The Benefits of Transfers in Crowdsourced Pickup-and-Delivery Systems

Afonso Sampaio [1], Martin Savelsbergh[2], Lucas P. Veelenturf[1], and Tom Van Woensel[1]

[1]Eindhoven University of Technology
[2]Georgia Institute of Technology

November 29, 2018

Abstract

Rapid urban growth, the increasing importance of e-commerce and high consumer service expectations have given rise to new and innovative models for freight delivery within urban environments. *Crowdsourced* solutions – where drivers are not employed by a carrier but occasionally offer their services through on-line platforms and are contracted as required by carriers – are receiving growing attention from industry. We consider a crowdsourced system where drivers express their availability to perform delivery tasks for a given period of time and the platform communicates a schedule with requests to serve. We investigate the potential benefits of introducing transfers to support driver activities. At transfer locations, drivers can drop off packages for pick up by other drivers at a later time. We frame the problem as a Multi-Depot Pickup and Delivery Problem with Time Windows and Transfers, and propose an Adaptive Large Neighborhood Search algorithm that effectively identifies beneficial transfer opportunities and synchronizes driver operations. Computational experiments indicate that introducing transfer options can significantly reduce system-wide travel distance as well as the number of drivers required to serve a given set of requests, especially when drivers have short availability and requests have high service requirements.

Keywords— crowdshipping, pickup and delivery, transfers, large neighborhood search

1 Introduction

When delivering goods in urban areas, transportation companies face many challenges, such as traffic congestion and access restrictions. Given the increasing share of e-commerce in retail sales and the continued growth of urbanization, these challenges have an increasing impact on the ability of carriers to satisfy customer demands and many companies are, therefore, exploring innovative ideas and concepts to overcome these challenges. One of these ideas is "crowdshipping", where transportation capacity is provided by individuals willing to modify

their planned trips or willing to provide their services for a period of time. In the latter case, a platform (e.g., Uber Freight or Amazon Flex) matches drivers with companies having items to transport. For an overview of crowded-based applications in transportation and logistics, see [Buldeo Rai et al., 2017, Sampaio et al., 2018].

Incorporating crowdshipping as a means to improve B2C delivery and to support high consumer service levels is explored by various companies. As an example, e-tailer Zalando relies on Trunkrs to offer same-day delivery for its customers in certain cities in Europe. Trunkrs uses crowdsourced delivery, but also established courier services to provide the reliability demanded by its customers (the e-tailers). Walmart is considering another form of a crowdsourced store-to-door delivery service by having in-store customers fulfilling the delivery of items purchased by its on-line clients [Barr and Wohl, 2013]. These innovative business practices lead to new variants of vehicle routing problems and new optimization challenges.

We investigate the Pickup and Delivery Problem with Time Windows and Transfers (PDPTW-T) as it can be used to model a variety of crowdsourced delivery settings. At transfer locations, drivers can drop off packages for pick up by other drivers at a later time. The number and location of transfer points are strategic decisions and not considered in our study. A transfer location can be situated at accessible locations – facilitating the operations between drivers – such as gas stations, stores and supermarkets. It can also be an automated facility, such as a locker station, situated in one of those locations. Characterizing delivery settings that might benefit from the introduction of transfers has yet to be studied (to the best of our knowledge, this is only investigated by [Mitrović and Laporte, 2006]).

When the transfer of requests is not an option, the problem is a Pickup and Delivery Problem with Time Windows (PDPTW) and defined as follows. A fleet of vehicles is available to satisfy a set of transportation requests, each defined by a load to be transported (goods or people) by a single vehicle from a pickup location (origin) to a delivery location (destination) where the pickup has to occur after a given time and delivery has to take place before a given time. Multiple requests can be served by the same vehicle, as long as the vehicle capacity is never exceeded. When all requests are known in advance, the problem is referred as the static PDPTW; when requests arrive during the execution of planned routes and routes have to be updated to accommodate new requests, the problem is referred to as a dynamic PDPTW. The objective is to determine a set of vehicle routes serving all requests, where the function that drives the costs is chosen according to the application at hand (e.g., distance, duration, number of vehicles, or pollution [Savelsbergh and Sol, 1995]). The reader is referred to [Berbeglia et al., 2010] for a survey on dynamic pickup and delivery problems.

As mentioned above, transfer points are locations in the network where requests can be transferred between vehicles and temporarily stored. Hence, more than one vehicle can be used to serve a single request, e.g., a request may be picked up at its origin by one vehicle, then dropped at a transfer point where another vehicle (potentially with other characteristics) picks it up and delivers it at its destination. Note that direct transportation from the pick up location to the delivery location by the same vehicle is still possible. Transfer points allow, among others, service by a mixed fleet of vehicles ((electric) truck, van, or bike), but also allow integration of freight and passenger transportation. As any pickup and delivery problem can be seen as a special case of a pickup and delivery problem with transfers, pickup and delivery problems with transfers are NP-hard. A critical challenge when solving pickup and delivery problems with transfers is synchronization. Whereas allowing goods to be temporarily stored (or people to wait) at an intermediate location provides more flexibility and, thus, may allow

more effective use of resources, these resources need to be carefully synchronized in order to obtain a feasible solution.

Our contributions can be summarized as follows. We analyze the potential benefits of transfers in pickup and delivery operations in urban areas, focusing specifically on settings in which drivers operate short shifts (as is likely to happen in crowdshipping settings models). In such settings, the flexibility provided by transfers may allow serving long-distance requests that would otherwise be impossible. We propose a heuristic for the solution of instances of the multiple-depot PDPTW-T. The heuristic produces high-quality solutions in a reasonable amount of time. We compare the performance of the heuristic to that of a state-of-the art heuristic for the (multiple-depot) PDPTW.

The remainder of the paper is organized as follows. The relevant literature and related problems are discussed in Section 2. The notation used throughout the text is introduced in Section 3, as well as a formal problem definition of the problem and a mathematical programming formulation. The heuristic, an Adaptive Large Neighborhood Search heuristic, is introduced in Section 4, and the results of an extensive computational study are presented in Section 5. Finally, we present conclusions and directions for further research in Section 6.

2 Literature Review

The literature considering the use of transfer points in pickup and delivery problems can be divided in two groups: (1) pickup and delivery problems with transshipment (PDP-T) and (2) pickup and delivery with cross-docking (PDP-CD). Whereas both, transfers and cross-docks, act similarly in providing a consolidation mechanism in which short term storage for a limited amount of requests allows vehicles to potentially improve their loading plans, a few differences are worth mentioning. First, in a cross-docking system transportation requests are executed by a set of separate pickup and delivery routes. Inbound vehicles collect the items, bring them to the CD where they are consolidated, and then a set of outbound vehicles delivers them to their final destinations ([Maknoon and Laporte, 2017]). Transfer locations, on the other hand, can be seen as a possible transfer opportunity i.e., a request can be carried out either through transfers and, in this case, one vehicle collects the load and another one delivers it to its final destination, or the request can be served directly by the same vehicle. Second, the CD is the start and end location for each route in the plan, whereas a transfer can be any location where vehicles can exchange loads throughout their routes. Third, in some CD applications, consolidation activities can only be executed when all vehicles are in the CD. In the PDP-T, this is not required as long as synchronization requirements are met.

The static (i.e., all transportation requests are known in advance of the optimization) PDP variants have given rise to a substantial amount of research [Berbeglia et al., 2007]. Only recently the possibility of allowing transfers of goods (or people) is addressed in the literature. [Mitrović and Laporte, 2006] proposed a two-phase heuristic to solve the PDP-T consisting of a construction phase, in which a start solution is obtained, and an improvement phase defined by iteratively removing and inserting requests in a candidate solution. A first mathematical formulation and a branch-and-cut approach were proposed in [Cortés et al., 2010], but only small instances could be solved. Motivated by an air cargo carrier application, [Qu and Bard, 2012] introduced an insertion heuristic to identify profitable circumstances to exploit the transshipment option. The authors developed a GRASP algorithm and proposed a set of randomly generated instances. More recently, [Rais et al., 2014] proposed a new model for the problem,

distinguishing between vehicle (routes) and request flows and using multi-commodity flows to match these two. [Masson et al., 2013] proposed an adaptive large neighborhood (ALNS) algorithm for the problem, and tackled the Dial-a-Ride Problem (DARP) with transfers in [Masson et al., 2014]. A similar problem was proposed by [Ghilas et al., 2016], where requests are allowed to be transferred to/from scheduled lines such as bus, train and metro, operating between two terminals. In a recent survey from [Guastaroba et al., 2016], pickup and delivery problems with transfers are mentioned as one of the extensions of the PDP with cross-docking.

An increasing number of new start-ups within the shared-economy context, exploring different ways of monetizing underused assets e.g., cars, rooms, is giving rise to new strategic and operational issues that have to be addressed in order to support such systems. It is only recently that crowdsourced delivery systems have received academic attention. Based on the *crowdshipping* concept envisioned by Walmart, [Archetti et al., 2016] introduce the Vehicle Routing Problem with Occasional Drivers (VRPOD). Occasional drivers are in-store customers willing to fulfill the delivery for an online customer on their journey (performing a small detour, if necessary) after leaving the store. The authors highlight the challenges associated with designing appropriate compensation mechanisms and the importance of employing company drivers in order to ensure a certain service level. [Dayarian and Savelsbergh, 2017] explore using in-store customers to supplement company drivers to deliver dynamic incoming, on-line customer orders.

The use of intermediate locations in crowdsourced delivery systems, as an interface between company drivers and the crowd, is invstigated by [Kafle et al., 2017]. The authors propose a two-tiered delivery system, in which the second tier is crowdsourced to cyclists and pedestrians (the crowd). In the system, a truck carrier posts pickup and delivery requests on a platform and individuals in the crowd bid to carry out a subset of those requests. Parcels can be transferred between a truck and (one or more) individuals at relay locations. The company decides on the winning bids and plans the truck routes that visit the relay points and delivery addresses of requests for which no bids were received or were too expensive. [Chen et al., 2017b] introduce the the Multi-Driver Multi-Parcel Matching Problem (MDMPMP), in which parcels may be transported by a single or by multiple drivers, using existing planned routes of the drivers. Parcels can be transferred between drivers, allowing for a more flexible matching of drivers and parcels, since drivers do not need to fulfill the complete parcel's journey and use transfer opportunities to bring the parcel closer to its final destination.

[Behrend and Meisel, 2018] investigate this form of *crowdshipping* – where private drivers offer to execute delivery jobs for other people on trips they would make anyway – within item-sharing contexts, where assets e.g. tools, leisure equipment, can be temporarily rented. Whereas the prevalent practice on item-sharing platforms is that the transportation of an asset is delegated to the consumer (either the consumer actually performs the task or hire a courier company), the authors show that crowdshipping can increase profits and service levels for an item-sharing platform.

In another realization of the *crowdshipping* concept, drivers willing to perform transportation services use an online platform (Uber-Freight, Amazon Flex) and are matched to demand for such services in real-time. The drivers are independent, work for a given period of time and are paid in a hourly basis. To the best of our knowledge, there is no work dealing specifically with such scenarios. Nevertheless, in a similar context, taxi drivers might be willing, at times, to move freight within the city. [Li et al., 2014] introduce and explore the Share-a-Ride Problem taking into account different requirements to transport people and freight using a taxi

network. Taxis are allowed to deliver parcels as long as the service level for the passenger does not deteriorate significantly. A Freight Insertion Problem (FIP) is proposed to insert parcel collections in a given routing plan for passengers aimed at minimal passenger disruptions (e.g., maximum ride-time for passengers, maximum detours, number of stops). [Chen and Pan, 2016] refers to a "crowd of taxis" to propose, in the same vein as the solution for reverse flows in [Chen et al., 2017a], using the taxi fleet and a network of 24/7 shops to satisfy last-mile delivery requests.

3 Problem definition

The Pickup and Delivery Problem with Time Windows and Transfers (PDPTW-T) is defined as follows. We are given a set of transportation requests $R = \{(i^+, i^-) |\ i^+ \in P, i^- \in D\}$, where P and D are the set of pickup and delivery locations, respectively, and where a request $r_i = (i^+, i^-)$ concerns the pickup of a load at i^+ and its delivery at i^- within time window $[E_{i^+}, L_{i^-}]$, specifying the earliest time, E_{i^+}, when the load is available at the origin and the latest time, L_{i^-}, when the load must be delivered at the destination. A fleet of vehicles, V, is available to serve the requests. Each vehicle $v \in V$ starts and ends its shift at a depot located at $m_v \in M$, with M the set of all depot locations. Each vehicle $v \in V$ has a duty period, or shift, $[E_v, L_v]$; the vehicle can depart its depot at E_v and has to return to its depot by L_v. There is a set Γ of transfer locations. A request $r_i \in R$ can be served by one vehicle visiting both i^+ and i^-, or by two vehicles with one vehicle visiting i^+ and the other visiting i^-. In the latter case, the request is transferred at a location $t \in \Gamma$. Note that we allow only a single transfer. Two vehicles serving a request are not required to be at the transfer location at the same time, i.e., transfer locations offer short-term storage (a vehicle can drop off a request and leave, while another vehicle can come and pick up the request at a later time). Multiple vehicles can visit a transfer location at the same time and vehicles can wait at a transfer location. We focus on settings where the number of drivers is large and shift durations are short (relative to the planning horizon) – which is representative of crowdshipping settings. We also assume, for simplicity of exposition, that the size of loads is small compared to the vehicle capacity, and that the vehicle capacity is never restricting.

The PDPTW-T can be modeled on a directed graph $G(W, A)$ with node set $W = P \cup D \cup M \cup \Gamma$ and $A = \{(i, j) \mid i, j \in W, i \neq j\}$. It takes a vehicle $\tau_{ij} > 0$ units of time to travel from i to j, $(i, j) \in A$, and incurs a cost $c_{ij} > 0$. We assume travel times satisfy the triangle inequality. The PDPTW-T seeks to find a minimum cost set of vehicle routes serving all requests (either by one vehicle or by two vehicles).

