2 contains a detailed problem description and high-level model; to make the presentation more concise, the mathematical formulation is provided in Appendix of Supplementary material. Section 3 reviews the literature. The proposed methodology is described in Sect. 4. Computational results are then reported and analyzed in Sect. 5, while conclusions and future works are considered in Sect. 6.

2 Problem description

The system is composed of a garage, g, where the fleet of vehicles of homogeneous capacity Q is based, a number of facilities where customer-specific loads are available during particular (hard) time windows and to where loads picked up at customers may be brought during one of their time windows, and customers waiting for their loads to be delivered or picked up, or both, during their time windows. The route planning is to be performed for a certain schedule length, T, each route visiting one or several facilities (hence the "multi-tour" characterization) during their respective time windows to bring in or take away time-dependent customer loads.

We model the time-dependency characterizing demand and operations in the MT-PDTWS through *time windows*, the well-known representation device for vehicle routing problems. We first model facilities, which become available to receive vehicles for loading and unloading operations at particular time periods only. A particular set of loads destined to specific customers may be available at each such time period, and must be taken away and distributed. Then, as a given facility may be available at several periods during the schedule length considered, with a different set of loads at each occurrence, we define *supply points* as particular combinations of facilities and availability time periods (definition similar to that of Nguyen et al. 2013). Each supply point $s \in \mathcal{S}$ has a no-wait, hard opening time window $[t(s) - \eta, t(s)]$, specifying the earliest and latest times a vehicle may be at s, respectively. Hence, the vehicle must not arrive at s sooner than $(t(s) - \eta)$ and no later than $t(s)$. To model various possibilities of handling the former case, waiting stations (e.g., a parking lots) $w \in \mathcal{W}$ are provided where the vehicle may wait before moving to s. Otherwise, if there is no waiting station available, the vehicle goes to the garage to finish its route.

The second time-dependency phenomenon concerns customers, which may receive several loads from different supply points and, thus, during different time windows. The same or different customers may also have loads to be picked up and transported to one of a given subset of supply points. We model this time dependency by identifying each particular load as a customer demand, characterized by the routing activity and the customer involved, the supply point where it is available or the set of supply points that may take it in, and the particular customer time window. We thus define a set of *delivery-customer demands*, each $d \in \mathcal{C}^D$ being characterized by the supply point where it is available, the customer it must be delivered to, and the time window when the delivery must be performed. We also define a set of *pickup-customer demands*, each $p \in \mathcal{C}^P$ being characterized by the customer shipping it and the time window within which the pickup must be performed, as well as the set of admissible supply points $\mathcal{S}_p \in \mathcal{S}$ to which the load may be delivered, the choice of a particular one being part of the decisions characterizing the MT-PDTWS. Then, for each customer demand $i \in \{\mathcal{C}^P \cup \mathcal{C}^D\}$, we set $(i, q_i, \delta(i), [e_i, l_i])$ to stand for the quantity q_i of demand to be delivered or picked up at the customer demand i within the hard time window $[e_i, l_i]$ with a service time $\delta(i)$.

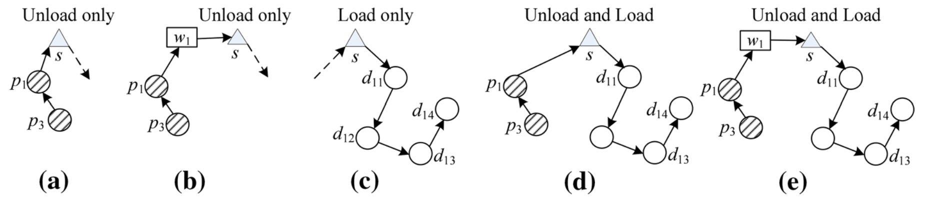

Fig. 1 Activities at supply points

Each supply point s may thus service a group of either pickup-customer demands $\mathcal{C}_s^P \subseteq \mathcal{C}^P$, or delivery-customer demands $\mathcal{C}_s^D \subseteq \mathcal{C}^D$, or both. The loads collected from pickup-customer demands in $\mathcal{C}_s^P$ are brought to s during its time windows. Similarly, the freight to be delivered to delivery-customer demands in $\mathcal{C}_s^D$ have to be loaded at s during the same time window. Let $\varphi(s)$ and $\varphi'(s)$ be the times required, respectively, to load and unload a vehicle at s. Figure 1 represents the possible loading and unloading activities of a vehicle at a supply point s. Striped and empty disks stand for pickup and delivery-customer demands, respectively, dashed lines indicating empty moves.

Figures 1a and 1b depict instances of the "unload only" operation in which, after arriving at the supply point with the collected freight from pickup-customer demands, the vehicle unloads all freight, then it leaves the supply point empty for its next tour or the garage to end its activity. The two instances differ in the level of synchronization only. From the last serviced customer demand, the vehicle goes directly to the supply point s, Fig. 1a, if it can arrive at s within the time window $[t(s) - \eta, t(s)]$. Otherwise, when the direct move gets the vehicle to s sooner than $t(s) - \eta$, the vehicle goes to a waiting station, Fig. 1b, and waits there in order to get to s within its time window. Figure 1c represents the "load only" case when the vehicle arrives empty at s and loads freight. Figures 1d and 1e depict instances of *unload & load* operations in which, after unloading all the freight collected from pickup-customer demands, the vehicle loads freight and leaves to deliver it to designated delivery-customer demands.

Let a *pickup* or *delivery leg* be a route that links one or several pickup or delivery-customer demands, respectively, and a supply point. We then define two types of pickup and delivery legs, together with their feasibility rules:

- *Direct-pickup leg* A route run by a vehicle that services one pickup-customer demand or a sequence of pickup-customer demands and then travels directly to the supply point to unload all freight (Fig. 1a); A pickup leg assigned is feasible if the vehicle with a total load not exceeding Q arrives at s within its time window $[t(s) - \eta, t(s)]$ after servicing a subset of pickup-customer demands in $\mathcal{C}_s^P$ within their time windows;
- *Indirect-pickup leg* Similar to the case of the direct-pickup leg, except that, after servicing the last pickup-customer demand, the vehicle has to go to a waiting station and wait there due to the synchronization requirement at the supply point (Fig. 1b);
- *Single-delivery leg* A route run by a vehicle that arrives empty at a supply point s, loads freight and delivers it to one delivery-customer demand or a sequence of delivery-customer demands in $\mathcal{C}_s^D$ (Fig. 1c); A single-delivery leg is feasible if the vehicle arrives empty at s at time $t' \in [t(s) - \eta, t(s)]$ to load freight not exceeding Q, and leaves s at time $t' + \varphi(s)$ to perform the delivery to the corresponding subset of customer demands in $\mathcal{C}_s^D$ within their time windows.
- *Coordinated-delivery leg* A route combining a single-delivery leg and either a direct-pickup (Fig. 1d) or an indirect-pickup leg (Fig. 1e) at a supply point s; A coordinated-

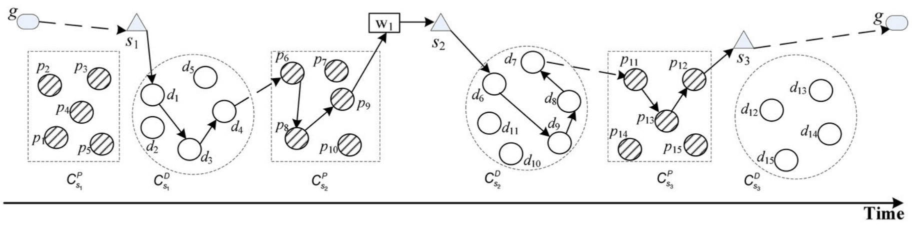
g
s_1
w_1
s_2
s_3
g
p_1 p_2 p_3 p_4 p_5
d_1 d_2 d_3 d_4 d_5
p_6 p_7 p_8 p_9 p_{10}
d_6 d_7 d_8 d_9 d_{10} d_{11}
p_{11} p_{12} p_{13} p_{14} p_{15}
d_{12} d_{13} d_{14} d_{15}
$\mathcal{C}^P_{s_1}$ $\mathcal{C}^D_{s_1}$ $\mathcal{C}^P_{s_2}$ $\mathcal{C}^D_{s_2}$ $\mathcal{C}^P_{s_3}$ $\mathcal{C}^D_{s_3}$
Time

Fig. 2 A four-leg work assignment illustration

delivery leg is feasible if the vehicle arrives at s at time $t' \in [t(s)-\eta, t(s)]$ to unload all the collected freight, then starts to load delivery demands not exceeding Q at time $t' + \varphi'(s)$, and leaves s at time $t(s) + \varphi'(s) + \varphi(s)$ to perform the delivery for the corresponding subset of customer demands in $\mathcal{C}^D_s$ within their time windows.

A sequence of legs, starting and ending at the garage and performed by a single vehicle, is called a ***work assignment***. Vehicles operate according to the *Pseudo-Backhaul* strategy of Crainic et al. (2012), in which any delivery or pickup leg must be completed before another one may start. Figure 2 illustrates a four-leg work assignment, where s_1, s_2, s_3 are supply points, g and w_1 are the garage and waiting station, respectively, and several pickup and delivery customer demand sets are given by $\mathcal{C}^P_{s_1} = \{p_1, p_2, p_3, p_4, p_5\}$, $\mathcal{C}^D_{s_1} = \{d_1, d_2, d_3, d_4, d_5\}$, $\mathcal{C}^P_{s_2} = \{p_6, p_7, p_8, p_9, p_{10}\}$, $\mathcal{C}^D_{s_2} = \{d_6, d_7, d_8, d_9, d_{10}, d_{11}\}$, $\mathcal{C}^P_{s_3} = \{p_{11}, p_{12}, p_{13}, p_{14}, p_{15}\}$, and $\mathcal{C}^D_{s_3} = \{d_{12}, d_{13}, d_{14}, d_{15}\}$. Dashed lines stand for the empty travel. The vehicle operating this work assignment performs a sequence of four legs $\{r_1, r_2, r_3, r_4\}$, where $r_1 = \{s_1, d_1, d_3, d_4\}$ is a single-delivery leg, $r_2 = \{p_6, p_8, p_9, w_1, s_2\}$ is an indirect-pickup leg, $r_3 = \{s_2, d_6, d_9, d_8, d_7\}$ is a coordinated-delivery leg, and $r_4 = \{p_{11}, p_{13}, p_{12}, s_3\}$ is a direct-pickup leg. The vehicle first moves empty out of the garage g to supply point s_1 and starts loading delivery demands. After loading for a time $\varphi(s_1)$, it leaves s_1 to service delivery-customer demands d_1, d_3, d_4 in $\mathcal{C}^D_{s_1}$, then moves empty to pickup customer zone $\mathcal{C}^P_{s_2}$ for collecting freight at pickup-customer demands p_6, p_8, p_9. For synchronization reasons, the vehicle goes from customer demand p_9 to the waiting station w_1 and waits there in order to arrive at s_2 within its opening time window. Once at s_2 (at some arrival time t), it performs unloading from t for a duration $\varphi'(s_2)$, and then loads from time $t + \varphi'(s_2)$ for a time $\varphi(s_2)$, after which it leaves s_2 to service delivery-customer demands d_6, d_9, d_8, d_7 in $\mathcal{C}^D_{s_2}$. After servicing the last delivery-customer demand d_7, it moves empty to pickup customer zone $\mathcal{C}^P_{s_3}$. There, after loading freight at pickup-customer demands p_{11}, p_{13}, p_{12}, the vehicle moves to supply point s_3 within its opening time window. Once at s_3, this vehicle starts unloading freight for a duration of $\varphi'(s_3)$. At the end, the vehicle moves back empty to g to complete its work assignment.

Let F stand for fixed cost for operating a vehicle work assignment. The set of available vehicles is denoted by $\mathcal{K}$. Let also c_{ij} to stand for the cost (money, time, distance, etc.) associated with each pair of sites (supply points, waiting stations, and customer demands) i and j making up the set of nodes of the complete space-time network describing the problem $(i, j \in \{g \cup \mathcal{C}^D \cup \mathcal{C}^P \cup \mathcal{S} \cup \mathcal{W}\})$.