3.1 Mathematical Formulation

[Rais et al., 2014] propose a flow-based mixed integer programming formulation for the PDPTW-T, in which request flows are linked with vehicle flows. The vehicle flow is modeled by binary variables x_{ij}^v, indicating the traversing of arc $(i, j) \in A$ by vehicle v. Request flows are modeled by binary variables y_{ij}^{vr} indicating whether vehicle $v \in V$ carries request $r \in R$ over arc $(i, j) \in A$. Synchronization of transfer operations is achieved through variables a_i^v and d_i^v, for arrival and departure times of vehicle v at location i, respectively, and binary variables s_{tr}^{vw} indicating whether request r is transferred between vehicles v and w at transfer location t. The formulation is given below

$$\min \sum_{v\in V,(i,j)\in A} c_{ij}x^v_{ij} \tag{1}$$

$$\text{s.t.} \sum_{(i,j)\in A} x^v_{ij} \leq 1 \qquad \forall v \in V, i = m_v \tag{2}$$

$$\sum_{(i,j)\in A} x^v_{ij} = \sum_{(j,i)\in A} x^v_{ji} \qquad \forall v \in V, i = m_v \tag{3}$$

$$\sum_{(i,j)\in A} x^v_{ij} - \sum_{(j,i)\in A} x^v_{ji} = 0 \qquad \forall v \in V, \forall i \in W \setminus M \tag{4}$$

$$\sum_{v\in V} \sum_{(i^+,j)\in A} y^{vr}_{i^+j} = 1 \qquad \forall r \in R, r = (i^+, i^-) \tag{5}$$

$$\sum_{v\in V} \sum_{(j,i^-)\in A} y^{vr}_{ji^-} = 1 \qquad \forall r \in R, r = (i^+, i^-) \tag{6}$$

$$\sum_{v\in V} \sum_{(t,j)\in A} y^{vr}_{tj} - \sum_{v\in V} \sum_{(j,t)\in A} y^{vr}_{jt} = 0 \qquad \forall r \in R, t \in \Gamma \tag{7}$$

$$\sum_{(i,j)\in A} y^{vr}_{ij} - \sum_{(j,i)\in A} y^{vr}_{ji} = 0 \qquad \forall v \in V, r = (i^+, i^-) \in R, i \in W \setminus \{\Gamma \cup \{i^+, i^-\}\} \tag{8}$$

$$y^{vr}_{ij} \leq x^v_{ij} \qquad \forall (i,j) \in A, v \in V, r \in \tag{9}$$

$$d^v_i + \tau_{ij} - a^v_j \leq M(1 - x^v_{ij}) \qquad \forall (i,j) \in A, v \in V \tag{10}$$

$$d^v_{i^+} \leq E_{i^+}, a^v_{i^-} \leq L_{i^-} \qquad \forall r = (i^+, i^-) \in R, v \in V \tag{11}$$

$$\sum_{(j,t)\in A} y^{vr}_{jt} + \sum_{(t,j)\in A} y^{wr}_{tj} \leq s^{vw}_{tr} + 1 \qquad \forall r \in R,\ t \in \Gamma,\ v, w \in V \tag{12}$$

$$a^v_t - d^w_t \leq M(1 - s^{vw}_{tr}) \qquad \forall r \in R,\ t \in \Gamma,\ v, w \in V \tag{13}$$

$$x^v_{ij} \in \{0, 1\} \qquad \forall (i,j) \in A,\ v \in V \tag{14}$$

$$y^{vr}_{ij} \in \{0, 1\} \qquad \forall (i,j) \in A,\ v \in V,\ r \in R \tag{15}$$

$$s^{vw}_{tr} \in \{0, 1\} \qquad \forall t \in \Gamma,\ r \in R,\ v, w \in V \tag{16}$$

$$a^v_i, d^v_i \geq 0 \qquad \forall i \in W,\ v \in V \tag{17}$$

The objective (1) is to find a minimum cost set of routes satisfying all requests. Constraints (2) enforce that each vehicle executes at most one route and constraints (3) enforce that a vehicle starts and ends a route at the same depot. Vehicle flow conservation is expressed by constraints (4). Constraints (5) and (6) guarantee that each request is served, i.e., it is picked up and delivered, respectively. The flow conservation for requests, at transfer locations, is enforced by constraints (7), and at non-transfer locations by constraints (8). Note that for transfer locations, a request entering the transfer is allowed to leave in a different vehicle, whereas for non-transfer locations a vehicle entering the location with a given request must leave the location with that request. Request and vehicle flows are linked by constraints (9), which enforce that a request has to travel in a vehicle. Consistency of arrival and departure times is enforced by big-M constraints (10) and time windows are enforced by constraints (11). Transfer operations are synchronized by constraints (12) and (13): if request $r \in R$ is transferred between vehicles v and w, $v, w \in V$, at location $t \in \Gamma$, then vehicle w can depart

from transfer location t only after the arrival of vehicle v at transfer location t.

As in the above formulation, we will assume for the remainder that a request is only transferred once, that a vehicle visits a given transfer location only once, and that two vehicles meet and exchange requests only once. A vehicle can visit multiple transfer locations on its route.

3.2 Benefits of Transfers

To illustrate and assess the potential benefits of using transfers, we analyze a set of stylized instances and compare solutions with and without transfers.

Consider an instance in which the pickup and delivery locations are vertices of a regular n-sided polygon with side length S, inscribed in a circle of radius Υ. A single depot ($m \in M$) is located at the center of the circle, and also acts as a transfer location ($t \in \Gamma$). Let the common shift for vehicles be $[0, 4T]$ and let the common time window for requests be $[0, 4T]$ as well.

Next, we locate the pickup and delivery locations of the requests in such a way that the ratio k_1/k_2, where k_1 is the minimum fleet size required to serve all requests when transfers are not allowed and k_2 is the minimum number of vehicles required when transfers are allowed, can get very large. Figures 1a, 1b, 1c, and 1d illustrate instances with 3, 4, 5, and 6 requests, respectively.

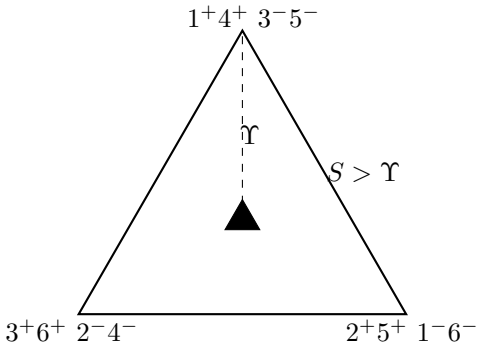
In Figure 1a, for example, six requests, $(1^+, 1^-), ..., (6^+, 6^-)$ are shown. If fewer than six vehicles are available to serve all requests when transfers are not allowed, then two requests must be picked up and delivered by the same vehicle. However, this is impossible, because two requests with a common pickup location do not have a common delivery location, which implies that the vehicle needs to visit three locations and the travel time is at least $T + S + S + T$ which exceeds $4T$ (since $S > T$). Observe that, if more requests are added, a fleet of six vehicles would still be enough to serve all requests, no matter where the pickup and delivery locations for the added requests are located. Thus, without transfers, $k_1 = 6$ vehicles are required.

When transfers are allowed, only $k_2 = 3$ vehicles are required. All loads are picked up and taken to the transfer location where they are consolidated and then taken to their final destination. It is not possible to use fewer that three vehicles. If two vehicle are used, then one of them has to visit two locations to pickup loads and take them to the transfer location, which would not leave enough time for delivery. Thus, for this setting, we have $k_1/k_2 = 2$.

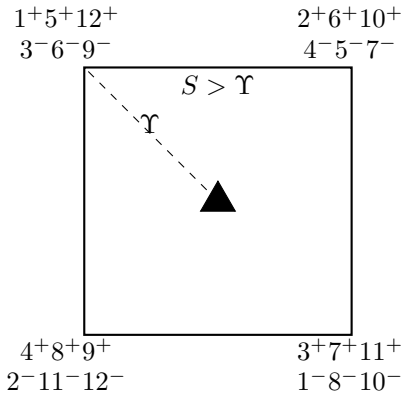
Figure 2, illustrates the potential benefits of transfers in a setting with two depots and a single transfer location. The common vehicle shift is such that it is not possible for a vehicle to serve two requests (and return to the depot in time). As a consequence, when transfers are not allowed, each request has to be served by a different vehicle (Figure 2a). When transfers are allowed, two vehicles, one from depot m_1 and one from depot m_2, can serve all requests (Figure 2b).

4 Adaptive Large Neighborhood Search

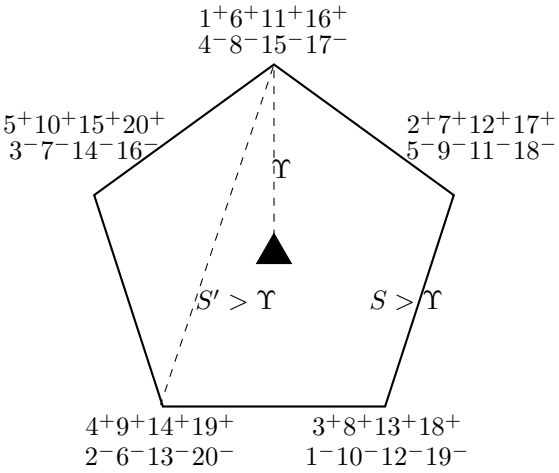
Solving instances of the mathematical model presented in Section 3 in a reasonable amount of computation time is only possible for small instances; [Rais et al., 2014] presents results for instances with 10 and 14 locations (5 and 7 requests, respectively) and where transfers are allowed at every location. This is due, in part, to the symmetry (vehicles are indistinguishable) and the use of big-M constraints, which results in weak linear relaxations. Therefore, we



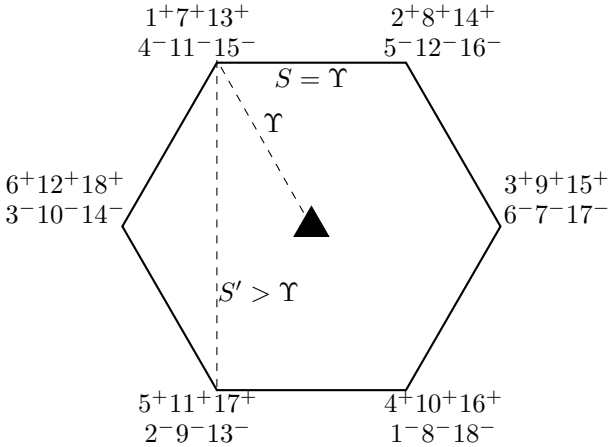
(a) $n = 3$. $k_1 = 6, k_2 = 3$, $k_1/k_2 = 2$.



(b) $n = 4$. $k_1 = 12, k_2 = 4$, $k_1/k_2 = 3$.

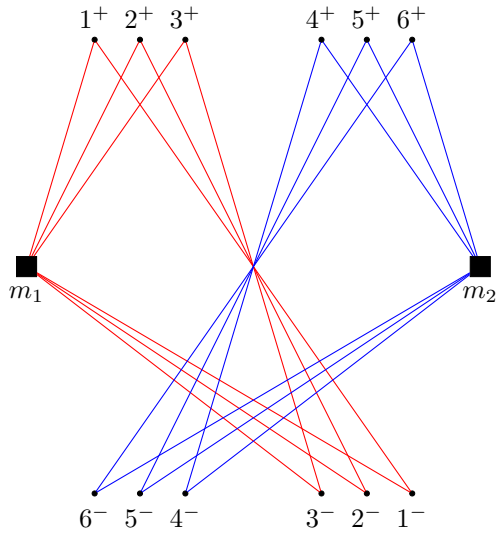


(c) $n = 5$. $k_1 = 20, k_2 = 5$, $k_1/k_2 = 4$.

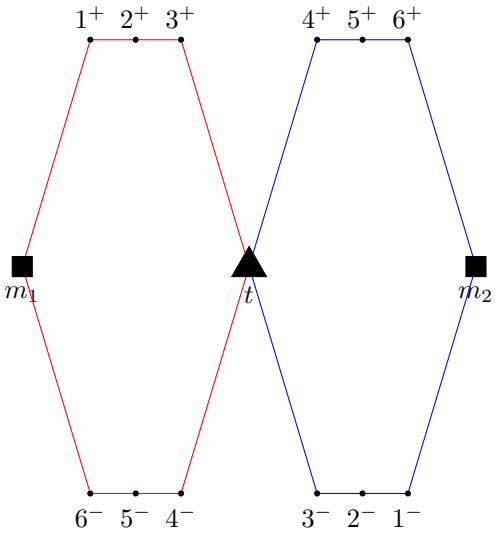


(d) $n = 6$. $k_1 = 18, k_2 = 6$, $k_1/k_2 = 3$.

Figure 1: PDPTW-T instances where transfers result in significant benefits.



(a)



(b)

Figure 2: A PDPTW-T instance where transfers result in significant benefits.

develop an Adaptive Large Neighborhood Search (ALNS) algorithm which is able to handle large instances in short computation times.

ALNS algorithms have been successfully applied to many hard combinatorial problems, in particular to many variants of the vehicle routing problem (see [Pisinger and Ropke, 2007]). An ALNS algorithm specifically designed for solve instances of the PDPTW is described in [Ropke and Pisinger, 2006]. A successful ALNS algorithm relies on the availability of several fast-to-explore local search neighborhoods for modifying a solution. We focus on neighborhoods that remove requests from the current solution and reinsert them to create a modified solution. At each iteration, one of the neighborhoods is selected, giving priority to neighborhoods that have been successful in earlier iterations. Improving solutions are always accepted, but a diversification mechanism is incorporated which sometimes accepts worse solutions. The basic flow of our ALNS algorithm is given in Algorithm 1.

Algorithm 1: Overview of the main steps in the proposed solution method (ALNS)

Input : Set $R = \{(i^+, i^-) | i^+ \in P, i^- \in D\}$ with pickup and delivery requests; Set of transfer locations, T

Output: Routing plan S^*

```
1  I ← Construct an initial solution having a subset of the requests transferred;
2  S*, S^c ← I;
3  while Termination criteria not satisfied do
4  |  Choose a removal (O^-) and an insertion (O^+) operators;
5  |  S ← O^+(O^-(S^c));
6  |  if z(S) ≤ z(S^c) then
7  |  |  S^c ← S;
8  |  |  if z(S) ≤ z(S*) then
9  |  |  |  S* ← S;
10 |  |  end
11 |  else
12 |  |  S^c ← accept(S, S^c);
13 |  end
14 |  Update weights used for operators selection;
15 end
16 return S*;
```

An initial solution, in which some of the requests are transferred, is obtained as described in Section 4.1. A removal and insertion operator are chosen (line 4) based on weights that reflect past performance (line 14). The removal and insertion operators are applied in sequence (line 5). The acceptance of the resulting solution is controlled by a scheme that is similar to those found in simulated annealing algorithms, i.e., an exponential function with a cooling rate α (line 12). The initial temperature (T_0) is given by $\frac{\omega}{-\ln(0.5)} z_0$, where z_0 is the cost of the initial solution and $0 < \omega < 1$, i.e., a solution with cost $(1 + \omega)z_0$ has probability 0.5 of being accepted. A higher value ω (and consequently T_0) allows for more diversification throughout the search, especially during the early stages, but may still be insufficient to avoid getting trapped in local optimum. Therefore, we take a similar approach to [Stenger et al., 2013] and reset the temperature to $\frac{\omega}{-\ln(0.5)} z^*$ after ν iterations in which no improving solution has been found, where z^* is the cost of the best solution found so far.

4.1 Initial Solution

A critical aspect of an ALNS algorithm for solving the PDPTW-T concerns the strategies embedded for deciding which requests to transfer and at which locations. Such strategies have to be embedded in the solution improvement framework (i.e., removal/insertion operators that explicitly consider transfers), but starting with an initial solution in which some requests are being transferred might help guide the search – requests are considered one at a time and introducing a (new) transfer is not likely to look attractive as it involves detours to visit a transfer location. Thus, by having (some) transfers in the initial solution, an ALNS algorithm is more likely to make effective use of transfer options.

Any solution, s, is characterized by a subset $R'_s \subset R$ of requests being transferred. We create an initial solution, s_0, by forcing a subset of requests to be transferred in the initial solution. Next, we describe how we obtain $R'_{s_0} \subset R$ (4.1.1) and how we decide on the transfer locations to use (4.1.2) when constructing s_0 (4.1.3).

4.1.1 Selecting requests to transfer

For a request $r_i = (i^+, i^-) \in R$, let τ_i be the (direct) travel time between i^+ and i^-. Selecting the requests that are forced to use a transfer in the initial solution is controlled by a threshold τ'. We let $\mathcal{R} = \{(i^+, i^-) \in R; \tau_i \geq \tau'\}$ be the candidate set for transferred requests in the initial solution.

Transferring of requests having relatively long travel times might yield cost savings as it may induce "regional" vehicle routes in which requests with pickup and delivery locations in a region are served without a transfer and requests with pickup and delivery locations in different regions are served using a transfer. The threshold τ' is defined based on the characteristics of an instance, e.g., half of the longest possible travel time.

4.1.2 Selecting a transfer location

Given the set of candidate requests to be transferred in the initial solution, the next decision concerns which transfer location to use. To prevent transfers that require a large detour (compared to traveling directly from pickup to delivery location), transferring $r_i \in \mathcal{R}$ via $t \in \Gamma$ is considered only when $\tau_{i^+,t} + \tau_{t,i^-} \leq \gamma\tau_i$, where $\gamma > 1$ is a parameter controlling the maximum allowed ratio between the detour $\tau_{i^+,t}+\tau_{t,i^-}$ and the direct travel time τ_i. Moreover, transferring request r_i via t has to be feasible for at least one combination of depots m_a and m_b, i.e., feasible routes $m_a - i^+ - t - m_a$ and $m_b - t - i^- - m_b$ that can be timed and synchronized to induce a feasible transfer at t have to exist. If there is no feasible transfer location for a request in the candidate set, the request is removed from the candidate set.

Among the feasible transfer locations $t \in \Gamma$ for request (i^+, i^-) we select the one that minimizes the difference between $\tau_{i^+,t}$ and τ_{t,i^-}. This strategy favors transfer locations that are located more centrally between the pickup and delivery locations of a request, thus better balancing the detour required to visit the transfer between the two vehicles serving the request. Figure 3 illustrates the possible transfer assignment for a request (i^+, i^-).

4.1.3 Constructing the Initial Solution

The initial solution, s_0, is constructed such that all requests in R'_{s_0} are served by two vehicles and transferred at the location selected using the strategy presented in Section 4.1.2. All

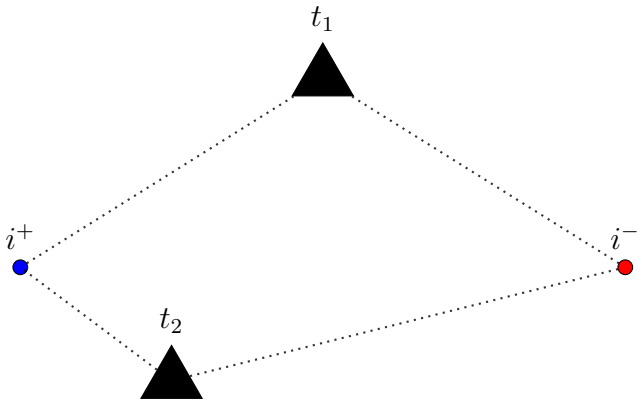


Figure 3: Selecting a transfer location for request (i^+, i^-). Location t_1 is selected, since it gives more balanced detours.

other requests $r_i \in R \setminus R'_{s_0}$ are served by a single vehicle. To construct a feasible solution, all requests in $R \setminus R'_{s_0}$ are put in the "request bank" and one iteration of the ALNS algorithm for the PDPTW by [Ropke and Pisinger, 2006] is executed to obtain a partial solution for those requests. Then, for each request $r_i \in R'_{s_0}$, we use a "Greedy insertion with transfer" operator of our ALNS framework (described in Section 4.2) to (try and) insert r_i using the selected transfer location. If unsuccessful, two new routes are created: $m_a - i^+ - t - m_a$ and $m_b - t - i^- - m_b$, where depots $m_a, m_b \in M$ are chosen such that the total travel time of the two new routes is minimized.

4.2 Improvement Phase

At the heart of any ALNS algorithm is a set of *operators*, each modifying the current solution in a specific and limited way, i.e., making small modifications to the current solution, and, thus, defining a neighborhood. As is common in many ALNS algorithms for routing problems, our operators either remove or insert sets of locations (either pickup or delivery locations).

For a given solution, let $\rho(i)$ and $\sigma(i)$ denote the direct (on the same route) predecessor and successor nodes, respectively, of node i. The detour cost associated with node i is $\delta_i = c_{\rho(i),i} + c_{i,\sigma(i)} - c_{\rho(i),\sigma(i)}$. The detour cost associated with request $r = (i^+, i^-)$ is $\delta_r = \delta_{i^+} + \delta_{i^-}$ when $\sigma(i^+) \neq i^-$, and $\delta_r = c_{\rho(i^+),i^+} + c_{i^+,i^-} + c_{i^-,\sigma(i^-)} - c_{\rho(i^+),\sigma(i^-)}$ when $\sigma(i^+) = i^-$. Similarly, if request r is not yet served in the solution, then the cost to insert its pickup i^+ (its delivery i^-) between nodes u and v ($v = \sigma(u)$) is the detour cost $c_{u,i^+} + c_{i^+,v} - c_{u,v}$ ($c_{u,i^-} + c_{i^-,v} - c_{u,v}$), which we also denote as δ_{i^+} (δ_{i^-}). The insertion cost of request $r = (i^+, i^-)$, given insertion positions (u^+, v^+) and (u^-, v^-) for the pickup and delivery nodes, respectively, is the detour cost $\delta_r = \delta_{i^+} + \delta_{i^-}$ when $(u^+, v^+) \neq (u^-, v^-)$, and $c_{u,i^+} + c_{i^+,i^-} + c_{i^-,v} - c_{u,v}$ when $(u^+, v^+) = (u^-, v^-) = (u, v)$.

4.2.1 Removal operators

A removal operator deletes a number of requests from the current solution and adds them to a request bank (from which insertion operators select requests to be inserted). Let $O_k^-(s)$ denote the (partial) solution that results after removal operator O_k^- is applied to solution s, and let $R_k^-(s)$ denote the set of requests deleted from solution s by operator O_k^-. Algorithm 2 illustrates the steps performed by a removal operator.

Algorithm 2: Steps of removal operator O_k^-

Input : Feasible solution, s
Output: Partial solution $O_k^-(s)$ and set of deleted requests $R_k^-(s)$

$q \leftarrow$ Number of requests to delete;
$O_k^-(s) \leftarrow s$;
$R_k^-(s) \leftarrow \emptyset$;
for $1, 2..., q$ **do**
  Select a request $r \in R \setminus R_k^-(s)$ to be deleted;
  $O_k^-(s) \leftarrow O_k^-(s) \setminus r$;
  $R_k^-(s) \leftarrow R_k^-(s) \cup r$;
end
return $O_k^-(s), R_k^-(s)$;

1. Worst request removal (O_1^-): The operator deletes q requests in non-decreasing order of their detour cost.

2. Random request removal (O_2^-): The operator deletes q requests selected at random.

3. Route removal (O_3^-): The operator uses a biased selection procedure to choose a route to delete from the current solution. Only routes that do not visit a transfer are considered. The probability that a route is selected to be deleted is proportional to the ratio of the waiting time in the route and the number of requests served in the route. Thus, the operator favors the deletion of less efficient routes. All requests in the chosen route are deleted. If the number of requests deleted is less than q, then another route is selected. The process continues until at least q requests are deleted.

4. Transfer-based request removal (O_4^-): This operator considers requests that have been transferred in one or more previous solutions. For each request $r_i = (i^+, i^-)$, let t_i^* denote the number of times request r_i has been transferred in an incumbent solution encountered during the search. The probability that a request r_i is selected is proportional to $1 - \frac{t_i^*}{I^*}$, where I^* denotes the (total) number of incumbent solutions encountered during the search. Note that requests that are seldom transferred in incumbent solutions are more likely to be deleted than requests that are often being transferred in incumbent solutions.

5. Cluster removal (O_5^-): This operator is an adaptation of an operator used in [Masson et al., 2013, Masson et al., 2014] and based on the idea that if the pickups (deliveries) of a set of requests form a cluster, it may be beneficial if they are picked-up (delivered) by the same vehicle and dropped off at (collected from) a transfer location. In our implementation, we randomly pick a "root" request (i^+, i^-) and compute sets C_i^+ and C_i^-, where C_i^+ contains pickup locations within radius μ from i^+ being serviced by a different vehicle (and the delivery location is not within μ from i^+) and C_i^- contains delivery locations within radius μ from i^- being serviced by a different vehicle (and the pickup location is not within μ from i^-). The requests in the larger set are deleted. The parameter q is ignored.

After the deletion of requests, it may happen that a vehicle visits a transfer location, but no transshipment of any request takes place. We allow such unnecessary visits to remain in the current solution for at most ϕ iterations. If in ϕ consecutive iterations a transfer location

t is not used to transfer any request, it is removed from the route(s). Allowing unnecessary visits to transfer locations can be beneficial, because inserting a request involving a transfer location that is not yet in the solution results in additional detour costs and, consequently, a deterioration in solution quality. If the transfer location is already in the solution, even if it is currently not used to transfer requests, the increase is only due to the detour cost incurred for the inserted request itself. Moreover, it also increases the likelihood that the transfer location is already visited by multiple vehicles. A downside of keeping transfer locations with unnecessary visits in the solution is that the time required to visit the transfer location may render an otherwise feasible insertion infeasible. Moreover, more insertions with transfers are evaluated and, as a consequence, execution time is longer. Despite these downsides, our experiments have shown that keeping transfer locations with empty visits provides a good mechanism for exploring transfer opportunities. A post-processing step at the end of the search removes any unnecessary visits at transfer locations in the best solution found.

4.2.2 Insertion operators

The objective of an insertion operator is to reintroduce requests in the set $R_i^-(s)$, i.e., the requests deleted by removal operator O_i^-. Let $O_j^+(s)$ denote the solution obtained after applying insertion operator O_j^+ to a (partial) solution s. Algorithm 3 presents the steps performed by an insertion operator that does not consider transfers, and Algorithm 4 presents the steps performed by an insertion operator that does consider transfers.

1. Greedy insertion without transfer. Requests are sequentially inserted in the least-cost position and route. If a request cannot feasibly be inserted, it remains in the request bank. The sequence in which requests are inserted defines an operator:

   - O_1^+: Requests are selected in decreasing order of direct travel time from pickup to delivery location.
   - O_2^+: Requests are selected in increasing order of insertion cost. More formally, if Δ_{rk} represents the least cost to insert request r in route k ($\Delta_{rk} = \infty$ if the insertion is infeasible), then the operator selects request r' to be inserted in route k' as $(r', k') = \text{argmin}_{r \in R_i^-(s),\, k \in K} \Delta_{rk}$. Values Δ_{rk} are updated after the selected request has been inserted in its associated route.
   - O_3^+: Requests are selected in decreasing order of regret, i.e., the absolute difference between least insertion cost and the second least insertion cost. More formally, if Δ_{rk}^1 is the least-cost insertion of request r in a route k and Δ_{rk}^2 is the second least-cost insertion of request r in a route k (where k may be the same or a different route), then the operator selects request r' to be inserted in route k' as $(r', k') = \text{argmax}_{r \in R_i^-(s),\, k \in K}(\Delta_{rk}^2 - \Delta_{rk}^1)$. Values Δ_{rk}^1 and Δ_{rk}^2 are updated after the selected request has been inserted in its associated route.
   - O_4^+: Similar to operator O_2^+, but the request is selected randomly among the best ψ insertions, where, after initial experiments, ψ is set to 30% of the feasible insertions.

2. Greedy insertion with transfer. Requests are sequentially inserted in the least-cost positions and routes. If a request cannot feasibly be inserted, it remains in the request bank. Given a request $r = (i^+, i^-)$, an operator seeks two routes, k_a and k_b, and a transfer location, t, such that the possible visit times at t for k_a and k_b overlap (i.e., there exist a

Algorithm 3: Steps of insertion operator that does not consider transfers

Input : Partial solution s and set of requests R to be inserted

Output: A complete solution $\bar{s}$, or a partial solution $\bar{s}$ plus a set $\bar{R}$ of requests that were not inserted

```
1  L ← requests in R ordered accordingly to some criterion;
2  R̄ ← ∅;
3  s̄ ← s;
4  I ← ∅ // list of feasible insertions;
5  for r = (i⁺, i⁻) ∈ L do
6  |  feasible ← False;
7  |  for each route v in s do
8  |  |  for k₁ ∈ {0, 1, ..., K}, positions within route v do
9  |  |  |  evaluate insertion of i⁺ at position k₁ in v;
10 |  |  |  if insertion is feasible then
11 |  |  |  |  for k₂ ∈ {k₁, ..., K} do
12 |  |  |  |  |  evaluate insertion of i⁻ at position k₂ in v;
13 |  |  |  |  |  if insertion is feasible then
14 |  |  |  |  |  |  feasible ← True;
15 |  |  |  |  |  |  I ← I ∪ (k₁, k₂, v);
16 |  |  |  |  |  end
17 |  |  |  |  end
18 |  |  |  end
19 |  |  end
20 |  end
21 |  if feasible=False then
22 |  |  R̄ ← R̄ ∪ r
23 |  end
24 |  insertion r in s̄ using least-cost insertion i ∈ I;
25 end
26 return s̄, R̄;
```

Algorithm 4: Steps of insertion operator that considers transfers.

Input : Partial solution s and set of requests R to be inserted
Output: A complete solution $\bar{s}$, or a partial solution $\bar{s}$ plus a set $\bar{R}$ with requests that were not inserted

```
L ← requests in R ordered accordingly to some criterion;
R̄ ← ∅;
s̄ ← s;
I ← ∅ // list of feasible insertions;
for r = (i⁺, i⁻) ∈ L do
    feasible ← False;
    for each route v and transfer location t visited by v in s do
        n ← position of t in v;
        for k₁ ∈ {0, 1, ..., n − 1}, positions in v do
            evaluate insertion of i⁺ at position k₁ in v;
            if insertion is feasible then
                for each route w visiting t do
                    m ← position of t inh w;
                    for k₂ ∈ {m + 1, ..., K} positions in route w do
                        evaluate insertion of i⁻ at position k₂ in w;
                        if insertion is feasible then
                            feasible ← True;
                            I ← I ∪ (k₁, k₂, t, v, w)
                        end
                    end
                end
            end
        end
    end
    if feasible=False then
        R̄ ← R̄ ∪ r
    end
    insert r in s̄ using least-cost insertion i ∈ I;
end
return s̄, R̄;
```

time when both vehicles can be at t). If it is possible to insert i^+ before t in k_a and i^- after t in k_b (properly accounting for any change in the arrival time at t of route k_a due to the insertion of i^+), the insertion is considered feasible. The sequence in which requests are inserted defines an operator:

- O_5^+: Requests are selected in decreasing order of direct travel time from pickup to delivery location.
- O_6^+: Requests are selected in increasing order of insertion cost. More formally, if $\Delta_{rk_1k_2}$ is the cheapest cost to insert request $r = (i^+, i^-)$, where i^+ and i^- are inserted in routes k_1 and k_2, respectively ($\Delta_{rk_1k_2} = \infty$ if the insertion is infeasible), then the operator selects a request r' and routes k_1' and k_2' by $(r', k_1', k_2') = \text{argmin}_{r \in R_i^-(s),\, k_1,k_2 \in K} \Delta_{rk_1k_2}$. Values $\Delta_{rk_1k_2}$ are updated after the selected request has been inserted in its associated routes.
- O_7^+: Requests are selected in decreasing order of regret. More formally, if $\Delta^1_{rk_1k_2}$ represents the least-cost insertion of request r using a transfer, and $\Delta^2_{rk_1k_2}$ represents the least-cost insertion of request r using a transfer (where the routes in which the pickup and delivery location are inserted may be the same or may differ), then the operator selects request r' and its associated routes as $\text{argmin}_{r \in R_i^-(s),\, k_1,k_2 \in K}(\Delta^2_{rk_1k_2} - \Delta^1_{rk_1k_2})$. Values $\Delta^1_{rk_1k_2}$ and $\Delta^2_{rk_1k_2}$ are updated after the selected request has been inserted in its associated routes.

3. Transfer insert O_8^+: This operator facilitates the inclusion of transfers in a solution by creating routes with an unnecessary visit at a transfer location. Requests are selected in decreasing order of direct travel time between pickup and delivery location. For a request $r = (i^+, i^-)$, let C_i be the circle with radius $\frac{\tau_i}{2}$, where τ_i is the distance between i^+ and i^-, centered at the midpoint of the line segment joining i^+ and i^-. If there is a transfer location t within C_i and a feasible route $m_v - i^+ - t - i^- - m_v'$ for vehicle $v \in V$ (not yet in the solution), this route is created. If there are multiple transfer locations in C_i, the one minimizing the difference $|\tau_{i^+,t} - \tau_{t,i^-}|$ is chosen.

It is possible that an insertion operator does not succeed to insert all requests deleted by a removal operator. In this case, the remaining requests are inserted as follows: find a route k_a with transfer location t_a maximizing the number of of delivery locations that can be feasibly inserted in k_a after t_a. Let the set of these delivery locations be denoted by D. Similarly, find a route k_b with transfer location t_b maximizing the the number of pickup locations that can be feasibly inserted in k_b before t_b. Let the set of these pickup locations be denoted by P. If $|D| \geq |P|$, the delivery locations in D are inserted in k_a (if feasible) after t_a. For each inserted delivery location, if it is not possible to insert the corresponding pickup in a route already in the solution, a new route $m_v - i^+ - t_a - m_v$ is created (assuming a vehicle v is available). If $|D| < |P|$, a similar process is used, but starting with the pickup locations. If, at the end, there are still remaining requests, these are inserted in new routes without transfers.