The MT-PDTWS can then be seen as the problem of (1) assigning pickup-customer demands to supply points, and (2) selecting a set of work assignments (pickup and delivery legs) each to be performed by one vehicle. The objective is to minimize the total cost, which is comprised of the routing cost of operating the work assignments and the fixed cost of using the vehicles, while the following conditions are satisfied:

1. Every vehicle starts and ends its leg sequence at the garage g;
2. Each pickup-customer demand p is assigned to exactly one supply point $s \in \mathcal{S}_p$;
3. Every vehicle required to service customer demands in $\mathcal{C}_s^P \cup \mathcal{C}_s^D$ must reach its supply point $s \in \mathcal{S}$ within its no-wait, hard opening time window (it may wait at a waiting station, eventually). Assume the arrival time at s is t; Once at s:
   - If the vehicle is not empty, the freight it contains, picked up from customer demands in $\mathcal{C}_s^P$, is first unloaded, this operation starting at time t and continuing for a duration of $\varphi'(s)$; Once empty, the vehicle may either:
     - (1) load goods for a duration of $\varphi(s)$ and then leave s to deliver to customer demands in $\mathcal{C}_s^D$, or
     - (2) move empty either to another pickup customer zone to collect goods, or directly to another supply point for loading goods, or
     - (3) go to the garage g to complete the work assignment;
   - Otherwise, the vehicle starts to load goods for customer demands in $\mathcal{C}_s^D$ at time t and continues loading for a duration of $\varphi(s)$, after which it leaves s to deliver the goods. After performing a tour within the delivery customer zone $\mathcal{C}_s^D$, the vehicle may continue its movement as either the situations (2) or (3) described above;
4. Every customer demand is visited by exactly one vehicle (it belongs to exactly one leg) with a total load not exceeding Q, and each customer demand $i \in \{\mathcal{C}^D \cup \mathcal{C}^P\}$ is serviced within its hard time window $[e_i, l_i]$, i.e., the vehicle may arrive before e_i and wait to begin service, but must not arrive later than l_i.

The full mathematical formulation is provided in Appendix of Supplementary material.

3 Literature review

The MT-PDTWS we introduce in this paper is a new variant in the vehicle routing problem class generalizing both a number of pickup and delivery problem settings and the routing problems typically studied in the City Logistics literature.

Relative to City Logistics routing, the MT-PDTWS extends the Time-dependent Multi-zone Multi-trip Vehicle Routing problem with Time Windows (TMZT-VRPTW) by considering an additional type of customer demands. The TMZT-VRPTW addresses only the demands for delivery within the CL-controlled area, which corresponds to only delivery-customer demands in our setting, while the MT-PDTWS considers both delivery and pickup-customer demands. Crainic et al. (2009) introduced the TMZT-VRPTW and proposed a decomposition-based heuristic approach to address it. The general idea is to solve each customer-zone routing out of each supply point subproblem independently, and then put the created vehicle tours 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. Nguyen et al. (2013) later investigated an alternative approach that addresses these two decisions simultaneously within a tabu search framework. The proposed method yields solutions with higher quality up to 4.42 % in term of total cost, requiring not only less vehicles, but also less usage of waiting stations, when compared to the previous approach.

There has been extensive research on the pickup and delivery problem variants as illustrated in the surveys and book cited in the Introduction. Based on the difference in the sequence of customer service, Parragh et al. (2008a, b) divided them into two subclasses: the first refers to transportation of goods from the depot to delivery (*linehaul*) customers and from pickup (*backhaul*) customers to the depot, while the second refers to those problems where goods are transported between pickup and delivery locations. As we follow the Pseudo-Backhaul strategy in which any delivery or pickup phase must be completed before another one may be started, the MT-PDTWS belongs to the first subclass.

One distinguishes between single-demand problem settings where linehaul and backhaul customers are disjoint, and the combined setting where the same customer has both a pickup and a delivery demand. In the former case, one finds problems in which linehaul customers of a given trip have to be serviced before backhaul customers of the same trip, called *Vehicle Routing problem with Backhauls* (*VRPB;* Osman and Wassan 2002; Brandão 2006), and problems in which linehaul and backhaul customers may be visited in any order called *Vehicle Routing problem with Mixed linehauls and Backhauls* (*VRPMB;* Dethloff 2002; Ropke and Pisinger 2006). In the combined case, each customer may be visited either exactly once (Nagy and Salhi 2005; Dell'Amico et al. 2006) or twice, once for delivery and once for pickup (Salhi and Nagy 1999; Gribkovskaia et al. 2001). Problems in this case are called *Vehicle Routing problem with Simultaneous Delivery and Pickup*.

The VRPB can be considered as a subproblem of the MT-PDTWS. More precisely, the VRPB addresses a single-tour routing of, first, delivery-customer demands out of the supply point s and, second, pickup-customer demands assigned to supply point s', where $t(s) < t(s')$. Time synchronization restrictions at supply points and waiting stations are not considered. Two variants, with and without time windows at customers, are considered in the VRPB literature. The number of studies dealing with the time-window variant is relatively smaller than those without time windows.

The *Vehicle Routing Problem with Cross-Docking* (*VRPCD*) partially shares the requirement of synchronizing vehicle operations with our problem. The VRPCD generally involves transporting products from a set of suppliers to their corresponding customers via a cross-dock. Products from the suppliers are picked up by a fleet of vehicles, consolidated at the cross-dock facility (i.e., sorted into groups according to their destinations), and immediately delivered to customers by the same set of vehicles, without delay or storage. A supplier and its customers are not necessarily served by the same vehicle. At the cross-dock facility, the unloading of a vehicle must be completed before reloading starts. Constraints might be imposed on the simultaneous vehicle arrival at the facility (Lee et al. 2006; Liao et al. 2010), or the arrival dependency among vehicles is determined by the consolidation decisions (Wen et al. 2008). Similarly to our problem, each vehicle thus operates pickup and delivery phases separately.

There are also significant differences between the VRPCD and the MT-PDTWS, however, and one might see the former as a very particular special case of the later. Thus, in the VRPCD, each vehicle performs a single-tour route composed of a sequence of two trips, first pickup and then delivery, using the cross-dock facility as intermediate storage. There are no such limitations in the MT-PDTWS, neither on the number of legs (trips), nor on their sequencing (note than the Pseudo-Backhaul rule permits sequencing several legs of the same type). This results in a multiple synchronization requirements for each MT-PDTWS work assignment (route).

Bettinelli et al. (2015) recently studied the Multi-zone multi-trip vehicle routing problem with time windows and separate delivery and collection (MZMT-VRPTW-DC). Similar to the MT-PDTWS, this problem involves scheduling a homogeneous fleet of vehicles to pick up or deliver loads at or to customers associated to a given set of supply points where vehicles synchronize operations. There are also differences between the two problem settings, however, notably, each pickup-customer demand is pre-assigned to a supply point, the departure times from supply points are fixed independently of the operation performed therein, vehicles arriving early at customers may wait paying a penalty cost proportional to the waiting time, and vehicles are allowed to stop at waiting stations at any time (including between customer visits). The MZMT-VRPTW-DC minimizes the sum of the vehicle fixed cost and the routing operating cost combining travel and waiting-penalty costs. The authors proposed a branch-and-cut-and-price algorithm to solve the problem. Experiments on a large set of instances with up to ten supply points and two hundred customers showed the algorithm to be very efficient for relatively small instances (all but two instances were solved to optimality within the time limit of one hour, tight lower bounds, slightly more than 1 % being obtained for the largest instances).

4 Solution method

We propose a tabu search (*TS*) meta-heuristic for the MT-PDTWS, inspired by and extending the method of Nguyen et al. (2013) introduced for the TMZT-VRPTW. The new developments address challenging characteristics of the problem at hand, namely the combined pickup and delivery operations, and the goal of scheduling service to pickup-customer demands. New neighborhoods are introduced to address these issues.

Section 4.1 introduces the general structure of the meta-heuristic. The search space is defined in Sect. 4.2, while Sect. 4.3 describes the construction of the initial solution. The main features of the tabu search algorithm are then given: the neighborhood structures (Sect. 4.4), the neighborhoods selection strategy (Sect. 4.6), the tabu status mechanism (Sect. 4.7), the diversification mechanism (Sect. 4.8), and the *post-optimization* procedure (Sect. 4.9).

4.1 General structure

The tabu search meta-heuristic exploits several neighborhoods operating on legs and routes, the neighborhood selection at each iteration being governed by a dynamically-adjusted *neighborhood-selection* parameter, $\overline{r}$ (Sect. 4.6). An elite set of solutions guides the long term behavior of the search, while a post-optimization procedure polishes the final best solution. The overall structure of the proposed tabu search algorithm for the MT-PDTWS is given in Algorithm 1.

An initial feasible solution z is generated using a greedy method seeking to fully utilize vehicles and minimize the total cost. One neighborhood is selected probabilistically at each iteration based on the current value of $\overline{r}$, then the selected neighborhood is explored, and the best move is chosen (lines 7-8). This move must not be tabu, unless it improves the current best TS solution z_{best} (aspiration criterion). The algorithm adds the new solution to an elite set $\mathcal{E}$ if it improves on z_{best}. It also remembers the value of the parameter $\overline{r}$ when this new best solution was found (lines 9–13), and finally updates the elite set $\mathcal{E}$ by removing a solution based on its value and the difference between solutions (Sect. 4.8).

Initially, the search freely explores the solution space by assigning the same selection probability to each neighborhood. Whenever the best TS solution z_{best} is not improved for IT_{cNS} TS iterations (line 15), the *Control* procedure (which updates the neighborhood-selection parameter) is called to reduce the probability of selecting leg neighborhoods (line 25). As a consequence, routing neighborhoods are selected proportionally more often, giving moves involving customers more opportunity to optimize routes. The search is re-initialized from the current best TS solution z_{best} after the execution of the $Control$ procedure (line 26). Moreover, after C_{cNS} consecutive executions of this procedure without improvement of the

Algorithm 1 Tabu search

```
1: Generate an initial feasible solution $z$
2: $z_{best} \leftarrow z$
3: Elite set $\mathcal{E} \leftarrow \oslash$
4: Initialize the neighborhood selection parameter $\overline{r} \leftarrow 1$
5: STOP $\leftarrow 0$
6: repeat
7:    A neighborhood is selected based on the value of $\overline{r}$
8:    Find the best solution $z'$ in the selected neighborhood of $z$
9:    if $z'$ is better than $z_{best}$ then
10:       $z_{best} \leftarrow z'$
11:       $\overline{r}_{best} \leftarrow \overline{r}$
12:       Add $(z_{best}, \overline{r}_{best})$ to the elite set $\mathcal{E}$; update $\mathcal{E}$
13:   end if
14:   $z \leftarrow z'$
15:   if $z_{best}$ not improved for $IT_{cNS}$ iterations then
16:      if $z_{best}$ not improved after $C_{cNS}$ consecutive executions of Control procedure then
17:         if $\mathcal{E} = \oslash$ then
18:            STOP $\leftarrow 1$
19:         else
20:            Select randomly $(z, \overline{r}_z)$ (and remove it) from the elite set $\mathcal{E}$
21:            Diversify the current solution $z$
22:            Set $\overline{r} \leftarrow \overline{r}_z$ and reset tabu lists
23:         end if
24:      else
25:         Apply Control procedure to update the value of $\overline{r}$
26:         $z \leftarrow z_{best}$
27:      end if
28:   end if
29: until STOP
30: $z_{best} \leftarrow$ Post-optimization($z_{best}$)
31: return $z_{best}$
```

current best TS solution z_{best}, a solution z is selected randomly and removed from the elite set (line 20), and a *Diversification* mechanism is applied to perturb z (line 21). The value of $\overline{r}$ is reset to the value it had when the corresponding elite solution was found, and all tabu lists are reset to the empty state (line 22). The search then proceeds from the new (perturbed) solution z. 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 z_{best} (line 30).

4.2 Search space

We allow the search to explore unfeasible solutions with respect to vehicle capacity and the time windows of customer demands and supply points, unfeasible solutions being penalized proportionally to the violations of these restrictions. More precisely, let $c(z)$ denote the total traveling cost for a solution z, and let $q(z)$, $w_c(z)$ and $w_s(z)$ denote the associated total violation of vehicle load, customer-demand time windows, and supply-point time windows, respectively. The total vehicle-load violation is computed on a leg basis with respect to the value Q, whereas the total violation of time windows of customer demands is set to $\sum_{i \in z} max\{(a_i - l_i), 0\}$, and the total violation of time windows of supply points is equal to $\sum_{s \in z} max\{(t(s) - \eta - a_s), (a_s - t(s)), 0\}$, where a_i and a_s are the arrival time at customer demand i and supply point s, respectively.

Solutions are then evaluated according to the weighted fitness function $f(z) = c(z) + \alpha^Q q(z) + \alpha^C w_c(z) + \alpha^S w_s(z) + F * m$, where m is the number of vehicles used in the current solution, while $\alpha^Q, \alpha^C, \alpha^S$ 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^Q, α^C and α^S are modified by a factor $1+\beta > 1$. If the current solution is feasible with respect to load constraints, the value of α^Q is divided by $1+\beta$; otherwise it is multiplied by $1+\beta$. The same rule applies to α^C and α^S with respect to time window constraints of customers and supply points, respectively. We set $\alpha^Q = \alpha^C = \alpha^S = 1$ and $\beta = 0.3$ in the experimentation reported on in Sect. 5.

4.3 Initial solution

To obtain an initial solution, the supply points are sorted and indexed in increasing order of their opening times, i.e., if $t(s_1) \leq t(s_2)$, then $s_1 < s_2$ and vice-versa. Next, each pickup-customer demand is assigned to one supply point, building each feasible work assignment sequentially.