4.2.3 Operator selection

The adaptive weight mechanism used to control the selection of a removal and insertion operator is implemented by a roulette wheel procedure, as described in [Pisinger and Ropke, 2007]. If the weight of operator O_i, $i \in 1, ...\Pi$, is π_i, expressing the desirability of selecting operator

O_i, then the probability of selecting O_i is given by $\pi_i / \sum_{j=1}^{\Pi} \pi_j$. The removal and insertion operators are chosen independently, i.e., with different roulette wheels. At the start of the search, all operators have equal weights. During the search, the weights are updated every κ iterations, similarly to [Pisinger and Ropke, 2007]. As discussed above, in our ALNS algorithm, two types of insertion operators are considered: operators in which transfers are considered, and operators in which transfer are not considered. Since the insertion of a node (request) can be feasible for one type, but not for the other, our ALNS algorithm selects two insertion operators, one of each type. The operator with the largest weight is applied first, and if there are remaining nodes (requests), then the other operator is applied.

The performance of each operator is recorded in its weight based on the solutions obtained after the operator is applied. In particular, the score $\bar{\pi}_i$ for operator O_i starts at 0 at the start of each segment of κ iterations, and is incremented by: σ_1 if an overall best solution is found, σ_2 if an improving solution is found, and σ_3 if a worse solution is found and accepted (as part of the annealing scheme). If a_i is the number of times operator O_i was used during the current segment (of κ iterations), then the new weight π_i is given by $\pi_i(1-\rho)+(\bar{\pi}_i/a_i)\rho$, where $\rho \in [0,1]$ adjusts the importance given to scores in past segments. Since two insertion operators may be applied in one iteration, their weights are updated independently.

4.2.4 Efficiency

The performance of an insertion-based neighborhood search heuristic relies heavily on its ability to efficiently evaluate the feasibility of an insertion, as a very large number of insertions will be attempted. In Appendix A, we discuss the information that is kept with the current solution to allow fast (constant time) evaluation of the feasibility of an insertion. Moreover, we show how to update this information when the current solution changes.

5 Computational Experiments

In this section, we present the results of a set of computational experiments conducted to evaluate the performance of the ALNS algorithm and to characterize transportation settings which might benefit from the introduction of transfers. The ALNS algorithm was coded in C++ and the experiments were conducted using an Intel Core i5-2450M machine, running at 2.5 GHz x 4 with 4 GB of RAM, under Ubuntu 16.04. To investigate the benefits of transfers, we compare the solutions obtained by our ALNS algorithm to the solutions obtained by the ALNS algorithm of [Ropke and Pisinger, 2006], which does not consider transfers. Furthermore, we analyze the impact of different problem characteristics, in particular, the vehicle's shift length, the number of transfer locations, the geographic distribution of transfer locations, and the geographic distribution of customers.

5.1 ALNS Algorithm Parameter Tuning

We have carefully calibrate the values of the parameters which we believe have a significant impact on the performance of our proposed ALNS algorithm. The other parameters are set to the values used in [Ropke and Pisinger, 2006], which can be found in Table 1.

Parameter tuning was done using a set of six instances, three with 30 and three with 50 requests. All parameters to be tuned are given a default value. Then, a single parameter

Table 1: **ALNS Parameters and values**

Parameter	Description	Value
ι	ALNS iterations	25000
α	SA cooling rate	0.98
ω	Initial temperature adjustment	0.15
κ	Level length	250
ν	Max. non-improving iterations	250
ρ	Reaction factor	0.2

value (or single parameter set values) is varied while the others are kept fixed. For each parameter value considered (of the parameter being tuned), we solve each of the six instances five times. The parameter value yielding the best results is chosen. The process continues until all parameters have been assigned a value. The parameters to be tuned, and their respective default values, are $((\psi_1, \psi_2), (\sigma_1, \sigma_2, \sigma_3), \phi) = ((6, 10), (10, 10, 10), 100)$. In the initial solution for an instance, each request is served individually by a single vehicle without a transfer. To assess the quality of different parameter values, we assign a score to each solution obtained: z^*/z, where z is the cost of the solution and z^* is the cost of best solution encountered during an experiment. Because we also want the proposed ALNS algorithm to be reasonably efficient, we adjust the score to take run times into account as well: $0.7z^*/z + 0.3t^*/t$, where t is the run time and t^* is the fastest run time encountered during an experiment. Thus, a value close to 1.0 for a parameter value (the average score over the five runs) indicates superior performance.

Table 2: **Calibration of ALNS parameters.**

(ψ_1, ψ_2)			$(\sigma_1, \sigma_2, \sigma_3)$			ϕ	
(0.1,0.2)	(0.2,0.3)	(0.3,0.4)	**(40,10,1)**	(4,2,1)	(1,1,1)	0	**100**
0.98	0.89	0.85	*0.96*	0.95	0.93	0.88	*0.96*
0.87	0.82	0.76	*0.88*	0.86	0.85	*0.90*	0.88
0.87	0.79	0.73	*0.95*	0.92	0.92	0.88	*0.95*
0.95	0.88	0.83	0.96	0.97	*0.98*	*0.97*	0.96
0.96	0.87	0.83	*0.95*	0.91	0.88	0.93	*0.95*
0.93	0.84	0.77	*0.90*	0.74	0.78	0.72	*0.90*

[1] Each row corresponds to an instance and each column to the average score over five runs for the relevant parameter value(s). The best parameter value(s) is shown in bold.

First, we tune the interval from which the number of requests to be removed by a removal operator is drawn uniform randomly, i.e., $[\psi_1|R|, \psi_2|R|]$, where parameters $0 < \psi_1 < \psi_2 < 1$ control the fraction of requests to be removed. The results show that $(\psi_1, \psi_2) = (0.1, 0.2)$ performs best. Allowing for a larger fraction of requests to be removed may lead to a wider exploration of the solution space, but it also increases time per remove-and-reinsert combination as requests are inserted one by one and execution time primarily depends on the number of requests inserted.

Next, we tune the reward values for the operators, where we note that their relative difference is more important than the values themselves. We consider three sets of parameters: one in which finding new best and better solutions is highly rewarded, i.e., $(\sigma_1, \sigma_2, \sigma_3) = (40, 10, 1)$, one in which success in finding new best and better solutions is recognized as a positive

and rewarded, but not as much, i.e., $(\sigma_1, \sigma_2, \sigma_3) = (4, 2, 1)$ and, finally, one in which we do not actively "encourage" the search for new best and better solutions, but rely on the randomness in the search process to encounter high-quality new best and better solutions, i.e., $(\sigma_1, \sigma_2, \sigma_3) = (1, 1, 1)$. With the exception of one instance, the best results are obtained by highly rewarding success in finding new best and better solutions. This indicates that the SA acceptance mechanism, in combination with the restarting procedure, is enough to provide a satisfactory level of diversification in the search, and avoids getting trapped in local optima.

Finally, we calibrate the maximum number of iterations an unnecessary transfer location visit is allowed to remain in the current solution, ϕ. We consider two scenarios: one in which an unnecessary transfer location visit is removed immediately, i.e., $\phi = 0$, and one in which an unnecessary transfer location visit is kept much longer, i.e., $\phi = 100$. Even though run times increase when an unnecessary transfer location visit is kept longer, it appears to be worthwhile to do so. This is likely due to the fact that it significantly simplifies the search for beneficial transfer opportunities.

5.2 Stylized Instances

We have used the proposed ALNS algorithm to solve stylized instances as shown in Figure 1, with 4, 5, 6, 7 and 8 sided polygons inscribed in a circle with radius 100 (where instances for 7 and 8 sided polygons are created as expected). We compare the solutions obtained by the proposed ALNS algorithm, where the center of the circle can be used as transfer location, to the solutions obtained by the ALNS algorithm of [Ropke and Pisinger, 2006] (i.e., without any transfers), and to the optimal solutions. The results can be found in Table 3. For each instance, we report the total travel distance (Dist), the number of vehicles used (Veh), the total computational time in seconds (Time) and the number of requests transferred (Trans).

Our proposed ALNS is able to obtain the optimal solution for all but one of the instances. The higher run times of our proposed ALNS algorithm compared to the one of [Ropke and Pisinger, 2006] are the result of more complex and more time consuming insertion operators, because the need to consider transfers. This is especially true for these stylized, highly symmetrical instances, where proper synchronization of transfers is critical.

Table 3: Algorithm comparison using stylized instances.

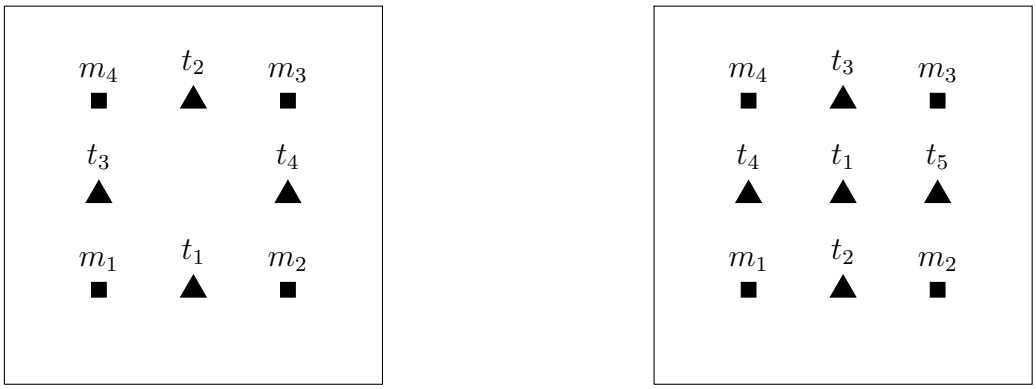
Instance		ALNS Ropke			Proposed ALNS				Optimal	
Polygon	$\|R\|$	Dist	Veh	Time	Dist	Veh	Trans	Time	Dist	Veh
4-sided	12	4336	12	0.9	1600	4	10	11.3	1600	4
5-sided	20	7104	20	1.5	2000	5	15	25.5	2000	5
6-sided	30	7244	18	2.2	2400	6	24	48.3	2400	6
7-sided	42	10652	28	13.6	2992	9	36	32.6	2800	7
8-sided	56	14988	40	17.8	3200	8	49	94.6	3200	8

5.3 Randomly Generated Instances

To assess the potential benefits of transfers in freight transportation systems employing crowdsourced drivers, we generate sets of instances that capture characteristics of such systems. We consider a squared geographical area of 120×120 units of distance and assume that one unit of

distance can be traveled in one time unit (e.g. 60 km/h). Four depots are located in the area as shown in Figure 4a, requests can be served from any depot, and the number of vehicles in each depot is unlimited. Pickup and delivery locations are drawn uniform randomly in the area, but we consider different scenarios for origin-destination proximity: long-distance requests only (L), i.e., at least 60 units between a pickup and a delivery location, short-distance requests only (S), i.e., no more than 60 units, but at least 30 units between a pickup and a delivery location, and a third scenario having both long-distance and short-distance requests (M).

We generate instances with 50, 75, and 100 requests, each with a 3-hour time window, i.e., $E_{i^+} = 0$ and $L_{i^-} = 180$ for all $r_i \in R$, for each class L, M and S. If $\gamma = {}^1/_{|R|} \sum_{r_i \in R} \tau_i$ indicates the origin-destination proximity of an instance, then for the generated instances in classes L, M, and S, we have, on average, $\gamma = 74.67$, $\gamma = 57.69$ and $\gamma = 45.82$, respectively. We consider vehicle shift lengths 180, 240, and 300 (i.e., drivers operate for three, four, or five hours). The instances are created such that even for the shortest shift length ($H = 180$), all requests can be served (i.e., there is at least one depot such that a vehicle from that depot can feasibly visit the pickup and delivery locations of a request on a route). Moreover, we consider two geographies for the transfer locations, MD-4T and MD-5T, shown in Figures 4a and 4b, respectively. In the former there are four transfer locations, and in the latter there are five transfer locations, with one additional transfer location in the center of the region. The full set of instances can be downloaded at http://dx.doi.org/10.17632/pywzcgyzrv.1.



(a) MD-4T (b) MD-5T

Figure 4: Basic settings for multi-depot instances generation. Squares and triangles represent depots and transfers, respectively.

5.4 Transfer benefits

We first illustrate the potential benefits of transferring requests by analyzing the results of a single instance of class L with 25 requests, shift length 180, and transfer location geography MD-4T. Figure 5 shows two solutions, one in which transfers are not considered (Figure 5a) obtained with the ALNS by [Ropke and Pisinger, 2006], and one in which transfers are considered (Figure 5c) obtained with our proposed ALNS (squares indicate depots, triangles indicate transfer locations, and circles indicate pickup and delivery locations).

In the solution without transfer (Figure 5a) all but six requests are served by a dedicated vehicle (i.e., a vehicle serving a single request), and 22 vehicles are used. In the solution with transfers (Figure 5c) only two requests are served by a dedicated vehicle and all other requests

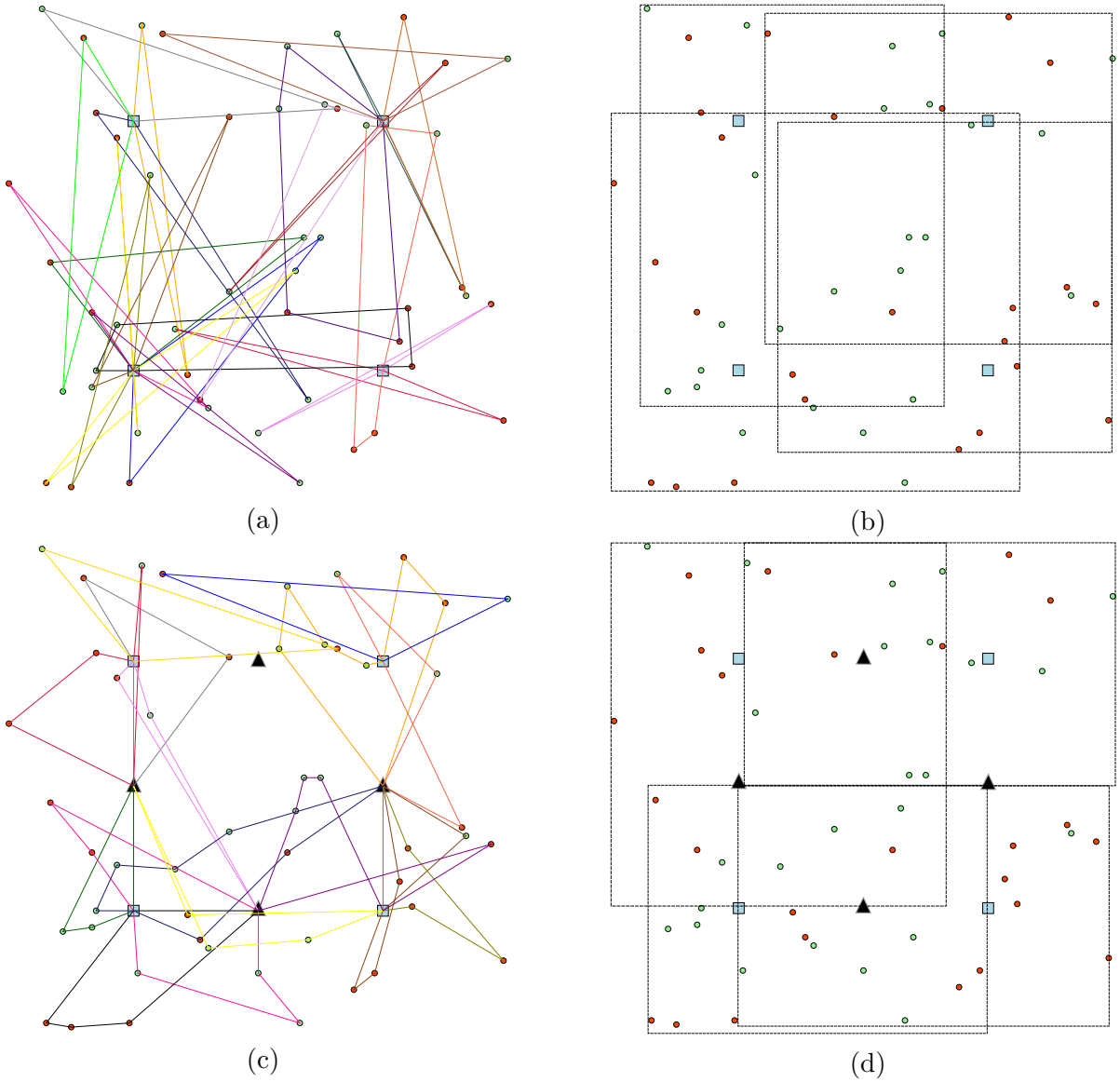


Figure 5: Overview of a solution without transferred requests (a), and a solution having requests transferred (c).

are transferred, and only 10 vehicles are used. To facilitate comparing the solutions with and without transfers and to better understand the benefits of transfers, we show, in Figures 5b and 5d, four rectangles, one for each depot, representing the area containing the routes originating from that depot. Observe that in Figure 5b, corresponding to the solution without transfers, there is a much larger overlap between the rectangles than in Figure 5d, corresponding to the solution with transfers. This illustrates that when transfers are possible, vehicles can stay closer to the depots and can perform pickups and deliveries of multiple requests within the shift; it is as if the transfer locations induce sub-regions within the region. When transfers are not possible, a vehicle performing a pickup has to also perform the corresponding delivery and consolidating multiple requests withing the shift becomes more difficult.

Similar settings i.e.m transfer location geography MD-4T, 3-hour time window, shift length 180, are used to generated a set of 15 instances with 10 requests, where five instances have

long-only, five instances have short-only and five instances have mixed-distance requests. These instances are solved using the IP formulation presented in Section 3 and employing Gurobi 8.01 as the IP solver with a time limit of two hours. For each instance, we compute a solution without transfers, using the ALNS algorithm by [Ropke and Pisinger, 2006], and provide it as an initial solution to the solver. We also compute a solution using our proposed ALNS algorithm and provide it as an initial solution. The results can be found in Table 4, where we report the total distance (*Dist*), the number of vehicles used (*Veh*), and the optimality gap after two hours (*Gap*). Solutions in which one or more requests are transferred are labeled with a superscript t.

Table 4: Experiments for instances with 10 requests.

	without transfers		Gurobi			with transfers		Gurobi		
Inst	Dist	Veh	Dist	Veh	Gap(%)	Dist	Veh	Dist	Veh	Gap(%)
1	1566	10	1566	10	68.3	1319^t	9	1289^t	8	61.4
2	1358	9	1358	9	74.5	863^t	6	863^t	6	58.8
3	1590	10	1590	10	77.5	1117^t	7	1117^t	7	65.4
4	1353	9	1187^t	8	67.2	1022^t	7	1022^t	7	64.0
5	1536	10	1536	10	76.7	1130^t	8	1033^t	6	64.5
6	645	5	645	5	32.7	645	5	645	5	32.7
7	734	5	734	5	46.4	718^t	5	718^t	5	47.2
8	613	4	613	4	37.2	613	4	613	4	37.2
9	840	6	816^t	6	38.1	801^t	6	801^t	6	39.5
10	803	5	747^t	5	42.7	793^t	5	747^t	5	46.0
11	1190	8	1190	8	71.7	658^t	4	619^t	4	37.3
12	1165	8	1100^t	7	70.8	985^t	7	957^t	6	61.1
13	1077	8	1077	8	67.5	845^t	6	845^t	6	57.8
14	903	6	903	6	52.5	725^t	5	725^t	5	40.8
15	935	6	911	6	50.3	868^t	6	868^t	6	60.0

[t] Solution has transferred requests.

The results show that for most instances, Gurobi is unable to improve the initial solution provided in two hours (note the large optimality gaps after two hours of computing). If a solution without transfers is provided, a better solution is found for five instances, where in four of them the improved solution has at least one request that is transferred. The solutions produced by our ALNS algorithm are equal or better (in terms of total distance and number of vehicles used) in 14 out of the 15 instances. When the solutions produced by our ALNS algorithm are provided as initial solution, Gurobi again finds a better solution for five instances, where in three of them the improved solution uses fewer vehicles.

5.5 Computational Study

A set of 10 instances was generated for each combination of number of requests, origin-destination proximity, and vehicle shift length. Each instance in a set is solved five times using the ALNS algorithm by [Ropke and Pisinger, 2006] and our proposed ALNS algorithm, and the average cost of the solutions as well as the lowest cost among them are reported. Two

initial solutions were considered in the runs with our proposed ALNS algorithm: the solution obtained with the ALNS algorithm by [Ropke and Pisinger, 2006] and the initial solution constructed as described in Section 4.1 using $\tau' = 0.8 \max_{r_i \in R}\{\tau_i\}$ and $\gamma = 1.25$. To evaluate the potential benefits of transfers, we compare the gap between solutions with transfers and solutions without transfers: $\Delta = (c^w - c)/c$, where c is the cost of the solution without transfers and c^w is the cost of the solution with transfers. Note that, because the solution found by the ALNS algorithm of [Ropke and Pisinger, 2006] is not necessarily optimal, we cannot claim that the benefits are solely due to adding transfers.

First, we consider transfer location geometry MD-4T. In Table 5 we report the results for instances with 50 requests for each origin-destination proximity class ($\mathcal{C}$) and vehicle shift lengths 180, 240 and 300 (H). Each row in the table presents average results over all instances for the given combination $(H, \mathcal{C})$. Column *RP ALNS* reports the cost (c) and the fleet size (Veh) of the solutions obtained with the ALNS algorithm by [Ropke and Pisinger, 2006] (in each of the five runs, the solution was found). Column *Proposed ALNS* reports the results obtained using our proposed ALNS algorithm. Column *Best* reports results for the best among the five solutions, whereas column *Average* reports the average over the five solutions. Column s_0 indicates the initial solution used: s_r, solution produced by the ALNS algorithm of [Ropke and Pisinger, 2006], or s_t, solution produced by the scheme described in Section 4.1. Column $\Delta_D(\%)$ reports the reduction in distance, column $\Delta_V(\%)$ reports the reduction in the number of vehicles used, column *Trans* reports the number of transferred requests, and column *CPU(s)* reports the run time in seconds, excluding the time to obtain the initial solution. Note that scheme s_t requires little time (less than one second) whereas scheme s_r requires running the ALNS algorithm by [Ropke and Pisinger, 2006] which takes a considerable amount of time (on average 60% of the running time of our proposed ALNS algorithm). Detailed results can be found in Appendix B.

We observe that transfers can provide significant benefits, especially when the driver shift length is short ($H = 180$) and the distance between pickup and delivery locations is long (L) – we see a reduction of almost 50% for both the total distance and the number of vehicles used. When the distance between pickup and delivery location is short (S), the benefits are minor, in the order of 1-2%, regardless of driver shift length. As expected, the benefits decrease when driver shift lengths increase, because with longer driver shift lengths routes can cover larger distances and serve more requests in the same route, which tends to be less costly than using transfers.

The effect of the initial solution appears to be minor. Starting from an initial solution in which requests are transferred tends to result in final solutions with slightly more requests being transferred than starting from an initial solution in which no requests are transferred. This is likely due to the fact that insertions with transfer operators are more likely to be rewarded during the early stages of the search when the initial solution already has some transfers. When starting from a high-quality (locally optimal) solution without transfers, introducing transfers in the solution is likely to increase the total distance and insertions with transfer operators are less likely to be rewarded. For instances in which the distance between pickup and delivery location is short (S), and where transfers have a limited value, especially for larger shift lengths ($H = 240$ and $H = 300$), starting from an initial solution with transfer is in fact detrimental to the quality of the final solution. We also note that the run time is slightly higher when starting from an initial solution with transfers, because, as noted earlier, the more time-consuming insert with transfer operations are performed more often.

Inst.	RP ALNS		Proposed ALNS							
			Best				Average			
$H, \mathcal{C}$	c	Veh	s_0	$\Delta_D(\%)$	$\Delta_V(\%)$	Trans	CPU(s)	$\Delta_D(\%)$	$\Delta_V(\%)$	Trans	CPU(s)
$180, L$	6322.1	39.0	s_r	-47.7	-46.2	47.2	58.5	-45.8	-44.1	45.5	59.5
			s_t	-46.7	-45.1	45.7	58.7	-45.2	-43.4	44.8	61.5
$180, M$	4116.3	24.8	s_r	-30.7	-27.4	40.1	50.9	-27.4	-23.8	32.9	50.0
			s_t	-30.9	-27.8	39.6	55.0	-28.3	-24.8	36.8	53.8
$180, S$	2459.2	15.0	s_r	-2.3	-2.7	8.6	35.8	-1.1	-0.4	7.2	38.0
			s_t	-4.9	-1.3	20.4	44.1	-2.6	0.0	16.0	42.1
$240, L$	3864.7	18.8	s_r	-25.6	-13.3	47.7	54.0	-19.4	-10.6	36.2	51.1
			s_t	-26.2	-16.0	47.2	55.1	-22.5	-11.6	43.3	54.1
$240, M$	3012.0	15.1	s_r	-11.3	-5.3	26.3	50.3	-6.5	-2.4	15.4	44.3
			s_t	-15.6	-5.3	41.1	54.2	-9.3	-1.3	27.3	49.7
$240, S$	2195.1	11.8	s_r	-1.8	0.0	12.2	45.6	-0.6	0.0	4.1	41.9
			s_t	-1.5	2.5	22.1	55.3	1.4	5.1	19.6	49.2
$300, L$	3653.1	16.6	s_r	-20.4	-3.0	46.5	55.2	-13.5	-2.2	30.8	50.8
			s_t	-21.7	-3.6	47.3	49.8	-17.6	-1.9	42.1	53.4
$300, M$	2952.8	14.4	s_r	-8.6	0.7	29.2	53.5	-4.1	-0.6	12.5	45.7
			s_t	-10.7	-0.7	33.4	45.8	-6.8	1.5	24.8	48.5
$300, S$	2188.9	11.7	s_r	-0.8	-1.7	5.8	43.6	-0.3	-0.7	2.4	41.2
			s_t	-1.0	0.9	12.9	42.4	1.4	4.3	15.3	47.3

Table 5: Results for generated instances with 50 requests with 3h time windows in the MD-4T setting

Next, we consider transfer location geometry MD-5T. The results can be found in Table 6. The benefits are similar to what we have seen for transfer location geometry MD-4T, but for the fact that, on average, more requests are transferred, and, consequently, slightly higher CPU times are observed.

To show that our proposed ALNS algorithm scales well, we present results for instances with 75 and 100 requests in Tables 7 and 8, respectively. In each table, we show the results obtained for transfer location geometry MD-5T and using an initial solution with transfers. We observe once more that for instances in which there are requests with a long distance between pickup and delivery location (L and M) and the shift lengths are relatively short (180 and 240), the proposed ALNS algorithm is able to find solutions that reduce both the total distance and the number of vehicles used.

5.6 Evaluation of remove and insert operators

The performance of the remove and insert operators employed in our ALNS algorithm has been evaluated on two representative instances. The first instance, I_1, has 50 long-only requests, shift length 180, and transfer location geometry MD-4T. As we have seen, such an instance is likely to benefit from transfers. The second instance, I_2, has 50 short-only requests, shift length 300, and transfer location geometry MD-4T – an instance less likely to benefit from transfers. We analyze which operators are used throughout the search when solving these two instances and also if this depends (strongly) on the initial solution.

Recall that the likelihood of an operator being selected is proportional to its weight, where the weight is initialized and updated during each segment of κ iterations (within a segment,

Inst.	RP ALNS		Proposed ALNS							
			Best				Average			
$H, \mathcal{C}$	c	Veh	s_0	$\Delta_D(\%)$	$\Delta_V(\%)$	Trans	CPU(s)	$\Delta_D(\%)$	$\Delta_V(\%)$	Trans	CPU(s)
$180, L$	6322.1	39.0	s_r	-46.6	-45.9	44.4	62.1	-45.1	-43.7	43.7	60.9
			s_t	-47.1	-46.2	45.2	55.8	-45.1	-43.6	44.0	60.9
$180, M$	4116.3	24.8	s_r	-31.2	-28.2	37.4	54.9	-29.2	-26.2	35.7	56.2
			s_t	-31.4	-29.0	37.6	55.8	-29.0	-25.6	36.2	55.9
$180, S$	2459.2	15.0	s_r	-4.9	-4.0	14.2	38.2	-2.7	-2.1	8.1	36.7
			s_t	-5.4	-2.7	20.5	47.2	-2.8	-0.3	15.8	44.5
$240, L$	3864.7	18.8	s_r	-28.5	-18.1	47.4	54.4	-23.2	-12.7	42.4	53.4
			s_t	-28.5	-17.6	46.3	55.5	-25.3	-13.8	45.3	57.6
$240, M$	3012.0	15.1	s_r	-15.2	-6.6	34.7	53.1	-8.4	-2.6	19.0	47.0
			s_t	-16.7	-7.3	38.4	58.9	-10.5	-2.1	29.1	53.1
$240, S$	2195.1	11.8	s_r	-2.2	0.0	12.5	47.6	-0.9	0.0	7.2	43.9
			s_t	-2.0	0.0	15.3	50.2	0.4	3.1	12.6	47.2
$300, L$	3653.1	16.6	s_r	-22.8	-5.4	46.7	56.9	-17.7	-2.2	40.9	54.3
			s_t	-23.1	-6.6	47.0	54.1	-17.5	-3.0	40.9	55.0
$300, M$	2952.8	14.4	s_r	-11.6	-1.4	33.5	55.0	-6.0	-1.3	17.5	45.7
			s_t	-14.4	-2.8	38.7	59.6	-9.0	1.8	30.2	53.1
$300, S$	2188.9	11.7	s_r	-1.3	0.0	4.3	42.3	-0.4	-0.5	2.2	41.6
			s_t	-2.2	0.9	17.9	53.1	0.9	4.4	19.1	51.7

Table 6: Results for generated instances with 50 requests with 3h time windows in the MD-5T setting

Inst.	RP ALNS		Proposed ALNS							
			Best				Average			
$H, \mathcal{C}$	c	Veh	$\Delta_D(\%)$	$\Delta_V(\%)$	Trans	CPU(s)	$\Delta_D(\%)$	$\Delta_V(\%)$	Trans	CPU(s)
$180, L$	8,883.3	53.8	-50.0	-47.6	70.8	142.1	-48.0	-45.7	68.4	159.5
$180, M$	5,166.8	30.8	-31.0	-26.6	57.0	134.9	-27.8	-23.2	49.6	137.8
$180, S$	3,268.6	19.6	-3.3	2.0	42.5	127.0	-0.5	3.8	36.7	133.7
$240, L$	5,162.3	24.7	-29.1	-17.0	71.4	164.7	-25.9	-13.5	68.7	162.3
$240, M$	3,743.1	18.7	-14.2	-1.1	62.5	156.6	-9.3	0.2	46.1	146.0
$240, S$	2,932.6	15.5	0.6	7.7	26.4	127.6	3.3	11.6	38.0	143.5
$300, L$	4,941.4	22.6	-25.5	-11.9	71.8	177.0	-20.3	-6.8	63.5	158.8
$300, M$	3,668.5	17.8	-12.2	1.7	64.7	180.4	-7.1	4.3	47.2	161.5
$300, S$	2,924.9	15.7	0.1	5.7	50.1	136.2	4.5	11.0	49.8	141.4

Table 7: Results for generated instances with 75 requests with 3h time windows in the MD-5T setting

operators are rewarded scores based on whether or not an operator advances the search, i.e., obtains a new best solution, obtains an improving solution, or obtains a worse, but accepted solution). Therefore, to analyze the performance of an operator during the search, we report how the weight of the operator changes during the search and its contribution to advancing the search (the ratio of the number of times the operator advanced the search and the total number of times the search advanced).

Figures 6 and 7 show the progressing of the operators for instances I_1 and I_2, respectively.