There are several ways to assign pickup-customer demands to supply points. For example, each pickup-customer demand can be assigned to its closest supply point. Another way would have each supply point s service a predefined number of its closest pickup-customer demands. However, these simple strategies do not take into account that significant variations in delivery loads may exist among supply points. Such strategies may create imbalances in pickup and delivery demands at some supply points, reducing the possibility of *unload & load* operations at those supply points and, thus, increasing the number of empty movements.

Our approach aims to avoid this pitfall and generate an initial solution with a small total traveling cost and balanced unloading and loading operations at supply points. Considering both the distance from pickup-customer demands to supply points and the capacity of the latter to receive such demands, we proceed as follows:

1. Compute the total delivery demands assigned to each supply point. Let K_s denote this number for supply point $s \in \mathcal{S}$;
2. Bound the total volume vehicles can pickup and unload at supply point s to K_s;
3. Randomly select a pickup-customer demand p until all are assigned, and
   - Assign p to the nearest supply point in $\mathcal{S}_p$;
   - When the assignment violates the maximum capacity of the nearest supply point in $\mathcal{S}_p$, the pickup-customer demand p is randomly allocated to the supply point in $\mathcal{S}_p$ whose residual capacity is large enough to accommodate it.

Once the assignment of pickup-customer demands to supply points is completed, initial work assignments are built sequentially until all customer demands are serviced (assigned to a work assignment). For each work assignment:

1. Determine the initial supply point of the first leg as the supply point s with earliest opening time and unserviced customer demands;
2. Create one or a sequence of legs between supply point s and either another supply point s' or the garage g using the following GREEDY algorithm:
   (a) Identify the set of supply points $S' = \{s' \in \mathcal{S} | s'$ with unserviced customer demands and $t(s') > t(s)\}$;
   (b) If $S' = \varnothing$, the leg ends at the garage g and STOP $\leftarrow$ TRUE;
   (c) Otherwise ($S' \neq \varnothing$), for each pair (s, s')
      - Build the list of candidate customers: unrouted pickup-customer demands of s first, then unrouted delivery-customer demands of s and, finally, unrouted pickup-customer demands of s';

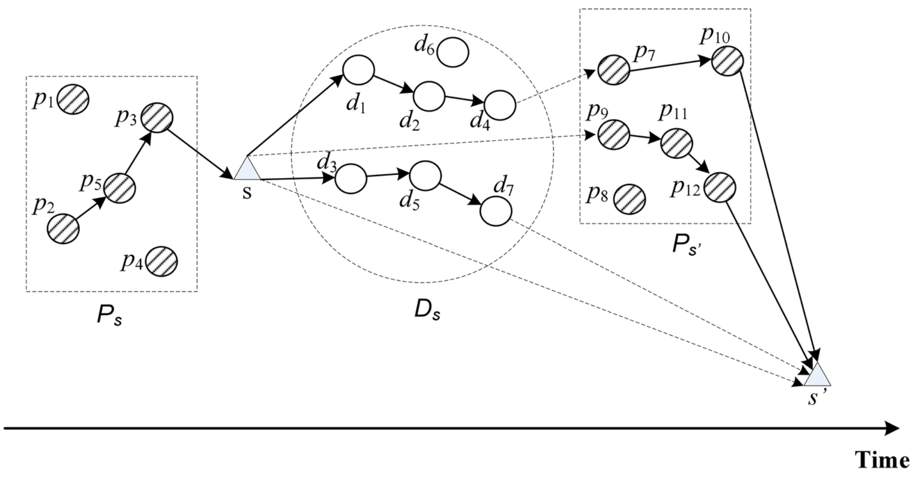

Fig. 3 A generation of a sequence of legs between two supply points

- Insert each candidate into the leg by applying the heuristic I1 of Solomon (1987) until the vehicle is full;

(d) Select the feasible leg with the smallest average cost per unit demand among all those generated between all pairs of s and s', and assign it to the current work assignment;

3. If the leg (or sequence of) ends at a supply point s', set $s \leftarrow s'$ and return to (2) to build the next leg(s);
4. Otherwise, i.e., the leg ends at the garage, STOP (the current work assignment is completed).

The average cost per unit demand is defined as the ratio of the total traveling time over the total demand carried by the vehicle between s and s', where the total demand for empty legs (no customers between s and s') is set to 1.

Figure 3 illustrates this procedure through a number of different possibilities when routing customer demands between two supply points s and s'. If there are unrouted pickup-customer demands of s, the greedy algorithm assigns them to the current work assignment, first generating the pickup leg $\{p_2, p_5, p_3, s\}$. Between supply point s and s', the algorithm may then generate (1) A sequence of a delivery leg $\{s, d_1, d_2, d_4\}$ and a pickup leg $\{p_7, p_{10}, s'\}$; (2) A pickup leg: $\{p_9, p_{11}, p_{12}, s'\}$; (3) A delivery leg: $\{s, d_3, d_5, d_7\}$; (4) An empty leg connecting s and s'. Which one is actually generated depends on the departure time at supply point s, the time windows and the distance between unserviced delivery- and pickup-customer demands of the supply points s and s', respectively.

4.4 Neighborhoods

A solution to the MT-PDTWS is a set of work assignments, each work assignment consisting of a sequence of legs with each leg corresponding to a sequence of customer demands. The neighborhood set of a current MT-PDTWS solution z is thus made up of all the solutions z' that can be obtained by perturbing in some way z. We use two types of perturbations, one that changes the sequence of customer demands within one or several legs and a second that changes the sequence of legs within one or several work assignments. Our neighborhood strategies use one or a combination of such perturbations in order to generate neighbor solutions. Each type of perturbation corresponds to working on a particular type of decision variable. We therefore group our neighborhood strategies into *routing neighborhoods* when they primarily change the sequence of customer demands in at least one leg, and *leg neighborhoods* when they primarily change the sequence of legs in at least one work assignment.

Note that, similar considerations were applied in Nguyen et al. (2013) to define neighborhoods for the TMZT-VRPTW that worked either on routing or scheduling decisions. The main difference and challenge for the MT-PDTWS is the presence of pickup-customer demands that not only need servicing but also require the determination of the delivery destination, that is, the assignment to a particular supply point. This translates into the definition of two types of leg sequences composed of either pickup- or delivery-customer demands, rather than a unique type for the TMZT-VRPTW, and work assignments made of variously interleaved legs of these two types. This also translates into new decision variables, determining the assignment of pick-customer demands to supply points, and more complex scheduling decisions. New neighborhoods are thus defined to address this challenge and handle these decisions for the MT-PDTWS.

4.4.1 Routing neighborhoods

Routing neighborhoods for the MT-PDTWS execute different intra- and inter-route (work assignment) moves commonly used in the VRP literature, Relocation, Exchange and 2-opt, attempting to improve the routing of the vehicle(s) servicing customer demands. Remember that each MT-PDTWS leg services either pickup- or delivery-customer demands but not both. As a result, when routing neighborhoods execute inter-route moves, the modified legs must continue to be of the same type, either pickup or delivery legs.

The definition of the delivery-customer demands specifies their assignment to particular supply points, which is similar to the case of the TMZT-VRPTW. Consequently, the corresponding neighborhoods are also similar, addressing two delivery-customer demands that belong to the same supply point (and, thus, to the same leg or different successive legs):

- *Relocation move* One of the two customer demands is taken from its current position and inserted after the other one;
- *Exchange move* Two customer demands are swapped;
- *2-opt move* For two customer demands in the

  Same leg The edges emanating from them are removed, two edges are added, one of which connects these two customer demands and the other connects their successor customer demands;
  Different legs The remaining customer sequences of these legs are swapped preserving the order of customer demands.

The situation is more complex for pickup-customer demand moves, the routing neighborhoods for pickup-customer demands differing substantially from the routing neighborhoods for delivery-customer demands. Indeed, while the latter involve legs that belong to the same supply point, this is not true for the former, as such pickup-customer demands may be reassigned to different supply points. We therefore define routing neighborhoods for pickup-customer demands that simultaneously modify the sequence of customer demands (the routing) and reassign them to supply points. The reassignment is achieved by allowing moves to be performed on legs that belong to different supply points.

The feasibility criterion for such a move, i.e., that reassigns a pickup-customer demand p from supply point s_i to supply point s_j, is that the latter belongs to the list of admissible supply points for p ($s_j \in \mathcal{S}_p$).

Table 1 List of pickup-customer demands and admissible supply points for Fig. 4

p	$\mathcal{S}_p$
p_1	$\{s_2, s_4\}$
p_2	$\{s_2, s_3, s_4\}$
p_3	$\{s_1, s_2, s_4\}$
p_4	$\{s_2, s_4\}$
p_5	$\{s_4\}$
p_6	$\{s_4, s_5\}$
p_7	$\{s_2, s_4\}$
p_8	$\{s_1, s_2, s_4\}$

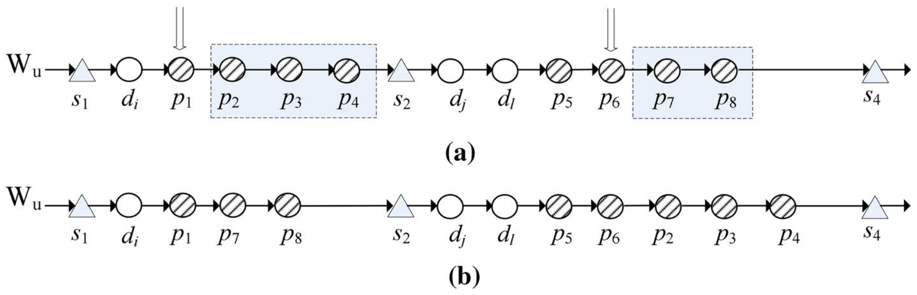
W_u
s_1 d_i p_1 p_2 p_3 p_4 s_2 d_j d_l p_5 p_6 p_7 p_8 s_4
(a)
W_u
s_1 d_i p_1 p_7 p_8 s_2 d_j d_l p_5 p_6 p_2 p_3 p_4 s_4
(b)

Fig. 4 An example of 2-opt routing neighborhood for pickup-customer demands. **a** Work assignment W_u before *2-opt*, **b** Work assignment W_u after *2-opt*

Three routing neighborhoods are thus considered for pairs of pickup-customer demands satisfying the feasibility criterion for supply-point reassignment:

- *Relocation move* One pickup-customer demand is shifted from its current position to another position, in the same or a different leg, which may be assigned to the same supply point or not;
- *Exchange move* The two pickup-customer demands are exchanged; They may belong to the same leg or, if the condition for supply-point reassignment allows it, to two distinct legs sharing one common supply point or not;
- *2-opt move* For two pickup-customer demands in the

  Same leg The edges emanating from them are removed, two edges are added, one of which connects these two pickup-customer demands, and the other connects their successor pickup-customer demands;
  Different legs, same supply point (thus in different work assignments) The remaining segments of these legs are swapped preserving the order of customer demands;
  Different legs, distinct supply points The remaining customer sequences of these legs are swapped preserving the order of customer demands.

Let us illustrate the condition for supply-point reassignment through a simple example. Consider Table 1 displaying the sets $\mathcal{S}_p$ of admissible supply points (Column 2) for pickup-customer demands $p \in \mathcal{C}^P$ (Column 1) for the work assignment W_u shown in Fig. 4a. Consider the two pickup-customer demands p_1 and p_6 in W_u belonging to different supply points, s_2 and s_4, respectively. The 2-opt move of p_1 and p_6 applied on W_u requires the supply-point reassignments of $\{p_2, p_3, p_4\}$ to s_4 and of $\{p_7, p_8\}$ to s_2. Pickup-customer demands p_7 and p_8 can be reassigned to supply point s_2 as $s_2 \in \{\mathcal{S}_{p_7} \cap \mathcal{S}_{p_8}\}$. Similarity, p_2, p_3, p_4 can be reassigned to s_4 as $s_4 \in \{\mathcal{S}_{p_2} \cap \mathcal{S}_{p_3} \cap \mathcal{S}_{p_4}\}$. The condition for supply-point reassignment is satisfied, therefore this 2-opt move is accepted. Figure 4b illustrates W_u after the move. On the other hand, the 2-opt move of p_1 and p_5 requires the supply-point reassignments of $\{p_2, p_3, p_4\}$ to s_4 and of $\{p_6, p_7, p_8\}$ to s_2. However, $s_2 \notin \mathcal{S}_{p_6}$, so p_6 can not be reassigned to supply point s_2. Due to the unfeasibility of the supply-point reassignment, this 2-opt move is not accepted.

All feasible neighbors are evaluated (Sect. 4.5) in the selected neighborhood (Sect. 4.6), and the best one is implemented.

4.4.2 Leg neighborhoods

Leg neighborhoods change the leg sequencing of work assignments and are described here in terms of supply-point moves. Indeed, each MT-PDTWS leg is assigned to the supply point where the vehicle either returns the collected freight or loads new freight (or both). Leg-neighborhood transformations can therefore be seen as the repositioning of supply points, together with the legs and customer demands associated with them, between work assignments. Two neighborhoods, Relocate and Exchange, are defined under the leg neighborhood category.

Relocate supply point moves remove a supply point, and the customer demands it services, from its current work assignment and inserts it into another work assignment. Exploration is performed for each work assignment W_u, each supply point $s_i \in W_u$, and each work assignment $W_v \neq W_u$, two cases being possible depending on whether the supply point to be reassigned belongs already to the target work assignment or not.

When $s_i \notin W_v$, for each two successive supply points $s_j, s_{j+1} \in W_v$, such that $s_j < s_i < s_{j+1}$, one moves s_i from work assignment W_u to W_v locating it between s_j and s_{j+1}. Figure 5 illustrates the case, where the relocation of supply point s_i also moved the associated pickup $\{p_i, p_j\}$ and delivery $\{d_m, d_n\}$ legs.

Once a supply point is relocated to a new work assignment, one may also perform the reassignment of pickup-customer demands to other supply points to maximize the *unload & load* operations at supply points and thus reduce empty movements. More precisely, whenever a pickup (or a single-delivery) leg assigned to s_i is relocated between s_j and s_{j+1}, and the vehicle only loads at s_{j+1} (or only unloads at s_j), one verifies the reassignment of the pickup-customer demands in the leg of s_i (or s_j) to supply point s_{j+1} (or s_i). If the reassignment is feasible, the customer demands in the leg of s_i are relocated between s_j and s_{j+1} to create a new *unload & load* operation at supply point s_{j+1} (or s_i) on the work assignment W_v. Otherwise, the leg assigned to s_i is just simply relocated between s_j and s_{j+1} as before. Figure 6 illustrates these possibilities when moving s_i on W_u, and its pickup leg $\{p_i, p_j\}$,

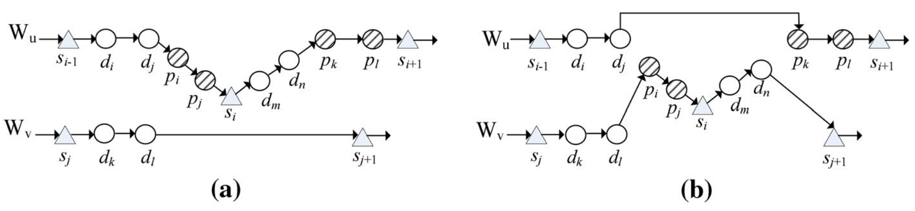

Fig. 5 Relocate a supply point with its pickup and delivery legs. **a** Work assignments before *Relocate*, **b** Work assignment after *Relocate*

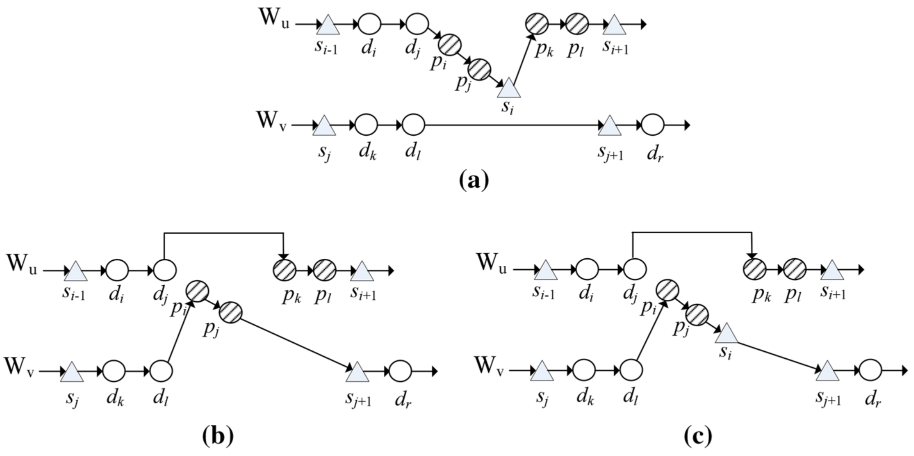

Fig. 6 Relocate a supply point and, eventually reassign a pickup leg. **a** Work assignments before *Relocate*, **b** Work assignments after *Relocate* pickup leg reassigned and supply point dropped, **c** Work assignments after *Relocate* pickup leg not reasigned

between s_j and s_{j+1} on W_v. As the leg assigned to s_{j+1} of W_v is a single delivery leg, one verifies whether reassigning p_i and p_j to supply point s_{j+1} is feasible; In the affirmative, i.e., $s_{j+1} \in \{\mathcal{S}_{p_i} \cap \mathcal{S}_{p_j}\}$, the reassignment is applied, and the movement yields the work assignment W_v shown in Fig. 6b. Notice that, an *unload & load* activity was created at s_{j+1} and that supply point s_i has been dropped from both work assignments. Figure 6c illustrates the case when this reassignment is not feasible.

When $s_i \in W_v$, three cases are possible according to the current activity at the supply point s_i of the vehicle operating the work assignment W_u:

- *Case 1 - Unload only.* Relocates the pickup leg r_i assigned to s_i in work assignment W_u. Three cases are possible according to the vehicle operation at supply point s_i in W_v prior the relocation:
  - *Case 1.1 - Unload only.* Let r_j be the pickup leg assigned to s_i in W_v; The move proceeds by concatenating the two pickup legs r_i and r_j. Appending r_i to r_j and r_j to r_i are both considered (Fig. 7).
  - *Case 1.2 - Load only.* Let r_j be the single-delivery leg assigned to s_i in W_v. The move proceeds by locating pickup leg r_i right before single-delivery leg r_j creating an *unload & load* operation at s_i (Fig. 8).
  - *Case 1.3 - Unload & load.* Let r_j be the pickup leg and r'_j the coordinate-delivery leg assigned to s_i in W_v, then move s_i from work assignment W_u to W_v by concatenating the two pickup legs r_i and r_j. Both cases of appending r_i to r_j and r_j to r_i are considered as in Case 1.1.
- *Case 2 - Load only.* Relocates the single-delivery leg r_i assigned to s_i within work assignment W_u. The three cases of vehicle operation at supply point s_i within work assignment W_v described above (Case 1) are also considered here. An *unload & load* operation is created in Case 2.1, while concatenation of delivery legs is attempted in Cases 2.2 and 2.3 (the concatenation of delivery legs r_i and r_j, already assigned to s_i in W_v, is also examined in two cases: one appending r_i to r_j and the other appending r_j to r_i).
- *Case 3 - Unload & load.* Relocates both the pickup leg r_i and the coordinate delivery leg r'_i assigned to the same supply point s_i in W_u. Three cases of vehicle operation at supply

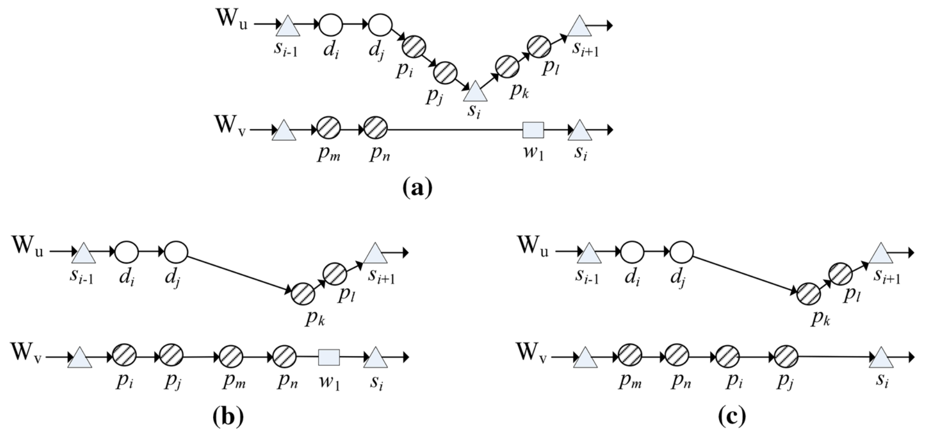

Fig. 7 Relocate a supply point: concatenation of two pickup legs. **a** Work assignments before *Relocate*, **b** Work assignments after *Relocate*, case: append (r_j, s_i) to (r_i, s_i), **c** Work assignments after *Relocate*, case: append (r_i, s_i) to (r_j, s_i)

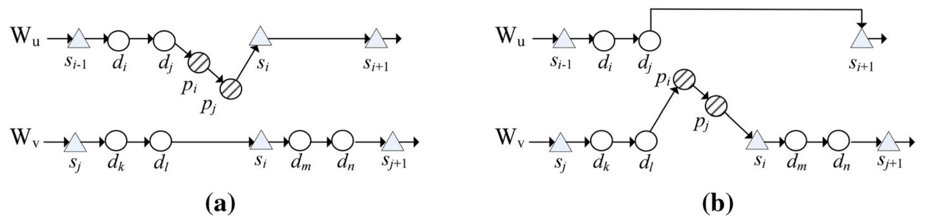

Fig. 8 Relocate a supply point: creation of an *unload & load* operation. **a** Work assignments before *Relocate*, **b** Work assignments after *Relocate*

point s_i in W_v are considered as in the previous cases. All possibilities of concatenation of delivery and pickup legs assigned to the same supply point s_i in both work assignments W_u and W_v are also examined.

Exchange supply point. The neighborhood exchanges supply points, and their associated legs and customer demands, between work assignments W_u and W_v. For supply points $s_i \in W_u$ and $s_j \in W_v$:

- When $s_{i-1} < s_j < s_{i+1}$
  - If $s_{j-1} < s_i < s_{j+1}$ then, swap s_i and s_j (swap both pickup- and delivery-customer demands if any);
  - If $s_{j-1} = s_i < s_{j+1}$ then, first swap s_i and s_j; Then, if there were pickup-customer demands assigned to s_i in both W_u and W_v, and because $s_{j-1} = s_i$, concatenate pickup-customer demands in W_v as described in Case 1.1 above; Also concatenate delivery-customer demands in both work assignments, if possible;
  - if $s_{j-1} < s_i = s_{j+1}$: same as item above.
- Otherwise, either $s_{i-1} = s_j$ or $s_j = s_{i+1}$. Then, swap supply points s_i and s_j and modify W_v and, possibly, W_u. Three cases are possible for W_v: 1) $s_{j-1} < s_i < s_{j+1}$; 2) $s_{j-1} = s_i < s_{j+1}$; and 3) $s_{j-1} < s_i = s_{j+1}$, and the treatment is the same as above. For W_u, when there were pickup (delivery)-customer demands assigned to s_j in both W_u

and W_v, and because $s_{i-1} = s_j$ (or $s_j = s_{i+1}$), concatenate pickup (delivery)-customer demands in W_u as described in Case 1.1 above.

The reassignment of pickup-customer demands to new supply points is performed as in the relocate supply point neighborhood, whenever it could create an *unload & load* operation. Only feasible reassignments are accepted, i.e., only if the new supply points belong to the list of permissible supply points of the pickup-customer demands.

4.5 Move evaluation

Moving legs or customer demands may change the transport cost and the number of vehicles, as well as the level of constraint violations of load and time windows (customer demands and supply points) restrictions. Consequently, the move value is defined as a sum of five terms $\Delta f = \Delta c + F * \Delta m + \Delta q + \Delta w_c + \Delta w_s$ representing the differences between the current and neighboring solutions in transport cost, fixed cost of using vehicles, and the violation of load, time windows at customer demands and supply points.

4.6 Neighborhood selection strategy

The proposed tabu search algorithm explores the search space of the MT-PDTWS using at each iteration one of the eight neighborhoods just described. The selection of the neighborhood is probabilistic, and controlled by the neighborhood-selection parameter $\overline{r}$ (see Nguyen et al. 2013, for a similar mechanism). We assign to routing and leg neighborhoods the selection probabilities $\overline{r}/(2 + 6\overline{r})$ and $1/(2 + 6\overline{r})$, respectively.

Leg and routing neighborhoods are given the same selection probability at the beginning of the search (by setting $\overline{r} = 1$). This allows the tabu search algorithm to freely explore the search space. Because the number of supply points is much smaller than the number of customer demands in most MT-PDTWS instances, the algorithm should perform more customer than leg moves to ensure adequate optimization of routes. Consequently, after the initial phase, the probability of selecting leg neighborhoods is gradually lowered, relative to the probability of selecting routing neighborhoods, by dynamically modifying the value of $\overline{r}$.

It is the *Control* procedure that varies the value of $\overline{r}$ during the execution of the tabu search 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 $\overline{r}_{k+1} = \overline{r}_k + \Delta\overline{r}$ is used, where $\Delta\overline{r}$ is a user-defined parameter, while $\overline{r}_{k+1}$ and $\overline{r}_k$ are values of $\overline{r}$ at iteration $k + 1$ and k, respectively.