Inst.	RP ALNS		Proposed ALNS							
			Best				Average			
$H, \mathcal{C}$	c	Veh	$\Delta_D(\%)$	$\Delta_V(\%)$	Trans	CPU(s)	$\Delta_D(\%)$	$\Delta_V(\%)$	Trans	CPU(s)
$180, L$	11613.5	69.7	-53.1	-50.5	97.6	277.6	-51.1	-48.2	95.4	300.2
$180, M$	7309.3	43.3	-35.4	-30.9	90.6	261.2	-35.2	-30.5	87.2	285.9
$180, S$	4071.9	24.0	-5.9	0.8	67.9	280.3	-2.4	3.0	50.6	280.8
$240, L$	6481.8	30.6	-30.9	-17.3	96.4	295.5	-27.3	-13.5	94.4	290.8
$240, M$	5069.5	24.9	-20.8	-7.2	93.0	310.8	-16.7	-4.3	85.9	322.1
$240, S$	3618.9	19.3	-3.0	6.2	73.3	298.0	0.0	6.4	53.3	294.8
$300, L$	6186.0	28.2	-26.0	-9.9	98.0	332.8	-22.3	-6.0	94.1	322.1
$300, M$	4985.4	23.7	-17.9	-1.7	93.7	386.2	-12.1	0.5	73.5	326.7
$300, S$	3619.7	19.1	-2.1	7.3	59.7	317.1	1.2	7.0	45.0	288.5

Table 8: Results for generated instances with 100 requests with 3h time windows in the MD-5T setting

In each figure, we show, separately, the progression of the removal and the insertion operators. We use a stacked bar graph to show the progression of the contribution to advancing the search and a line plot to show the progression of the weight. The values are collected at the end of every 500 iterations (2κ). The graphs on the left (labeled (a) on both Figures 6 and 7) show results obtained when using an initial solution without transfers, and the graphs on the right (labeled (b) on both Figures 6 and 7) show results obtained when using an initial solution with transfers.

Note that the stacked bars for the removal operators always sum up to 1, whereas some of the stacked bars for an insertion operator (with or without transfers) sum up to less than 1. This is due to the fact that two insert operators are selected at each iteration. If the first operator (with the largest weight) succeeds inserting all requests, then the second one is not used. Thus, for some iterations the search is advanced after applying only one of the selected operators.

Figure 6 shows that for both initial solutions, insertion operators using transfers are the ones contributing the most for advancing the search (note that the scales of the vertical axis of the weight charts are not same). The stacked bars show that, in many iterations, an insertion with transfer operator alone is able to advance the search (because in many cases the stacked bars for the insertion without transfer operators sums up to less than 1, indicating that the they were not used). The larger weights of the insertion with transfer operators also demonstrate that these operators are more successful in finding new best and better solutions. We see that insertion with transfer operator O_8^+ is mostly used at the start of the search process and more so when the initial solution does not have transfers. None of the other three insertion with transfer operators clearly dominates the others. However, we see that when the initial solution already has transfers, insertion with transfer operator 0_5^+ is able to quickly find good transfer options for the requests with long distances between pickup and delivery locations (the initial spike in the weight chart shows the operator finds new best solutions). Regarding removal operators, note the important contribution of the transfer-based removal O_4^-, especially when starting with a solution without transfers.

When an instance benefits little from transfers, as is the case for instance I_2, the proposed ALNS algorithm recognizes this and adjusts accordingly (Figure 7). When the initial solution

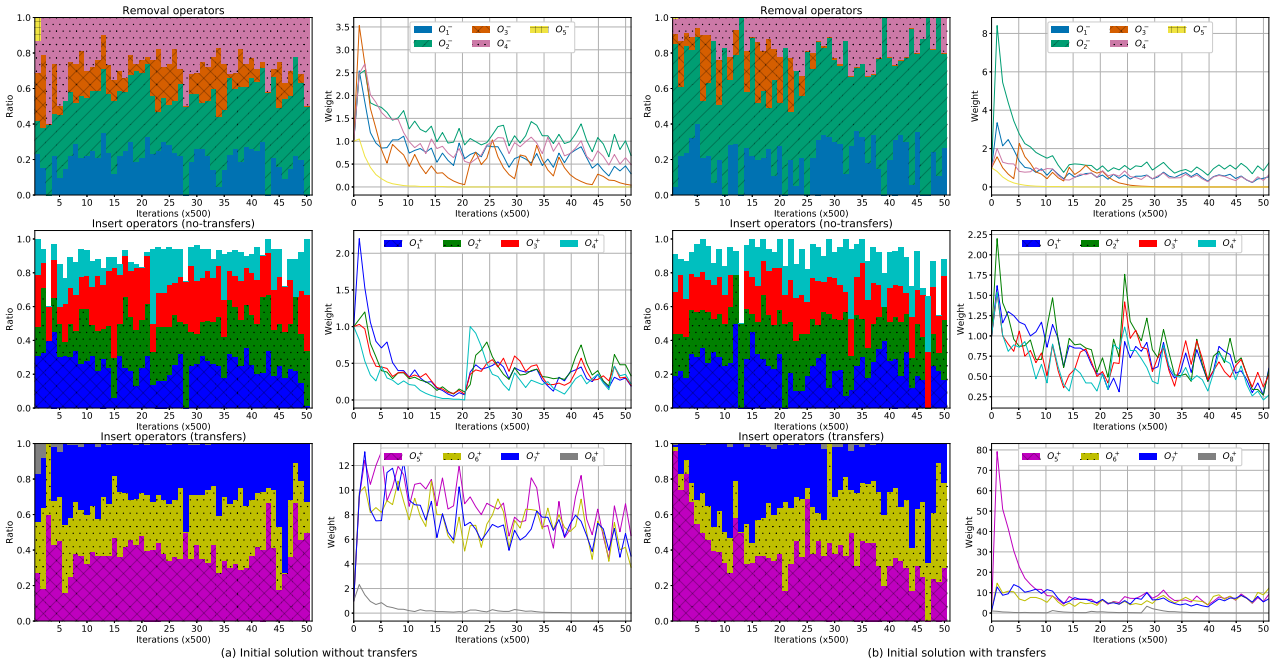


Figure 6: Progression of operators during the search when solving instance I_1 starting from an initial solution without transfers (left) and with transfers (right). Removal operators: O_1^- – worst request; O_2^- – random request; O_3^- – route; O_4^- – transfer-based request; O_5^- – cluster. Insertion operators without transfers: O_1^+ – travel time; O_2^+ – insertion cost; O_3^+ – regret; O_4^+ – random. Insertion with transfers: O_5^+ – travel time; O_6^+ – insertion cost; O_7^+ – regret; O_8^+ – transfer.

has no transfers, the search advances mainly using insertion without transfer operators. Even though insertion with transfer operator O_8^+ introduces a few unnecessary transfer visits early in the search, the transfer location is eventually removed from the solution. When the initial solution has transfers, improving solutions involving transfers are found in the early iterations, but as the search progresses and the insertion without transfer operators successfully insert more and more requests, the search advances primarily due to these operators (the turning point appears somewhere around iteration 17000).

6 Final Remarks

Our research has focused on investigating the potential benefits of using transfers in urban freight delivery systems. Our results show that these benefits can be significant, especially in settings where pickup and delivery locations are relatively far apart and driver shifts are relatively short. Thus, transfers may be especially valuable for urban freight delivery systems that rely, in part or completely, on crowdsourced delivery capacity, because crowdshippers tend to work for relatively short periods of time. Our results also show that the ability to transfer requests can substantially reduce the number of drivers required to serve a given number of requests. This too may be especially valuable in urban freight delivery systems that rely on crowdsourced delivery capacity, because crowdshippers are often compensated based on the number of requests they serve, and when the number of requests served per driver increases, it will become easier to attract crowdshippers to the system.

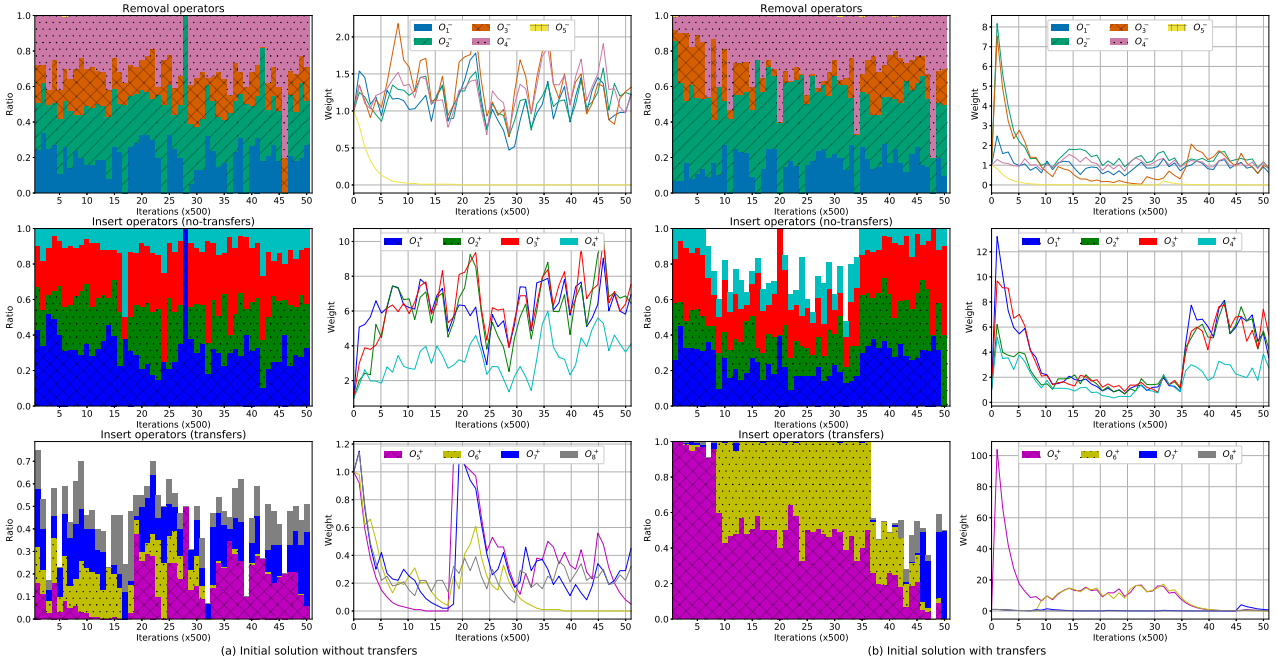


Figure 7: Progression of operators during the search when solving instance I_2 starting from an initial solution without transfers (left) and with transfers (right). Removal operators: O_1^- – worst request; O_2^- – random request; O_3^- – route; O_4^- – transfer-based request; O_5^- – cluster. Insertion operators without transfers: O_1^+ – travel time; O_2^+ – insertion cost; O_3^+ – regret; O_4^+ – random. Insertion with transfers: O_5^+ – travel time; O_6^+ – insertion cost; O_7^+ – regret; O_8^+ – transfer.

We are currently modifying our ALNS algorithm for use in dynamic environments, where requests appear throughout the operating period and (some of the) drivers may also enter and leave the system during the operating period, as this better reflects real-life settings.

Acknowledgments

Funding for this research was provided by Netherlands Organization for Scientific Research (NWO), via the project DATAS for Cities (No. 438-15-507) and the Physical Internet Reisbeurzenprogramma (No. 439-17-703). The authors gratefully acknowledge the grants received to support their work.

References

[Archetti et al., 2016] Archetti, C., Savelsbergh, M., and Speranza, M. G. (2016). The vehicle routing problem with occasional drivers. *European Journal of Operational Research*, 254(2):472 – 480.

[Barr and Wohl, 2013] Barr, A. and Wohl, J. (2013). Exclusive: Wal-mart may get customers to deliver packages to online buyers. Reuters. https://tinyurl.com/yccje985. Accessed 16-October-2017.

[Behrend and Meisel, 2018] Behrend, M. and Meisel, F. (2018). The integration of item-sharing and crowdshipping: Can collaborative consumption be pushed by delivering through the crowd? *Transportation Research Part B: Methodological*, 111:227 – 243.

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

[Berbeglia et al., 2010] Berbeglia, G., Cordeau, J.-F., and Laporte, G. (2010). Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202(1):8–15.

[Buldeo Rai et al., 2017] Buldeo Rai, H., Verlinde, S., Merckx, J., and Macharis, C. (2017). Crowd logistics: an opportunity for more sustainable urban freight transport? *European Transport Research Review*, 9(3):39.

[Campbell and Savelsbergh, 2004] Campbell, A. M. and Savelsbergh, M. (2004). Efficient insertion heuristics for vehicle routing and scheduling problems. *Transportation Science*, 38(3):369–378.

[Chen and Pan, 2016] Chen, C. and Pan, S. (2016). Using the crowd of taxis to last mile delivery in e-commerce: a methodological research. In Borangiu, T., Trentesaux, D., Thomas, A., and McFarlane, D., editors, *Service Orientation in Holonic and Multi-Agent Manufacturing*, pages 61–70. Springer International Publishing.

[Chen et al., 2017a] Chen, C., Pan, S., Wang, Z., and Zhong, R. Y. (2017a). Using taxis to collect citywide e-commerce reverse flows: a crowdsourcing solution. *International Journal of Production Research*, 55(7):1833–1844.

[Chen et al., 2017b] Chen, W., Mes, M., and Schutten, M. (2017b). Multi-hop driver-parcel matching problem with time windows. *Flexible Services and Manufacturing Journal*.

[Cortés et al., 2010] Cortés, C. E., Matamala, M., and Contardo, C. (2010). The pickup and delivery problem with transfers: Formulation and a branch-and-cut solution method. *European Journal of Operational Research*, 200(3):711 – 724.

[Dayarian and Savelsbergh, 2017] Dayarian, I. and Savelsbergh, M. (2017). Crowdshipping and same-day delivery: Employing in-store customers to deliver online orders. *Optimization Online 2017-07-6142*.

[Ghilas et al., 2016] Ghilas, V., Demir, E., and Van Woensel, T. (2016). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows and scheduled lines. *Computers & Operations Research*, 72:12 – 30.

[Guastaroba et al., 2016] Guastaroba, G., Speranza, M. G., and Vigo, D. (2016). Intermediate facilities in freight transportation planning: A survey. *Transportation Science*, 50(3):763–789.

[Kafle et al., 2017] Kafle, N., Zou, B., and Lin, J. (2017). Design and modeling of a crowdsource-enabled system for urban parcel relay and delivery. *Transportation Research Part B: Methodological*, 99:62 – 82.

[Li et al., 2014] Li, B., Krushinsky, D., Reijers, H. A., and Woensel, T. V. (2014). The share-a-ride problem: People and parcels sharing taxis. *European Journal of Operational Research*, 238(1):31 – 40.

[Maknoon and Laporte, 2017] Maknoon, Y. and Laporte, G. (2017). Vehicle routing with cross-dock selection. *Computers & Operations Research*, 77:254 – 266.

[Masson et al., 2013] Masson, R., Lehuédé, F., and Péton, O. (2013). An adaptive large neighborhood search for the pickup and delivery problem with transfers. *Transportation Science*, 47(3):344–355.

[Masson et al., 2014] Masson, R., Lehuédé, F., and Péton, O. (2014). The dial-a-ride problem with transfers. *Computers & Operations Research*, 41:12 – 23.

[Mitrović and Laporte, 2006] Mitrović, S. and Laporte, G. (2006). The pickup and delivery problem with time windows and transshipment. *INFOR: Information Systems and Operational Research*, 44(3):217–227.

[Pisinger and Ropke, 2007] Pisinger, D. and Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403 – 2435.

[Qu and Bard, 2012] Qu, Y. and Bard, J. F. (2012). A GRASP with adaptive large neighborhood search for pickup and delivery problems with transshipment. *Computers & Operations Research*, 39(10):2439 – 2456.

[Rais et al., 2014] Rais, A., Alvelos, F., and Carvalho, M. (2014). New mixed integer-programming model for the pickup-and-delivery problem with transshipment. *European Journal of Operational Research*, 235(3):530 – 539.