4.7 Tabu lists and tabu duration

Five tabu lists are included in the meta-heuristic, one list for each type of leg and routing move (tabu lists do not distinguish between delivery- and pickup-customer demands, but the length of the tabu tenure does). The solution elements receiving a tabu status following a leg move are

- Relocation move: 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 move: supply points s_i and s_j just swapped cannot be swapped again while they are tabu;

while for routing moves

- Relocation move: the position of customer demand i just inserted after customer demand j cannot be changed by the same type of move while it is tabu;
- Exchange move: customer demands i and j just swapped cannot be swapped again while they are tabu;
- 2-opt move: a 2-opt move applied to customer demands i and j cannot be applied again to the same customer demands while tabu.

A tabu status is assigned to an element for θ iterations, where θ is randomly selected from a uniform interval. Any move declared tabu cannot be performed unless it would yield a solution improving the current best solution. Generally, the tabu status of a move should stay so for a number of iterations proportional to the number of possible moves. Consequently, we define different intervals for selecting the duration of the tabu tenure for leg and routing moves.

There are $O(m' * |\mathcal{S}|)$ possible leg moves. Consequently, the interval of the tabu list size for leg moves is set to $[m' * |\mathcal{S}|/a_1, m' * |\mathcal{S}|/a_2]$, where m' is the number of vehicles used in the initial solution, a_1 and a_2 are user-defined parameters where $a_1 > a_2$.

There are different tabu tenure intervals for routing moves depending on whether delivery- or pickup-customer demands are considered. As delivery-customer demands are pre-assigned to a particular supply point, moves involving delivery-customer demands may only occur within the same customer zone. Consequently, the tabu tenure interval for delivery-customer demand routing moves depends on the supply point s and its associated delivery-customer demands $|\mathcal{C}_s^D|$, and is calculated as $[a_3 log_{10}(|\mathcal{C}_s^D|), a_4 log_{10}(|\mathcal{C}_s^D|)]$, where a_3 and a_4 are user defined parameters, and $a_3 < a_4$. The number of iterations during which such a move remains tabu is increased only when the algorithm deals with delivery-customer demands in the corresponding zone.

In contrast, pickup-customer-demand-to-supply point assignments are not known in advance, rather, each pickup-customer demand has a list of available supply points that can service it. The routing moves we defined are thus modifying not only the position of the pickup-customer demands within work assignments, but also their assignments to supply points. Consequently, routing moves for pickup-customer demands are not restricted to a single supply point (as for delivery-customer demands above), but rather to a number of shared supply points. Hence, the tabu tenure interval for pickup-customer demand routing moves is proportional to the total number of pickup-customer demands ($|\mathcal{C}^P|$), and is calculated as $[a_5 log_{10}(|\mathcal{C}^P|), a_6 log_{10}(|\mathcal{C}^P|)]$, where a_5 and a_6 are user defined parameters, $a_5 < a_6$.

4.8 Diversification strategy

The diversification strategy, based on an elite set and a frequency memory, directs the search to potentially unexplored promising regions when the search 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 existing solutions in the elite set. The elimination is based on the Hamming distance $\Delta(z_1, z_2)$ measuring not only the number of customer demand positions that differ between solutions z_1 and z_2 (as for the TMZT-VRPTW), but also the differences between supply-point assignments of pickup-customer demands. This distance is computed according to Equation (1), where $\mathbf{T}(cond)$ is a valuation function that returns 1 if the condition $cond$ is true, 0, otherwise; $N_z[i]$ is the next location (a customer demand, the garage, or a supply point) visited by the vehicle after servicing customer demand i in solution z; and $S_z[i]$ is the supply point assigned to pickup-customer demand i in solution z.

$$\Delta(z_1, z_2) = \sum_{i \in \{\mathcal{C}^P \cup \mathcal{C}^D\}} \mathbf{T}(N_{z_1}[i] \neq N_{z_2}[i]) + \sum_{i \in \mathcal{C}^P} \mathbf{T}(S_{z_1}[i] \neq S_{z_2}[i]) \tag{1}$$

The elimination of a solution from the elite set is considered each time a new best solution z_{best} is inserted. There are two cases. If the elite set is not yet full, we delete only when there exists a solution very similar to the new z_{best}, i.e., we delete the solution z with the smallest $\Delta(z, z_{best}) \leq 0.05(|\mathcal{C}^D| + 2|\mathcal{C}^P| + |\mathcal{S}|)$. When the elite set is full, z_{best} replaces the solution z that is the most similar to it, i.e., the one with the smallest $\Delta(z, z_{best})$.

The long-term frequency memory keeps a history of the arcs most frequently added to the current solution, as well as of the supply-point assignments of pickup-customer demands most frequently used. 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. Similarly, let t'_{ps} be the number of times pickup-customer demand p has been assigned to supply point s during the search. The frequency of the supply-point assignment of customer demand p to s is defined as $\chi_{ps} = t'_{ps}/T$.

Diversification then proceeds to perturb the search that starts from the solution taken from the elite set by removing arcs with high frequency, inserting arcs with low frequency and promoting never-seen supply-point assignments. Thus, the evaluation of neighbor solutions is biased to penalize the arcs most frequently added to the current solution and the supply-point assignment most frequently used. The corresponding two penalties, $g_1(\bar{z})$ and $g_2(\bar{z})$, which are added to the fitness evaluation $f(\bar{z})$ (Sect. 4.2) of a neighbor $\bar{z}$ of the current solution z are given by equations 2 and 3, respectively,

$$g_1(\bar{z}) = \bar{C}\left(\sum_{(i,j)\in A_a} \rho_{ij} + \sum_{(i',j')\in A_r} (1 - \rho_{i'j'})\right) \tag{2}$$

$$g_2(\bar{z}) = \bar{C}\sum_{p\in\mathcal{C}^P}\left[\sum_{\substack{s\in\mathcal{S}_p\\ S_z(p)=S_{\bar{z}}(p)=s}} \chi_{ps} + \sum_{\substack{s\in\mathcal{S}_p\\ S_z(p)\neq s\\ S_{\bar{z}}(p)=s}} \chi_{ps} + \sum_{\substack{s\in\mathcal{S}_p\\ S_z(p)=s\\ S_{\bar{z}}(p)\neq s}} (1 - \chi_{ps})\right], \tag{3}$$

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 z in the move to $\bar{z}$, respectively. The diversification mechanism is executed IT_{div} iterations.

4.9 Post optimization

The best solution obtained during the tabu search is enhanced by applying a local-search *Supply-point-improvement* procedure followed by a *Leg-improvement* procedure. The purpose of these two procedures is to improve the routing and the supply-point assignments of the solution.

The Supply-point-improvement procedure proceeds by assigning a new supply point to each pickup-customer demand, keeping those that actually improve the solution. Pickup-customer demands are handled in random order. Then, for each pickup-customer demand p and each of its unassigned supply point $s \in \mathcal{S}_p$ (if any), p is removed from its current leg (i.e., current assigned supply point) and the cheapest fitness insertion is performed to insert

p into each pickup leg assigned to s. The best feasible improvement is executed (if any). The procedure then proceeds to the next unassigned supply point or, if all have been tried out, to the next pickup-customer demand.

Leg-improvement consists in 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 delivery-customer demand is re-allocated only to legs with the same initial supply point. This procedure is therefore executed for each delivery customer zone separately. For pickup-customer demands, the procedure is executed for all pairs of pickup-customer demands satisfying the supply-point assignment.

The Leg-improvement 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) and stopped on the first feasible 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 each vehicle in turn. The intra-route 2-opt and Or-opt neighborhoods are sequentially and repeatedly applied until no more improvement is found.

5 Experiments

The goal of the numerical experiments is threefold: (1) 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 best design (Sect. 5.2); (2) to evaluate the performance of the method through comparisons with published results for the MZMT-VRPTW-DC and the VRPB with and without time windows (Sect. 5.4); and (3) to analyze the impact on solution behavior and quality of sharing the same fleet of vehicles and synchronization schemes (Sects. 5.5 and 5.6, respectively).

The tabu search algorithm is implemented in C++. Experiments were run on a 2.8 GHz Intel Xeon 4-core processor with 16GB of RAM. We initiate this part of the paper with the description of the instances used for the experiments.

5.1 Test data generation

We generated MT-PDTWS test instances by adding pickup-customer demands to the TMZT-VRPTW instances of Crainic et al. (2009).

The quantity of pickup demand injected into an instance was determined by the ratio $BH = |\sum_{p\in\mathcal{C}^P} q_p / \sum_{i\in\{\mathcal{C}^P\cup\mathcal{C}^D\}} q_i|$ of the total pickup demand over the total demand (delivery and pickup). Based on the general observation that the volume of goods moving out of the city is relatively lower compared to the volume of goods moving in, we set the values of BH at $\{0.1, 0.3, 0.5\}$. For the sake of simplification, we have also used BH as the ratio of the number of pickup-customer demands over the total number of customer demands.

The attributes of each pickup-customer demand p for a given problem instance were generated as follows:

- Coordinates $[X_p, Y_p]$: uniformly distributed in the same interval used to generate the coordinates of the delivery-customer demands in the corresponding TMZT-VRPTW;

- Volume of demand q_p: randomly generated in the same interval as for delivery-customer demands, i.e., [5, 25], with respect to the value of BH;
- Service time $\delta(p)$: set to 20 as for TMZT-VRPTW;
- Number of supply points admissible for the pickup-customer demand p: selected randomly in the range [1, M_{SP}], where $M_{SP} = max_{p \in \mathcal{C}^P} \left\| S_p \right\|$. Let x denote this number. Then, the list of permissible supply points for p was determined by randomly selecting $s_1, s_2, ..., s_x$ supply points, sorted in increasing order of opening times;
- Time window $[e_p, l_p]$: e_p and l_p chosen randomly in the intervals [E_p - 300, E_p] and [L_p - 300, L_p], respectively (to ensure feasibility), where $E_p = t(s_1) - \delta(p) - \lceil c_{p,s_1} \rceil$ and $L_p = t(s_x) - \delta(p) - \lceil c_{p,s_x} \rceil$.

All other attributes are the same as in the TMZT-VRPTW instances. We thus generated six sets of 15 instances each, for a total of 90 problem instances. The six sets are called A1, A2, B1, B2, C1, and C2. Each set is further divided into three groups of 5 instances, each group being defined by one of the three different values of $BH = \{0.1, 0.3, 0.5\}$. Table 2 summarizes the parameters of all the MT-PDTWS instances. First and last columns give the instance name for the MT-PDTWS and for the original TMZT-VRPTW data, respectively. The next five columns display the numbers of supply points and waiting stations, respectively, the BH value, and the numbers of delivery and pickup customer demands. The X and Y coordinates of the square where supply points, waiting stations, and customers are uniformly distributed are shown in the next column, followed by the value of M_{SP}.

The opening times of supply points were generated randomly in the [1000, 15,400] range, while the limited allowable waiting time at supply points was set to $\eta = 100$. The vehicle-loading and vehicle-unloading times at supply points were set to 30, for all supply points. The fixed cost and the capacity of each vehicle were set to 500 and 100, respectively, for all instance sets.

5.2 Algorithm design and calibration

We aim for a general algorithmic structure avoiding instance-related parameter settings. We therefore defined settings as function of the problem size for the main parameters of the proposed algorithm, the tabu tenures, the neighborhood selection probabilities, and the diversification.

5.2.1 Tabu tenure calibration

The intervals for the tabu list tenures for leg, delivery, and pickup routing moves were defined in Sect. 4.7 as $[m'*|\mathcal{S}|/a_1, m'*|\mathcal{S}|/a_2]$, $[a_3 log_{10}(|\mathcal{C}_s^D|), a_4 log_{10}(|\mathcal{C}_s^D|)]$, and $[a_5 log_{10}(|\mathcal{C}^P|), a_6 log_{10}(|\mathcal{C}^P|)]$, respectively. Using a large interval for routing moves, [10, 20], we tested different values for a_1 in the integer interval [7, 10] and for a_2 in the integer 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 have therefore set a_1 and a_2 to 7 and 5, respectively.

A similar process was used to explore different values for a_3, a_4, a_5, a_6 in the integer intervals [4, 6], [7, 9], [6, 8] and [10, 12], respectively, using delivery and pickup routing tabu as defined above. We used a larger value of tabu tenure for routing moves on pickup-customer demands as they are not restricted to one customer zone as those on delivery-customer demands. We found that the most appropriate values for a_3, a_4, a_5 and a_6 are 6, 8, 7 and 10, respectively.

Table 2 Summary of test instances

Instance set	Instance names	# Supply points	# Waiting stations	*BH*	# Customers instances		[X,Y] coordinates	M_{SP}	Original
					Delivery	Pickup			
A1	A1-1 ... A1-5	4	4	0.1	400	44	[0,100]	2	A1-1 ... A1-5
	A1-6 ... A1-10			0.3		171			
	A1-11 ... A1-15			0.5		400			
A2	A2-1 ... A2-5	8	4	0.1	400	44	[0,100]	2	A2-1 ... A2-5
	A2-6 ... A2-10			0.3		171			
	A2-11 ... A2-15			0.5		400			
B1	B1-1 ... B1-5	16	16	0.1	1600	177	[0,200]	3	B1-1 ... B1-5
	B1-6 ... B1-10			0.3		685			
	B1-11 ... B1-15			0.5		1600			
B2	B2-1 ... B2-5	32	16	0.1	1600	177	[0,200]	3	B2-1 ... B2-5
	B2-6 ... B2-10			0.3		685			
	B2-11 ... B2-15			0.5		1600			
C1	C1-1 ... C1-5	36	36	0.1	3600	400	[0,300]	4	C1-1 ... C1-5
	C1-6 ... C1-10			0.3		1542			
	C1-11 ... C1-15			0.5		3600			
C2	C2-1 ... C2-5	72	36	0.1	3600	400	[0,300]	4	C2-1 ... C2-5
	C2-6 ... C2-10			0.3		1542			
	C2-11 ... C2-15			0.5		3600			

Table 3 Performance comparison between (e_1, e_2) combinations

e_1	e_2						
	1 (%)	2 (%)	3 (%)	4 (%)	5 (%)	6 (%)	7 (%)
1	1.25	1.04	0.43	0.34	0.32	0.28	0.28
2	1.14	0.98	0.21	0.23	0.26	0.31	0.31
3	1.12	0.73	0.09	0.06	0	0.08	0.17
4	0.97	0.71	0.14	0.08	0.04	0.18	0.21
5	1.05	0.68	0.12	0.07	0.05	0.17	0.28

5.2.2 Calibration of the neighborhood selection probabilities

The neighborhood selection probabilities are adjusted based on the evolution of two parameters, IT_{cNS}, the number of consecutive iterations without improvement of the best solution, triggering the execution of the *Control* procedure that modifies the probabilities, and $\Delta\overline{r}$, the amplitude of the adjustment of the neighborhood-selection parameter $\overline{r}$.

Intuitively, the value of IT_{cNS} should be large enough to give each customer and supply point in each leg the possibility to be involved in a move. We therefore define it as a function of the problem size, $IT_{cNS} = e_1 * (m' * |\mathcal{S}| + n)$, where m' is the number of vehicles used in the initial solution, $|\mathcal{S}|$ and n are the numbers of supply points and customer demands, respectively, and e_1 is a user defined parameter. Similarly, $\Delta\overline{r}$ should gradually guide the algorithm towards a more frequent use of routing neighborhoods. We thus define it proportional to the ratio of the number of customer demands relative to the number of supply points, i.e., $\Delta\overline{r} = e_2 \log_{10}(n/|S|)$, where e_2 is a user defined parameter.

Searching for a good combination of values for e_1 and e_2 concerns balancing the search between exploration and exploitation. Thus, e.g., 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, however, too high a IT_{cNS} value may waste time in useless moves. We have experimented with different combinations of the e_1 and e_2 parameters, varying e_1 in the integer interval [1,5] and e_2 in the integer interval [1, 7], as shown in Table 3. Three runs, with one million iterations for each, were performed for each combination of values (e_1, e_2), and the best solution out of the three was collected. We then selected the “best” combination, that is, the one giving the best solutions on average over all instances. Computational results are summed up in Table 3 displaying the average gaps between the values of the best solutions obtained by each combination and the selected best combination.

Table 3 indicates that (3,5) is the most appropriate combination for (e_1, e_2), giving best solutions on average. We have also observed that executing the algorithm with $\overline{r}$ greater than 60 $\log_{10}(n/|S|)$ yields an average improvement of the best solution of <0.1 %, while requiring about 41 % more time. Based on these results, we used (e_1, e_2) = (3, 5) and $\overline{r}_{max}$ $= 60 \log_{10}(n/|S|)$, the maximum value of $\overline{r}$, in the remaining experiments.

5.2.3 Diversification and the elite set

We now turn to the parameters characterizing the diversification procedure and the elite set utilization, and examine their impact on the performance of the algorithm. Four variants of the algorithm were studied corresponding to the different ways to set up 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 z at random and re-start the algorithm from it.

Table 4 Performance comparison between diversification settings

Elite set size	Without diversification				With diversification			
	1st variant		2nd variant		3rd variant		4th variant	
	$r = r_z$		$r = r_z/2$		$r = r_z$		$r = r_z/2$	
	GAP (%)	Time	GAP (%)	Time	GAP (%)	Time	GAP (%)	Time
0	0	50	–	–	–	–	–	–
1	−0.37	66	−0.36	92	−1.02	88	−1.05	103
5	−0.64	95	−0.69	117	−1.54	157	−1.48	194
10	−0.78	121	−0.74	139	−1.55	223	−1.50	260

The *Diversification* mechanism described in Sect. 4.8 is applied in the last two variants to diversify from the elite solution z.

The initialization of the $\overline{r}$ parameter following the selection of z is common to the four variants. We have studied two alternatives where $\overline{r}$ was set to either the full or half the value at which z was found, respectively (i.e. $\overline{r} = \overline{r}_z$ or $\overline{r} = \overline{r}_z/2$). 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 have 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 customer demands, respectively.

We have also set the number of consecutive executions of the *Control* procedure without improvement of the best solution to $C_{cNS} = min(3\log_{10}(n/|S|), (\overline{r}_{max} - \overline{r})/\Delta_{\overline{r}})$, 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 $\overline{r}$ (remember that $\overline{r}_{max} = 60\log_{10}(n/|S|)$). Intuitively, in the beginning, $\overline{r}$ is small and C_{cNS} takes the value $3\log_{10}(n/|S|)$, while when $\overline{r}$ becomes large enough, C_{cNS} takes the value $(\overline{r}_{max} - \overline{r})/\Delta_{\overline{r}}$.

Table 4 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 value of the best solutions obtained by it from those obtained without using the elite set and diversification, together with the corresponding average computation time in minutes over 10 runs.

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 worst solutions compared to all the variants using the elite set. Comparing the two variants corresponding to the two values at which $\overline{r}$ is reset, one observes that the solution quality is not very sensitive to this value, but the computing effort is increasing when the value of $\overline{r}$ is lower ($\overline{r} = \overline{r}_z/2$).

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 the 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, 0.01 %, but requires 42 % more time. We therefore set the size of the elite set to 5 and reset $\overline{r} = \overline{r}_z$.

5.3 Tabu Search performance

Table 5 displays the results obtained by the proposed tabu search meta-heuristic over 10 runs for each group of instances. It gives the average (Avg 10 column) and best (Best 10 column) objective value, the number of vehicles (Number Vehicles column), the percentage of times vehicles move directly to supply points without using waiting stations (DM (%) column), and the percentage of times vehicles perform *unload & load* operations once they arrive at supply points (PD (%) column). Average computation times in minutes are displayed in the Time column.

The experimental results in Table 5 show that, overall, 4874 vehicles are used in the 90 problem instances, servicing a total of 39,790 legs. Hence, on average, each vehicle services 8 legs. Table 5 also shows that the percentage of times vehicles perform *unload & load* operations increases proportionally to the percentage of pickup-customer demands (i.e., the value of BH). On average, in almost 50 % of the cases, vehicles perform both unloading and loading once they arrive at supply points. The number of *unload & load* operations at supply points not only reduces the number of empty moves but also reduces the traveling cost. Moreover, experiments show that the traveling cost and the number of vehicles in the initial solutions are 32.45 and 20.76 % greater than those of the best solutions on average, respectively, illustrating the significant solution-improvement effect of the proposed algorithm. The Appendix in Supplementary material provides detailed results.

Table 5 Performance of Tabu Search on all instances

Instance set	BH	Avg 10	Best 10	Number vehicles	DM (%)	PD (%)	Time (min)
A1	0.1	19,873.29	19,758.67	21.8	10.45	21.89	20
	0.3	21,007.60	20,854.25	22	27.44	60.93	34
	0.5	23,455.87	23,245.62	22.2	51.29	87.1	58
A2	0.1	16,884.05	16,756.85	16.4	14.77	21.52	12
	0.3	18,462.56	18,295.76	16.4	31.75	56.75	19
	0.5	21,150.77	20,981.06	17.2	45.28	88.05	33
B1	0.1	66,979.79	66,763.80	46.8	19.33	15.01	66
	0.3	75,587.05	75,398.22	47.8	31.3	46.73	139
	0.5	99,155.77	99,025.96	54.8	38.31	80.39	231
B2	0.1	59,828.68	59,717.48	36.4	19.06	16.53	42
	0.3	72,098.73	71,945.56	40	23.64	46.76	97
	0.5	94,024.35	93,838.52	46	32.63	78.41	198
C1	0.1	153,335.20	153,106.40	90.4	17.65	13.84	172
	0.3	200,072.40	199,848.80	99.4	21.78	46.01	310
	0.5	292,032.84	291,836.60	119.8	30.91	82.58	705
C2	0.1	141,018.12	140,803.04	76.2	18.26	15.65	112
	0.3	195,573.18	195,206.00	94.4	24.59	41.92	213
	0.5	278,354.82	278,058.20	106.8	26.45	77.77	348
Average		102,716.39	102,524.49	54.16	26.94	49.88	156.06

Table 6 Performance comparison for the MZMT-VRPTW-DC

Data	$\lvert S\rvert$ set	Cust/ zone	Customer demands	GAP to lower bound (%) BCP	GAP to lower bound (%) TS	CNV/CTD BCP	CNV/CTD TS
D1	5	5	50	0	0.51	38/20304.68	38/20504.54
D2	5	7	70	0	1.00	64/23791.44	63/24855.07
D3	5	9	90	0.6	1.95	43/21194.93	43/21746.44
D4	5	6	60	0	0.96	43/22979.74	42/23901.18
D5	10	6	120	0	2.57	64/45367.54	59/49880.16
D6	15	6	180	0	3.29	72/61900.42	69/66589.68
Average	8	7	95	0.1	1.71	252/133638.33	245/140887.39

5.4 Comparison with results in the literature

The MT-PDTWS is considered for the first time in the literature and there are no previous results to compare to. Therefore, in order to provide an assessment of the performance of the proposed algorithm, we run it on instances of the MZMT-VRPTW-DC and the VRPB, and compared the results of the proposed tabu search algorithm to results available in the literature for these two problems.

5.4.1 Comparison with the MZMT-VRPTW-DC

The comparison with the branch-and-cut-and-price algorithm proposed by Bettinelli et al. (2015) for the MZMT-VRPTW-DC required slightly modifying both problem settings to merge them into a single one the two algorithms could address in a relatively straightforward manner. Two modifications were made to the MZMT-VRPTW-DC problem setting: 1) vehicles are not allowed to stop at waiting stations when moving between two customer demands, i.e., vehicle must go directly from one customer demand to another; and 2) vehicles cannot wait at supply points. The modifications to the MT-PDTWS were: 1) each pick-up customer demand is pre-assigned to a supply point; 2) the departure time from each supply point s is fixed to $t(s)$; and 3) the waiting time is included into the objective function.

Both algorithms were then applied to the 60 instances proposed in Bettinelli et al. (2015) and grouped into six subsets (D1 to D6) of 10 instances each. The best solutions obtained over 10 runs of the tabu search algorithm are used for the performance comparisons displayed in Table 6. Columns 2 to 4 describe the data sets (identified in the first column): the number of supply points, the number of pickup or delivery customer-demands per supply point, and the total number of customer demands, respectively. The rest of the table consists of two major columns, each divided into two sub-columns, one for the Branch-and-Cut-and-Price (Bettinelli et al. 2015) and the other for the tabu search algorithm proposed in this paper. The column GAP to lower bound (%) displays the average gaps between the best solutions obtained and the lower bounds identified by Bettinelli et al. (2015), while column CNV/CTD displays the cumulative number of vehicles (CNV) and the cumulative total distance (CTD) for the best solutions obtained of each instance set.

All instances, except 3 instances of D3, were solved to optimality by the Branch-and-Cut-and-Price. The optimality gaps for the three unsolved instances (D3-02, D3-06 and D3-10), were 1.47, 0.97, and 3.56 %, respectively, yielding an average gap of 0.6 % for the 10 instances of the set D3. The tabu search algorithm yielded solutions with an average optimality gap of 1.71 %, which supports the high-performance claim. As expected, the difference grows with the problem dimensions, but stays within very reasonable margins. It is revealing to try explore where the difference comes from. This may be inferred from by examining the performances in terms of number of vehicles and total distance. The results displayed in the last column clearly indicate that the tabu search targets more the former while the Branch-and-Cut-and-Price aims to minimize the latter. On average for instances of sets D5 and D6 (with 10 and 15 supply points, respectively), the tabu search produces solutions with 5.88 % lower number of vehicles for, but requiring 8.57 % higher operating cost. Such a behavior is compatible with the objective of City Logistics systems, which was the initial motivation of our work.