[Ropke and Pisinger, 2006] Ropke, S. and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472.

[Sampaio et al., 2018] Sampaio, A., Savelsbergh, M., Veelenturf, L., and Woensel, T. V. (2018). Crowd-based city logistics. In Faulin, J., Grasman, S. E., Juan, A. A., and Hirisch, P., editors, *Sustainable Transportation and Smart Logistics: Decision-Making Models and Solutions*, chapter 15, pages 382–398. Joe Hayton.

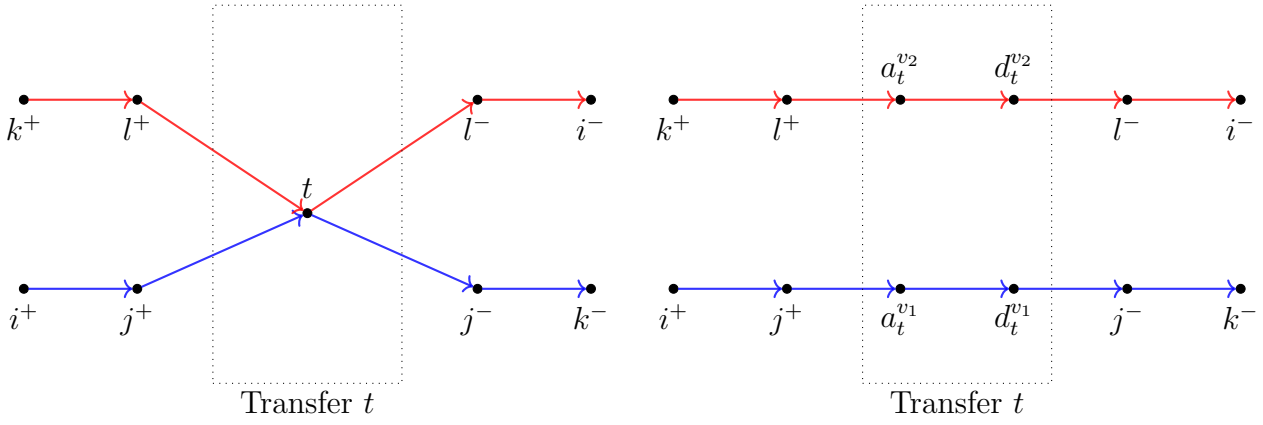
[Savelsbergh and Sol, 1995] Savelsbergh, M. W. P. and Sol, M. (1995). The general pickup and delivery problem. *Transportation Science*, 29(1):17–29.

[Stenger et al., 2013] Stenger, A., Vigo, D., Enz, S., and Schwind, M. (2013). An adaptive variable neighborhood search algorithm for a vehicle routing problem arising in small package shipping. *Transportation Science*, 47(1):64–80.

Appendix A Modeling Transfer Operations

In oder to model transferred requests along the routes of a solution, most works in the literature ([Mitrović and Laporte, 2006], [Masson et al., 2013], [Masson et al., 2014]) consider the duplication of requests at the transfer (i.e. request r_i is duplicated in (i^+, t_i^-) and (t_i^+, i^-)). In order

to avoid those duplications while modeling interactions between vehicles, we propose a model in which transfer locations are duplicated in inbound and outbound nodes for a given route at a transfer, regardless of how many requests are being transferred (dropped-off or picked-up) by a given route at a particular transfer location. Given a (partial) solution of the PDPTW-T, s, let $R_s \subset R$ be the set of requests served by this solution. The set of pickups (deliveries) serviced in s is denoted $P_s(D_s)$. If a vehicle v_1 visits a transfer location $t \in \Gamma$, nodes $a_t^{v_1}$ and $d_t^{v_1}$ representing the arrival (inbound) and departure (outbound) operations, respectively, of v_1 at t are created. Denote the set of transfers arrival's and departure's nodes in solution s by I_s and O_s, respectively. Figures 8a and 8b illustrate the correspondence between routing implementation and the model. Requests r_i and r_k are transferred between vehicles v_1 and v_2 at transfer location t.



(a) Routing implementation. Vehicle v_1 (blue) and v_2 (red) meet at transfer location t.

(b) Transfer representation in routes. $I_s = \{a_t^{v_1}, a_t^{v_2}\}, O_s = \{d_t^{v_1}, d_t^{v_2}\}$.

Figure 8: Modeling Transfer Operations

The *support graph* of solution s is the directed graph $G_s(W_s, A_s)$ where $W_s = M \cup P_s \cup D_s \cup I_s \cup O_s$ and A_s contains arcs (i, j) such that i and j are visited by the same vehicle and j is visited immediately after i, or $i \in I_s$, $j \in O_s$ such that i is an arrival vertex at a transfer for one vehicle and j the departure vertex at the same transfer location for another vehicle. Those last arcs (between arrival and departure nodes) capture relations between two vehicles at a transfer: if a vehicle v_1 transfers one or more requests to another vehicle, v_2, at transfer location t, then arc $(a_t^{v_1}, d_t^{v_2}) \in A_s$ i.e. the departure of v_2 at t depends on the arrival of vehicle v_1 at t. Figure 9 illustrates a solution with three vehicles and two transfers, t_1, t_2. At transfer location t_2, vehicle v_1 receives request r_i from vehicle v_2, and vehicle v_2 receives requests r_j and r_l from v_1. Vehicle v_3 only visits location t_1 to drop-off request r_k, which is picked-up at t_1 by vehicle v_2.

A.1 Evaluating and updating solutions

Given a solution s and its associated support (precedence) graph G_s, a route for vehicle $v \in V$ is a trip starting at its correspondent depot, m_v, visiting a sequence of vertices $i \in P \cup D \cup I_s \cup O_s$ and back to m_v. Let $k_v = \{m_v, i_1, i_2, ..., i_n, m_v\}$ represent the sequence visited by vehicle $v \in V$ in solution s and denote by ρ_i and σ_i the predecessor and successor sets, respectively, of vertex i on its route. Note that vertices $i \in P \cup D$ have exactly one direct predecessor and one direct

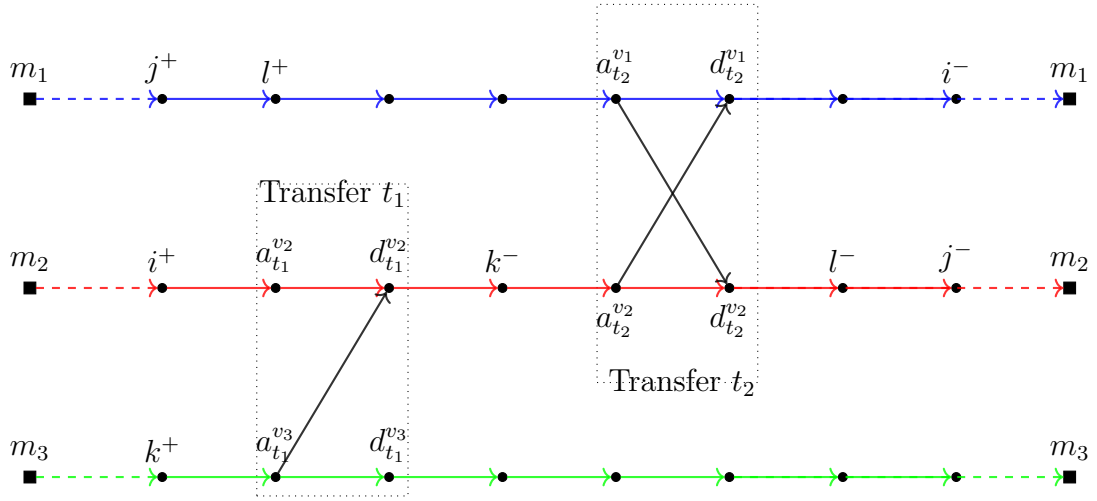


Figure 9: Vehicles v_1 and v_2 are synchronized at t_2 by $(a_{t_2}^{v_1}, d_{t_2}^{v_2})$ and $(a_{t_2}^{v_2}, d_{t_2}^{v_1})$, and vehicles v_2 and v_3 by arc $(a_{t_1}^{v_3}, d_{t_1}^{v_2})$. $I_s = \{a_{t_2}^{v_1}, a_{t_1}^{v_2}, a_{t_2}^{v_2}, a_{t_1}^{v_3}\}, O_s = \{d_{t_2}^{v_1}, d_{t_1}^{v_2} d_{t_2}^{v_2}, d_{t_1}^{v_3}\}$

successor (i.e. $|\rho_i| = |\sigma_i| = 1$), whereas vertices $i \in I_s \cup O_s$ (inbound and outbound operations at transfers) might have more than one successor ($i \in I_s$) or more than one predecessor ($i \in O_s$). With a slight abuse of notation, we refer to the (unique) direct predecessor (successor) for vertices $i \in P \cup D$ as ρ_i (σ_i).

Information maintained on each node

For every location $i \in P \cup D$ already assigned to a route, let e_i and l_i be the earliest time and the latest time, respectively, that service can start at location i (note that E_i and L_i are the time windows for request r_i, but e_i and l_i are variables changing accordingly to other elements in the route). For the vertices representing transfer locations in solution s, let e_i, l_i for $i = a_t^v \in I_s$, denote the earliest and latest possible arrival times for vehicle $v \in V$ at transfer $t \in \Gamma$, respectively. For $i = d_t^v \in O_s$, let e_i, l_i denote the earliest and latest possible departure times for vehicle v at transfer t. For each vehicle $v \in V$ in the solution, let $e_v = E_v$, the earliest time it can leave from its depot, and $l_v = L_v$, the latest time it can be back at its correspondent depot.

Given a solution s, earliest and latest times for visits in a route $\{i_0 = m_v, i_1, i_2, ..., i_n, i_{n+1} = m_v\}$ can be computed as follows:

- For $k = 1, ..., n$, let $e_{i_k} = \max\{E_{i_k}, e_{i_{k-1}} + \tau_{i_{k-1}, i_k}\}$ if $i_k \in P \cup D \cup I_s$, and $e_{i_k} = \max_{j \in \rho(i_k)} e_j$ if $i_k \in O_s$.
- For $k = n, ..., 1$, let $l_{i_k} = \min\{L_{i_k}, l_{i_{k+1}} - \tau_{i_k, i_{k+1}}\}$ if $i_k \in P \cup D \cup O_s$, and $l_{i_k} = \min_{j \in \sigma(i_k)} l_j$ if $i_k \in I_s$.

Checking the feasibility of insertions

Let $r_i = (i^+, i^-)$ a request to be inserted in a (partial) solution s, for which earliest and latest values are already computed for nodes visited by the routes in s. The feasibility of inserting i^+ between i_k and i_{k+1} is checked by computing:

$$e_{i^+} = \max\{E_{i^+}, e_{i_k} + \tau_{i_k, i^+}\}$$

and

$$l_{i^+} = \min\{L_{i^+}, l_{i_{k+1}} - \tau_{i^+,i_{k+1}}\}$$

if $e_{i^+} \leq l_{i^+}$ the insertion is feasible. Given that the insertion of i^+ is feasible, a feasible insertion for i^- can be searched on the same route wherein the insertion of i^+ was checked, or on a different route, using a transfer location. Checking the feasibility of inserting i^- consists in the same procedure described before, but considering updated values e_i, l_i for vertices i in all routes affected by the (feasible) insertion of i^+. Even if pickup and delivery are inserted on the same route, more than one route might need to be updated after an insertion due to the presence of transfers in the solution.

Updating routes after an insertion

The insertion of a node (either a pickup or delivery) affects nodes before and after it on route $k_v = \{m_v, i_1, i_2, ..., i_n, m_v\}$ for vehicle v. When the vehicle does not visit any transfer location, inserting i between nodes i_k and i_{k+1} might modify earliest times for nodes $i_{k+1}, ..., i_n$ and latest times for nodes $i_1, ..., i_k$: the detour time to visit the inserted location might require that subsequent visits have to start later (increased earliest time) and service at prior visits might not be postponed as much later as before (decreased latest time) [Campbell and Savelsbergh, 2004].

When route $k_v = \{m_v, i_1, i_2, ..., i_n, m_v\}$ visits one or more transfer locations, an insertion can alter earliest and latest times of nodes in the routes of vehicles transshipping requests with v at any of those transfer locations. Figure 10 illustrates a simple example with two vehicle routes, v_1 and v_2 (blue and red routes, respectively). If node i is to be inserted in route v_1, between nodes i_k and i_{k+1}, then, due to the fact that v_1 transfers one or more requests to v_2 at transfer location $t \in \Gamma$ (the arc (a_t^1, d_t^2) indicates this fact), vehicle v_2 can only leave transfer t after the arrival of vehicle v_1 i.e. $e_{d_t^2} \geq e_{a_t^1}$. Thus, earliest times for nodes visited by v_2 in the sub-path starting at d_t^2 onwards, until the depot, might have to be updated. If v_2 transfer any request to other vehicles at any other transfer in that sub-path (i.e. an arc $(a_t'^2, d_t^w)$ for a transfer $t' \in \Gamma$ and a vehicle $w \in V$), then earliest times for nodes visited by those vehicles might also have to be updated.

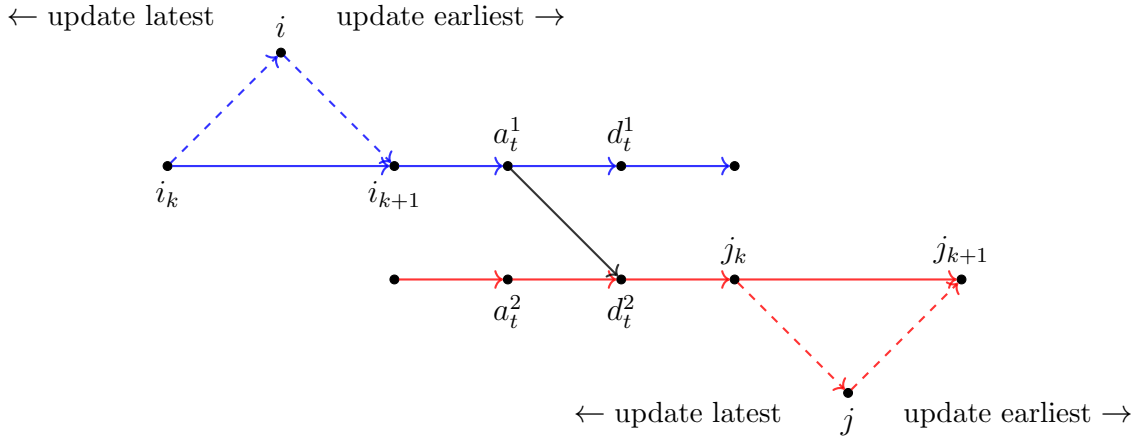


Figure 10: Insertion and update procedure

Similarly, inserting j in the route $k_{v_2} = \{m_{v_2}, j_1, j_2, ..., j_n, m_{v_2}\}$ for vehicle v_2, between nodes j_k and j_{k+1} can alter the latest service times for nodes visited by vehicle v_1 since the the latest

time vehicle v_1 can arrive at t ($l_{a_t^1}$) is bounded by the latest departure time at transfer t for vehicle v_2 ($l_{d_t^2}$). Latest times for nodes visited by other routes connected with v_1 at transfer locations in the sub-path starting at a_t^1 backwards might also need to be updated.

Observe that if node i was inserted in the route for vehicle v_1 after the visit to transfer t, then this insertion would not require that latest times for nodes in the route for vehicle v_2 to be updated. Similarly, if j was inserted in the route for vehicle v_2 before the visit to transfer t, then earliest times for nodes visited by vehicle v_1 would not need to be updated.