5.4.2 Comparison with the VRPB

We compared the performance of the proposed tabu search algorithm with existing algorithms in the literature for the VRPB, with and without time windows. Recall that in the VRPB vehicles perform a single tour delivering first, and picking up on the "return" path to the depot. There are no multi-tours, no synchronization (no waiting stations), and no need to determine the assignment of pickup-customer demands to supply points. We therefore discarded all parameters and algorithmic components related to these characteristics, runing the tabu search using the routing neighborhoods only.

The *Vehicle Routing problem with Backhauls and Time windows* (VRPBTW) considers time windows at customers and limits on the duration of routes. Experiments were carried out on the 15 VRPBTW 100-customer instances proposed by Gélinas et al. (1995), broadly used in the literature. The results of the proposed tabu search meta-heuristic are compared to *GDDS95*, the branch-and-bound method based on column generation proposed by Gélinas et al. (1995), which found optimal solutions to 6 test problems; PDG96, the heuristic proposed by Potvin et al. (1996), which first uses a genetic algorithm to identify an ordering of customers that produces good routes, and then greedily builds routes by inserting customers into routes based on this ordering; *TPS96*, the construction followed by improvement heuristic (λ-interchange and 2-opt*) of Thangiah et al. (1996); *RDH02*, the ant system approach (with only global pheromone updating) of Reimann et al. (2002); *ZC05*, the two-phase heuristic of Zhong and Cole (2005), which first clusters customers, and then improves routes (2-opt, 1-move, 1-exchange) within a guided local search framework; *RU06*, the ant colony optimization of Reimann and Ulrich (2006); *RP06*, the large neighborhood search of Ropke and Pisinger (2006); and *VCGP14*, the unified hybrid genetic algorithm proposed by Vidal et al. (2014) for a very broad set of vehicle routing problem settings.

Table 7 displays the results of the comparison for each of the 15 instances and each competing algorithm, in terms of the number of vehicles and the total travel distance of the best reported solutions. The 15 instances are divided into five groups, R101, R102, R103, R104, and R105, with three different percentages of backhaul customers (%BH) in each group. In the bottom group of rows, CNV and CTD, indicate the cumulative number of vehicles and the cumulative total distance over the 15 instances, respectively. The last three rows provide average measures over all instances: the original computation time, the scaled computational time, using the Dongarra (2014) factors and our machine (Xeon 2.8 GHz) as the baseline, and the type of processor used by each algorithm. Times are in CPU minutes.

Most algorithms in the literature (except Gélinas et al. 1995) aim to first reduce the number of vehicles, while there are no vehicle fixed costs in the VRPBTW instances. On the other hand, the MT-PDTWS formulation minimizes the generalized cost of the system, vehicle

Table 7 Performance comparison for the VRPBTW

Instance	% BH	GDDS95	TPS96	PDG96	RDH02	ZC05	RU06	RP06	VCGP14	TS
			Best 5 versions	Best	Best 2 versions	Best 5 runs	Best	Best 10 runs	Best 10 runs	Best 10 runs
R101	10	–	24	23	22	24	22	22	22	22
		1767.9	1842.3	1815.0	1831.68	1848.04	1853.45	1818.86	1818.86	1823.64
	30	–	24	23	23	24	23	23	23	24
		1877.6	1928.6	1896.6	1999.16	2034.61	1985.23	1959.56	1959.52	1903.21
	50	–	25	24	24	25	24	24	24	24
		1895.1	1937.6	1905.9	1945.29	2057.05	1964.04	1939.1	1939.1	1917.87
R102	10	–	20	20	19	–	19	19	19	19
		1600.5	1654.1	1622.9	1677.62	–	1663.16	1653.19	1653.18	1665.93
	30	–	21	20	22	–	22	22	22	22
		1639.2	1764.3	1688.1	1754.43	–	1759.02	1750.7	1750.7	1758.31
	50	–	21	21	22	–	22	22	22	22
		1721.3	1745.7	1735.7	1782.21	–	1782.91	1775.76	1775.76	1781.46
R103	10	–	15	16	16	–	15	15	15	15
		–	1371.6	1343.7	1348.41	–	1454.25	1387.57	1385.38	1397.02
	30	–	16	15	16	–	15	15	15	15
		–	1477.6	1381.6	1395.88	–	1407.29	1390.33	1390.32	1382.08
	50	–	17	17	17	–	17	17	17	17
		–	1543.2	1456.6	1467.66	–	1478.48	1456.58	1456.48	1471.43

Table 7 continued

Instance	%BH	GDDS95	TPS96	PDG96	RDH02	ZC05	RU06	RP06	VCGP14	TS
			Best 5 versions	Best	Best 2 versions	Best 5 runs	Best	Best 10 runs	Best 10 runs	Best 10 runs
R104	10	–	13	12	11	–	11	11	10	11
		–	1220.3	1117.7	1205.78	–	1153.06	1084.17	1204.57	1102.21
	30	–	12	12	12	–	11	11	11	11
		–	1302.5	1169.1	1128.3	–	1228.62	1154.84	1154.84	1181.17
	50	–	13	13	12	–	11	11	11	12
		–	1346.6	1203.7	1208.46	–	1306.97	1191.38	1190.2	1204.59
R105	10	–	17	17	16	17	16	15	15	16
		–	1553.4	1621	1544.81	1590.54	1570.11	1561.28	1560.15	1571.42
	30	–	18	16	16	17	16	16	16	16
		–	1706.7	1652.8	1592.23	1667.92	1646.11	1583.3	1583.3	1586.66
	50	–	18	18	17	19	17	16	16	17
		–	1657.4	1706.7	1633.01	1699.88	1689.74	1710.19	1709.66	1648.32
CNV		–	274	267	265	–	261	259	258	263
CTD		–	24,051.9	23,317.1	23,514.93	–	23,942.44	23,416.81	23,532.02	23,395.32
Avg origin time(min)		–	0.35	3.37	2.50	–	1.25	1.90	4.10	1.74
Avg scaled time(min)		–	0.003	0.08	0.63	–	0.51	0.80	2.87	1.74
Processor		–	NeXT 33MHz	Sun Sparc 10	P3 900MHz	P2 450MHz	P4 1.5GHz	P4 1.5GHz	Opt 2.2GHz	Xeon 2.8GHz

Table 8 Performance with different values of vehicle fixed cost on the VRPBTW instances

Fixed cost of vehicle	CNV	CTD	GAP (%)
0	263	23,395.32	0
$\bar{F}$	261	23,965.28	2.44
$1.1 * \bar{F}$	261	23,978.43	2.49
$1.2 * \bar{F}$	261	23,974.53	2.48
$1.3 * \bar{F}$	261	23,983.16	2.51
$2 * \bar{F}$	261	23,996.17	2.57
$10 * \bar{F}$	260	24,233.06	3.58
$100 * \bar{F}$	260	24,256.24	3.67
$1000 * \bar{F}$	260	24,275.62	3.76

fixed usage costs plus routing cost, and the tabu search we propose does not aim to enforce one dimension over the other. Applying the tabu search to the VRPBTW instances therefore corresponds to minimizing the routing cost only, without taking into account the number of vehicles. The proposed meta-heuristic proves to be very competitive with respect to the total distance, outperforming five meta-heuristics and being very close to the best ones (with an average gap of 1.00 %, a maximal gap of 2.81 % and a minimal gap of −0.34 %). We run a second series of tests to better understand the role of the vehicle fixed cost on reducing the number of vehicles. We set the vehicle fixed cost to a multiple of the average arc cost $\bar{F}$, and repeatedly solved the VRPBTW instances increasing this multiplying factor. Table 8 displays the results for each value of the vehicle fixed cost: the number of vehicles (CNV) and cumulative total distance (CTD) over the 15 instances, as well as the increase (in %) in the total distance with respect to the case without fixed costs. The results show that increasing the fixed cost on the use of vehicles, decreases the number of vehicles used and increases the total distance traveled. This corresponds to what is generally observed in VRP solutions, as is the observation that increasing the fixed cost yields decreasing returns passed a certain threshold (equal to $10 * \bar{F}$ for these problem instances). The proposed tabu search yields now solutions that are very close (<0.58 % of average gap) to those of the best methods in the literature for the VRPBTW (Ropke and Pisinger 2006; Vidal et al. 2014), which is remarkable for a solution method not designed for the particular problem setting.

The next round of experiments focused on the ***Vehicle Routing problem with Backhauls*** (VRPB), which is obtained by removing from the VRPBTW the constraints on time windows at customers and the route duration. The performance of our tabu search is evaluated through comparisons with results of other tabu search algorithms on two sets of instances in the VRPB literature. The first set of 62 instances was proposed in Goetschalckx and Jacobs-Blecha (1989). The instances range in size between 25 and 150 customers with backhauls ranging between 20 and 50 %. The second set of 33 instances was proposed by Toth and Vigo (1997), with the number of customers ranging between 21 and 100, and backhauls percentages of 20, 34 or 50 %. Two ways of computing the Euclidean distances between pairs of customers are used in the VRPB literature, namely real-valued and integer-valued, respectively. The former was used for the three tabu search algorithms with which we compare our method, and is therefore, used for our tabu search method as well.

Table 9 sums up the comparisons with respect to the average of the best solutions for the two sets of instances. The first two columns give the references and the processors used for each study. Then, in groups of four columns for each instance set, the table displays the average of the best solutions obtained by each method (Columns Cost), the gaps to the

Table 9 Performance comparison for the VRPB

Authors	Processor	Goetschalckx and Jacobs-Blecha (1989)				Toth and Vigo (1997)			
		Cost	GAP to BKS (%)	Avg time original (m)	Avg time scaled (m)	Cost	GAP BKS (%)	Avg time original (m)	Avg time scaled (m)
Osman and Wassan (2002)	Sun Sparc 1000	291261.7	0.25	67.1	1.6	708.42	1.09	26.5	0.7
Brandão (2006)	P3 500MHz	291160.5	0.21	13.6	2.3	702.15	0.19	5.0	0.9
Wassan (2007)	Sun Sparc 1000	290981.8	0.15	30.6	0.7	706.48	0.81	10.1	0.2
TS	Xeon 2.8GHz	290964.7	0.14	1.6	1.6	705.49	0.67	0.8	0.8

average of the best known solutions (Column GAP to BKS (%)), as well as the original and the scaled (to equal Xeon 2.8GHZ Dongarra 2014) CPU times in minutes. One observes that the proposed tabu search performs well, outperforming all three other tabu search algorithms on Goetschalckx and Jacobs-Blecha (1989) instances (with an average gap of 0.05 % and a maximal gap of 0.1 %), and two out of three methods on Toth and Vigo (1997) instances, and being not far from Brandão (2006) (with a gap of −0.47 %). This is, again, remarkable for a solution method not designed for the particular problem setting.

5.5 Combining linehauls and backhauls

Combining linehaul and backhaul-customer demands on each vehicle is expected to reduce the total number of vehicles and the total travel cost with respect to the case where linehauls and backhauls are serviced by separate vehicles. Table 10 compares the best solutions for these alternatives for all instances over 10 runs. The LH-BH columns report results for the combined service case: average number of vehicles (Column #Vehicles), travel cost (Travel cost), and total cost (Total cost). The LH+BH columns refer to the summing the solutions to the problems with only linehaul- and only backhaul-customer demands, respectively. Each entry under this heading gives the gap with respect to the corresponding LH-BH measure (average number of vehicles, and average travel and total cost). As expected, the results indicate that assigning linehauls and backhauls to separate fleets leads to significant increases in all

Table 10 Comparison of separate and combined linehaul and backhaul solutions in the number of vehicles, traveling cost, and total cost

Problem set	BH	LH−BH			LH+BH		
		#Vehicles	Travel cost	Total cost	GAP (%)		
A1	0.1	21.8	8,858.67	19,758.67	12.84	9.08	11.16
	0.3	22	9,854.25	20,854.25	45.45	26.25	36.38
	0.5	22.2	12,145.62	23,245.62	89.19	38.06	62.48
A2	0.1	16.4	8,556.85	16,756.85	13.41	10.14	11.74
	0.3	16.4	10,095.76	18,295.76	35.37	21.91	27.94
	0.5	17.2	12,381.06	20,981.06	69.77	34.57	49.00
B1	0.1	46.8	43,363.80	66,763.80	8.55	11.52	10.48
	0.3	47.8	51,498.22	75,398.22	35.98	29.50	31.55
	0.5	54.8	71,625.96	99,025.96	48.18	33.71	37.71
B2	0.1	36.4	41,517.48	59,717.48	10.99	10.65	10.76
	0.3	40	51,945.56	71,945.56	38.00	28.73	31.31
	0.5	46	70,838.52	93,838.52	57.39	29.76	36.53
C1	0.1	90.4	107,906.40	153,106.40	0.22	14.68	10.41
	0.3	99.4	150,148.80	199,848.80	24.55	28.87	27.79
	0.5	119.8	231,936.60	291,836.60	39.07	26.11	28.77
C2	0.1	76.2	102,703.04	140,803.04	0.79	22.56	16.67
	0.3	94.4	148006.00	195,206.00	10.59	31.48	26.43
	0.5	106.8	224,658.20	278,058.20	35.39	29.00	30.23
Average		54.16	75,446.71	102,524.49	31.98	24.25	27.63

performance measures. This increase becomes increasingly significant when more backhauls need service.

5.6 Synchronization at supply points

We model supply points as combinations of a satellite and a time period availability. The vehicles must thus arrive at supply points during these predefined periods to unload or load freight. We analyze in this section the impact on solution quality of this synchronization requirement on operations.

The time period availability at each supply point s was characterized in all previous experiments by a single time window $[e_s, l_s]$ used for both unloading and loading operations. In order to analyze the impact of the availability requirements without modifying the time windows at customer demands, we introduce two time windows at each supply point, one for unloading and one for loading, but keep the availability time periods of the supply points unchanged. More precisely, we define $[e_s^u, l_s^u]$ and $[e_s^l, l_s^l]$, specifying the earliest and latest times at which a vehicle has to be available at s for unloading collected demands and loading delivery demands, respectively, where $l_s^u + \varphi'(s) \leq l_s^l$, $e_s^u = e_s$ and $l_s^l = l_s$. Activities of a vehicle at s may then be described as follows:

- Unload only. The vehicle arrives with pickup demands at time t within its unloading time window $[e_s^u, l_s^u]$, i.e., the vehicle must not arrive at s sooner than e_s^u nor later than l_s^u; it takes $\varphi'(s)$ for unloading and leaves s empty at time $t + \varphi'(s)$;
- Load only. The vehicle arrives empty at time t within its loading time window $[e_s^l, l_s^l]$, i.e., the vehicle must not arrive at s sooner than e_s^l nor later than l_s^l; it takes $\varphi(s)$ for loading the delivery demands, with a total load not exceeding the vehicle's capacity Q, and leaves s at time $t + \varphi(s)$ to perform the delivery to a subset of delivery customers in $\mathcal{C}_s^D$;
- Unload and load. The vehicle arrives with pickup demands at time t within its unloading time window $[e_s^u, l_s^u]$ and takes $\varphi'(s)$ to unload; when $t + \varphi'(s) < e_s^l$, the vehicle has to wait at the supply point until e_s^l to start loading freight; otherwise it starts loading at $t + \varphi'(s)$; it takes $\varphi(s)$ to loading, then the vehicle leaves s to deliver the loaded freight to a subset of customers in $\mathcal{C}_s^D$.

Operations at a supply point s are then guided by the length of each time window, noted len_u and len_l for unloading and loading, respectively, and the separation time Dif between the end of the unloading time window l_s^u and the beginning of the loading time window e_s^l (see Fig. 9; $Dif = 0$ when the two time windows split equally the total activity time). Recalling that the activity time is 100 in the case of the single time window, we set $len_u = len_l$, and run three experiments with values (20, 60), (30, 40) and (40, 20) for $(len_u = len_l, Dif)$.

The experiment was run on all instances and Table 11 sums up the impact on solution quality of splitting the operation times at supply points, for each of the three cases compared

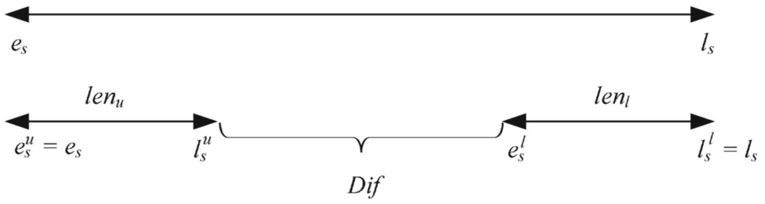

Fig. 9 Illustration of two time windows at a supply point s

Table 11 Impact of synchronization at supply points on solution quality

	One time window	Two time windows		
		(20,60)	(30,40)	(40,20)
#Vehicles (%)	0	1.03	0.79	0.52
Travel cost (%)	0	2.17	0.82	0.74
Total cost (%)	0	1.94	0.88	0.75
PD (%)	49.88	45.40	46.98	48.31
DM (%)	26.94	22.45	23.95	24.01

to the base case of a single time window and no separation of operations. The table displays the increase, in %, in the number of vehicles, travel cost, and total cost. It also gives the percentage of times the vehicles perform *unload & load* operations at supply points (*PD(%)* row) and the percentage of times vehicles move directly to a supply point without using waiting stations (*DM(%)* row).

Results clearly indicate that solutions with two time windows are worse than those with a single time window with respect to all performance measures. Allowing to mix operations at supply points results, in particular, in vehicles moving directly to supply points more frequently and undertaking more *unload & load* operations (26.94 and 49.88 % respectively, for the single time window case). Such operations may result in more complex operations management at supply points, but increases efficiency and decreases the presence of vehicles within the system. This could be very beneficial in many cases, City Logistics in particular.

6 Conclusions

We introduced the Multi-trip Pickup and Delivery Problem with Time Windows and Synchronization, MT-PDTWS, a new class of vehicle routing problems variant in which each vehicle performs multiple sequences of delivery and pickup operations through supply points within hard time synchronization restrictions. The MT-PDTWS generalizes several classes of pickup and delivery with backhauls problem settings, as well as a number of problems defined within City Logistics applications. We proposed a first model formulation and a tabu search meta-heuristic integrating multiple neighborhoods for the MT-PDTWS.

The computational study was performed on a new set of instances with up to 72 supply points and 7200 customer demands. By restricting the model, the tabu search meta-heuristic was also compared to exact and meta-heuristic methods for the pickup and delivery with backhauls problems with and without time windows. Test instances present in the literature for the latter problems were used for this evaluation. Our experiments showed that the proposed meta-heuristic performs very well, being competitive with the other methods within their particular settings, and efficiently addressing all the new instances.

The tabu search meta-heuristic provided the tools to evaluate a number of problem characteristics, in particular the value of servicing pickup and delivery customers with the same vehicle and routes, as well as strategies in setting up the service time windows at supply points. The experiments revealed that integration of customer types within a single service is beneficial, as is the integration of the two types of operations within the activity period of supply points.

Acknowledgements While working on this project, the first author was doctoral student with the Computer Science and Operations Research Department, Université de Montréal, and the second author was Adjuct Professor with the same department. The authors thank Professor Daniel Vigo and Dr. Andrea Bettinelli for their great kindness, cooperation and efforts to adjust their code and perform the exact-resolution part of the comparison experiments. Partial funding for this project comes from the Discovery Grant and the Discovery Accelerator Supplements Programs of the Natural Science and Engineering Research Council of Canada, and the Strategic Clusters program of the Fonds québécois de la recherche sur la nature et les technologies. The authors thank the two institutions for supporting this research.

References

Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I., & Laporte, G. (2007). Static pickup and delivery problems: A classification scheme and survey. *TOP*, *15*, 1–31.

Berbeglia, G., Cordeau, J.-F., & Laporte, G. (2010). Dynamic pickup and delivery problems. *European Journal of Operational Research*, *202*, 8–15.

Bettinelli, A., Crainic, T.G., & Vigo, D. (2015). The multi-zone multi-trip vehicle routing problem with separate delivery and collection. Publication, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, Université de Montréal, Montréal, QC, Canada. forthcoming.

Brandão, J. (2006). A new tabu search algorithm for the Vehicle Routing Problem with backhauls. *European Journal of Operational Research*, *173*(2), 540–555.

Cordeau, J.-F., Laporte, G., & Mercier, A. (2001). A unified Tabu Search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, *52*, 928–936.

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

Crainic, T. G., Errico, F., Rei, W., & Ricciardi, N. (2012). Integrating c2e and c2c Traffic into City logistics planning. *Procedia Social and Behavioral Sciences*, *39*, 47–60.

Dell'Amico, M., Righini, G., & Salani, M. (2006). A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. *Transportation Science*, *40*(2), 235–247.

Dethloff, J. (2002). Relation between vehicle routing problems: An insertion heuristic for the vehicle routing problem with simultaneous delivery and pick-up applied to the vehicle routing problem with backhauls. *Journal of the Operational Research Society*, *53*(1), 115–118.

Dongarra, J.J. (2014). Performance of various computers using standard linear equations software. Technical report, University of Tennessee.

Gélinas, S., Desrochers, M., Desrosiers, J., & Solomon, M. (1995). A new branching strategy for time constrained routing problems with application to backhauling. *Annals of Operations Research*, *61*(1), 91–109.

Goetschalckx, M., & Jacobs-Blecha, C. (1989). The Vehicle Routing Problem with backhauls. *European Journal of Operational Research*, *42*(1), 39–51.

Gribkovskaia, I., Halskau, O., Myklebost, K. (2001). Models for pick-up and deliveries from depots with Lasso solutions. In *Proceedings of the 13th annual conference on logistics research* (pp. 279–293). NOFOMA 2001, Collaboration in logistics: Connecting Islands using Information Technology.

Lee, Y. H., Jung, J. W., & Lee, K. M. (2006). Vehicle routing scheduling for cross-docking in the supply chain. *Computers & Industrial Engineering*, *51*(2), 247–256.

Liao, C.-J., Lin, Y., & Shih, S. C. (2010). Vehicle routing with cross-docking in the supply chain. *Expert Systems with Applications*, *37*(10), 6868–6873.

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

Nagy, G., & Salhi, S. (2005). Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. *European Journal of Operational Research*, *162*(1), 126–141.

Nguyen, P. K., Crainic, T. G., & Toulouse, M. (2013). A tabu search for time-dependent multi-zone multi-trip vehicle routing problem with time windows. *European Journal of Operational Research*, *231*(1), 43–56.

Or, I. (1976). *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.

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

Osman, I. H., & Wassan, N. (2002). A reactive tabu search meta-heuristic for the vehicle routing problem with back-hauls. *Journal of Scheduling*, *5*(4), 263–285.

Parragh, S., Doerner, K., & Hartl, R. (2008a). A survey on pickup and delivery problems. Part I: Transportation between customers and depot. *Journal für Betriebswirtschaft*, *58*, 21–51.

Parragh, S., Doerner, K., & Hartl, R. (2008b). A survey on pickup and delivery problems. Part II Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft*, *58*, 81–117.

Potvin, J.-Y., Duhamel, C., & Guertin, F. (1996). A genetic algorithm for vehicle routing with backhauling. *Applied Intelligence*, *6*(4), 345–355.

Reimann, M., & Ulrich, H. (2006). Comparing backhauling strategies in vehicle routing using ant colony optimization. *Central European Journal of Operations Research*, *14*(2), 105–123.

Reimann, M., Doerner, K., & Hartl, R. (2002). Insertion based ants for vehicle routing problems with backhauls and time windows. In M. Dorigo, G. Caro, & M. Sampels (Eds.), Ant algorithms volume 2463 of lecture notes in computer science (pp. 135–148). Berlin: Springer.

Ropke, S., & Pisinger, D. (2006). A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, *2004*, 750–775.

Salhi, S., & Nagy, G. (1999). A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. *Journal of the Operational Research Society*, *50*(10), 1034–1042.

Savelsbergh, M. W. P., & Solomon, M. M. (1995). The general pickup and delivery problem. *Transportation Science*, *29*, 17–29.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, *35*, 254–265.

Taillard, E. D., Badeau, P., Gendreau, M., Guertin, F., & Potvin, J.-Y. (1997). A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, *31*, 170–186.

Thangiah, S. R., Potvin, J.-Y., & Sun, T. (1996). Heuristic approaches to vehicle routing with backhauls and time windows. *Computers & Operations Research*, *23*(11), 1043–1057.

Toth, P., & Vigo, D. (1997). An exact algorithm for the vehicle routing problem with backhauls. *Transportation Science*, *31*(4), 372–385.

Toth, P., & Vigo, D. (2002). *The vehicle routing problem*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. (2014). A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research*, *234*(3), 658–673.

Wassan, N. (2007). Reactive tabu adaptive memory programming search for the vehicle routing problem with backhauls. *Journal of the Operational Research Society*, *58*(12), 1630–1641.

Wen, M., Larsen, J., Clausen, J., Cordeau, J.-F., & Laporte, G. (2008). Vehicle routing with cross-docking. *Journal of the Operational Research Society*, *60*, 1708–1718.

Zhong, Y., & Cole, M. H. (2005). A vehicle routing problem with backhauls and time windows: A guided local search solution. *Transportation Research Part E: Logistics and Transportation Review*, *41*(2), 131–144.