Updating is performed similarly to the process for computing all earliest and latest times for a solution, but taking into account that the solution already contains previously computed values e_i and l_i, so only locations that might be affected by the insertion are considered. After the insertion of node i between nodes i_k and i_{k+1} in the route $\{m_v, i_1, i_2, ..., i_n, m_v\}$, the update procedure is as follows:

- For $k' = k+1, ..., n$, let $e_{i_{k'}} = \max\{E_{i_{k'}}, e_{i_{k'-1}} + \tau_{i_{k'-1}, i_{k'}}\}$ if $i_{k'} \in P \cup D \cup I_s$, and $e_{i_{k'}} = \max_{j \in \rho(i_{k'})} e_j$ if $i_{k'} \in O_s$.
- For $k' = k, ..., 1$, let $l_{i_{k'}} = \min\{L_{i_{k'}}, l_{i_{k'+1}} - \tau_{i_{k'}, i_{k'+1}}\}$ if $i_{k'} \in P \cup D \cup O_s$, and $l_{i_{k'}} = \min_{j \in \sigma(i_{k'})} l_j$ if $i_{k'} \in I_s$.

To speed up the procedure, observe that whenever an earliest or latest update does not change the original value, updating can stop. This is especially useful since that might avoid unnecessary updates for routes connected at transfers. For example, after the insertion of j between j_k and j_{k+1}, if the latest time for node d_t^2 does not change, it is not necessary to continue with the update for v_2 and to update v_1 backwards from a_t^1.

Appendix B Complete Results

	$H = 180$			$H = 240$			$H = 300$		
$\mathcal{C}$	Dist	Veh	CPU(s)	Dist	Veh	CPU(s)	Dist	Veh	CPU(s)
L	6606	41	18.8	4040	20	30.9	3815	17	33.1
	6185	38	18.6	3697	18	32.7	3376	15	34.3
	6108	37	19.6	4122	21	31.2	3814	17	32.9
	6006	37	19.3	3844	19	30.7	3727	17	32.8
	6012	37	18.6	3739	18	31.6	3648	17	34.0
	6385	39	18.4	3683	17	33.2	3541	16	34.9
	6655	41	16.6	3828	19	31.4	3644	17	33.4
	5856	36	19.4	3950	19	32.2	3634	16	34.4
	6710	42	17.3	3873	19	31.6	3703	18	33.9
	6698	42	17.6	3871	18	31.4	3629	16	33.8
M	3616	22	25.7	2747	14	36.8	2713	13	37.5
	3642	22	24.7	2722	14	36.4	2744	13	37.5
	3723	23	23.8	2889	15	34.9	2762	14	36.3
	4423	26	22.6	3285	17	33.8	3132	15	35.1
	4586	28	21.3	3168	15	34.0	3108	15	35.3
	4728	28	22.7	3179	15	34.6	3095	15	35.7
	3977	24	22.9	2977	15	33.5	2880	14	36.2
	4415	27	23.3	3187	16	34.2	3161	16	34.1
	4018	24	24.5	3141	16	34.1	3124	15	34.7
	4035	24	24.1	2825	14	36.3	2809	14	37.1
S	2440	15	30.4	2207	12	38.6	2217	12	39.3
	2477	15	27.8	2301	13	37.4	2293	12	38.5
	2386	15	31.2	2167	12	39.3	2158	12	39.4
	2823	17	27.8	2451	13	36.7	2454	13	38.6
	2554	16	28.5	2303	12	38.4	2283	12	39.9
	2571	16	30.2	2243	12	38.9	2243	12	38.8
	2352	14	29.9	2208	12	39.1	2203	12	39.2
	2468	15	30.5	2129	11	40.1	2102	11	40.7
	2206	13	31.4	1970	11	41.6	1970	11	41.8
	2315	14	30.3	1972	10	40.8	1966	10	41.6

Table 9: Results obtained with the ALNS proposed by [Ropke and Pisinger, 2006] for instances with 50 requests.

	$H = 180$			$H = 240$			$H = 300$		
$\mathcal{C}$	Dist	Veh	CPU(s)	Dist	Veh	CPU(s)	Dist	Veh	CPU(s)
L	8467	51	103.3	5019	24	214.6	4795	21	202.75
	9125	55	80.5	5312	25	225.6	5135	24	220.0
	9399	58	82.6	5478	26	180.3	5217	24	185.1
	8105	49	86.5	4969	24	197.6	4794	22	164.5
	9764	59	79.9	5636	27	200.8	5478	25	181.4
	9071	55	114.4	5329	25	221.3	5062	23	198.2
	8759	53	118.3	5014	24	222.4	4954	23	169.8
	8299	50	118.4	4787	23	177.0	4540	21	178.0
	8869	54	105.8	5158	25	191.9	4871	22	200.7
	8975	54	86.9	4921	24	210.0	4568	21	208.4
M	5640	34	124.8	3866	19	218.5	3790	18	260.4
	4409	27	190.6	3405	17	262.0	3370	17	204.1
	6278	37	128.6	4310	21	191.4	4234	20	218.2
	5127	30	179.7	3685	18	236.9	3537	17	237.1
	4769	29	161.7	3650	19	204.6	3566	18	224.8
	5339	32	136.5	3943	20	201.2	3768	18	267.0
	5164	31	130.9	3655	18	183.5	3640	17	228.4
	5294	31	124.0	3787	19	194.3	3676	18	250.1
	4570	27	180.7	3493	18	241.9	3468	17	260.2
	5078	30	155.0	3637	18	190.1	3636	18	230.0
S	3106	19	190.6	2831	15	223.3	2833	16	188.4
	3068	18	176.6	2831	15	173.6	2793	15	174.6
	3345	20	174.8	2904	15	181.4	2863	15	196.1
	3430	20	176.3	3026	16	182.8	3047	16	179.3
	3427	21	192.1	3056	16	182.8	2986	16	195.2
	3165	19	188.6	2837	15	189.0	2836	15	188.2
	3258	20	187.4	2993	16	192.1	2990	16	175.9
	3281	20	171.5	2963	16	175.4	2963	16	176.1
	3133	18	214.7	2817	15	174.6	2829	15	173.5
	3473	21	169.5	3068	16	160.7	3109	17	158.5

Table 10: Results obtained with the ALNS proposed by [Ropke and Pisinger, 2006] for instances with 75 requests.

	$H = 180$			$H = 240$			$H = 300$		
$\mathcal{C}$	Dist	Veh	CPU(s)	Dist	Veh	CPU(s)	Dist	Veh	CPU(s)
L	11572	70	176.3	6649	32	360.8	6345	29	425.5
	11989	73	161.4	6574	31	368.1	6178	29	427.9
	11716	70	163.5	6723	32	360.5	6189	27	437.8
	11185	66	175.1	6306	29	370.9	5859	26	450.2
	11965	72	144.3	6515	31	358.7	6226	29	422.0
	11684	71	166.6	6499	31	382.9	6408	30	430.8
	12137	73	161.7	6640	31	391.5	6440	29	426.0
	10812	64	162.1	6013	28	430.7	5914	27	437.5
	11892	71	158.9	6771	32	405.1	6274	28	424.5
	11183	67	169.3	6128	29	407.3	6027	28	443.5
M	8221	50	235.7	5286	26	445.4	5273	25	463.7
	7219	42	255.2	5216	25	456.2	5060	24	462.2
	6611	39	274.9	4975	25	450.6	4892	23	463.6
	7429	44	235.9	5259	26	430.1	5134	25	444.7
	7409	43	253.3	5040	25	457.5	4917	23	459.2
	7692	46	231.0	5270	26	433.1	5210	25	457.8
	6574	38	283.3	4786	24	456.3	4803	24	467.4
	7255	43	266.4	4796	23	466.3	4662	22	468.4
	8314	50	238.8	5420	26	451.8	5369	25	448.4
	6369	38	290.0	4647	23	456.5	4534	21	476.3
S	3945	24	350.2	3539	19	490.1	3525	19	488.1
	4062	24	332.3	3540	19	493.8	3535	19	499.3
	4102	24	338.5	3623	19	484.8	3661	19	500.1
	4131	24	341.4	3705	20	476.5	3713	19	475.3
	4138	24	329.0	3674	19	472.1	3696	19	449.4
	3951	23	321.4	3520	19	491.0	3532	19	455.6
	4123	24	333.0	3720	20	472.1	3740	20	432.2
	4153	25	333.0	3607	19	484.3	3541	18	446.8
	4219	25	326.9	3753	20	474.4	3770	20	426.3
	3895	23	346.9	3508	19	493.0	3484	19	442.3

Table 11: Results obtained with the ALNS proposed by [Ropke and Pisinger, 2006] for instances with 100 requests.

	Proposed ALNS															
	Initial solution without transfer(Ropke)								Initial Solution with transfers							
	Best of 5				Average of 5				Best of 5				Average of 5			
$\mathcal{C}$	Cost	Veh	Trs	CPU(s)	Cost	Veh	Trs	CPU(s)	Cost	Veh	Trs	CPU(s)	Cost	Veh	Trs	CPU(s)
L	3547	22	47	60.5	3708.8	23.6	45.2	65.08	3468	22	47	58.7	3651.2	23.4	46.4	63.8
	2995	19	47	59.9	3048.8	19.0	45.0	58.0	3013	19	42	58.8	3083.0	19.6	43.2	59.8
	3335	21	48	65.4	3480.8	22.2	44.8	62.4	3366	21	48	60.6	3426.2	21.2	46.4	62.7
	3225	20	48	52.7	3296.4	20.8	47.6	56.1	3278	20	43	57.4	3315.0	21.2	45.6	54.1
	3432	23	50	58.3	3554.0	23.2	46.8	61.3	3593	24	46	60.3	3636.5	23.0	44.0	67.9
	3293	21	45	50.9	3423.8	21.8	45.0	56.5	3336	22	48	49.1	3513.7	22.7	44.5	64.4
	3365	21	47	65.1	3430.2	21.8	46.2	57.8	3306	21	46	58.8	3368.0	22.0	47.2	59.2
	3250	21	48	57.3	3404.0	21.6	46.2	57.2	3437	21	44	66.1	3617.0	22.7	41.7	60.6
	3338	21	45	53.6	3473.2	22.2	44.0	58.6	3431	22	45	55.9	3570.2	22.7	43.5	60.1
	3304	21	47	60.8	3465.4	21.8	44.6	61.9	3451	22	48	60.8	3502.0	22.0	46.2	61.1
M	2720	17	42	46.4	2815.0	17.6	20.8	45.0	2660	17	40	46.7	2740.6	17.6	38.4	52.1
	2729	18	41	48.5	2877.4	18.4	25.2	47.6	2665	17	38	44.7	2815.4	18.0	31.6	51.5
	2605	16	37	48.0	2704.4	17.0	37.6	54.9	2614	16	36	67.6	2664.5	17.25	38.7	58.0
	3052	19	35	58.0	3323.0	21.2	24.6	50.6	3054	20	38	53.9	3273.5	20.5	26.2	52.4
	2896	18	43	49.8	2946.4	18.8	39.8	47.9	2872	19	44	54.9	2893.0	18.25	42.2	54.6
	3039	20	41	49.1	3213.4	20.4	42.6	50.1	2968	18	42	54.6	3072.7	19.0	40.2	56.1
	2798	18	40	51.7	2904.6	18.6	30.0	49.0	2743	17	43	53.6	2787.6	17.6	38.2	53.4
	2829	17	40	59.4	3062.0	19.2	35.2	52.4	2913	18	40	56.0	3130.2	19.8	39.6	56.1
	2994	19	41	54.7	3091.8	19.4	34.4	54.2	3052	19	40	63.0	3169.8	20.0	36.8	55.3
	2862	18	41	43.8	2936.2	18.4	38.6	48.6	2905	18	35	54.5	2923.6	18.2	38.6	50.9
S	2349	15	14	36.1	2399.2	15.4	19	40.9	2342	15	40	62.2	2408.4	15.2	20.4	50.1
	2202	14	6	43.8	2237.8	14.0	8.4	42.8	2432	15	4	36.6	2446.8	15.0	8.6	39.9
	2685	16	7	43.8	2712.6	16.4	10.8	47.4	2203	14	37	62.4	2234.2	14.4	34.8	50.7
	2489	15	7	34.8	2523.8	15.6	2.2	40.6	2600	16	24	56.7	2652.4	16.4	23.2	49.0
	2685	16	7	43.8	2712.6	16.4	10.8	47.4	2438	16	28	52.8	2488.0	15.6	18.0	43.9
	2489	15	7	34.8	2523.8	15.6	2.2	40.6	2356	16	27	43.5	2444.4	15.8	15.0	41.2
	2339	14	2	30.3	2349.4	14.0	0.4	36.9	2354	14	0	33.4	2384.4	14.4	4.8	46.3
	2359	15	16	30.0	2417.4	15.0	9.0	28.8	2210	14	37	33.0	2371.4	14.6	14.2	37.5
	2157	13	18	32.6	2192.4	13.0	4.2	30.5	2160	14	6	33.9	2211.4	14.6	17.8	34.9
	2265	13	2	28.2	2293.0	13.8	1.0	27.9	2290	14	1	26.6	2315.0	14.0	2.8	27.7

Table 12: Results on instances with 50 requests for vehicle shift length $H = 180$ and MD-4T setting

	Proposed ALNS															
	Initial solution without transfer(Ropke)								Initial Solution with transfers							
	Best of 5				Average of 5				Best of 5				Average of 5			
$\mathcal{C}$	Cost	Veh	Trs	CPU(s)	Cost	Veh	Trs	CPU(s)	Cost	Veh	Trs	CPU(s)	Cost	Veh	Trs	CPU(s)
L	2741	16	48	55.9	2826.4	15.8	46.6	56.7	2737	15	50	49.1	2879.8	15.2	46.4	57.7
	2655	15	49	40.6	2770.6	15.2	39.8	46.5	2600	14	47	49.3	2783.8	15.6	40.4	49.0
	2799	16	49	58.6	3160.2	17.4	39.4	52.4	2758	15	49	53.5	2971.4	16.6	47.4	49.0
	2955	18	49	55.1	3182.8	17.4	33.2	52.1	2864	17	50	63.7	2955.0	17.0	45.0	60.0
	2892	16	45	46.9	3293.4	16.8	24.4	46.6	2917	16	45	55.1	3019.6	17.4	43.8	53.0
	2739	16	48	65.7	2888.6	16.0	44.8	60.2	2771	16	48	48.3	2958.6	16.4	43.0	58.9
	2968	17	47	54.6	3264.4	17.2	25.6	45.2	2760	15	46	61.0	2940.6	16.2	44.0	55.7
	3000	16	46	59.2	3345.6	17.6	30.0	50.8	3069	17	47	61.7	3140.8	17.4	43.0	56.4
	3125	17	47	53.8	3289.0	17.4	36.8	49.7	3072	16	42	51.7	3287.4	17.2	33.0	47.8
	2897	16	49	49.5	3127.0	17.2	41.8	50.5	2964	17	48	57.1	3010.8	17.2	46.6	53.1
M	2575	13	6	38.3	2636.4	13.8	9.8	40.88	2493	14	43	53.9	2595	14.2	25.8	47.0
	2488	13	10	44.7	2574.2	13.8	11.4	41.72	2327	13	39	64.2	2549	13.6	16.8	47.1
	2458	14	42	64.5	2641.2	14.2	23.4	50.62	2428	14	46	64.9	2544.4	14.2	36.6	55.2
	2976	16	15	39.9	3070	16	11.4	37.98	2941	17	44	34.2	3013.6	16.2	25.8	41.8
	2574	15	36	58.8	2833.2	15.2	21.6	46.72	2426	14	40	56.2	2693	15.2	33	58.1
	2651	14	40	63.1	2976	15	14.6	43.8	2660	15	42	62.5	2746	15	40.2	55.0
	2523	14	44	54.2	2716.8	14.4	20.8	48.28	2420	13	43	48.5	2671.6	15.2	32.8	51.4
	2765	16	48	57.7	2882.6	15.8	26.6	50.52	2550	15	37	55.7	2897.2	16	22.8	46.9
	3000	15	13	44.6	3058	15.4	8.8	42.92	2711	14	39	54.5	2909.8	15	23.8	51.6
	2708	13	9	37.6	2781.8	13.8	5.4	39.2	2473	14	38	47.5	2706	14.4	15.6	43.2
S	2206	12	0	38.5	2206.8	12	0.0	40.2	2183	13	44	85.3	2277.8	12.8	19.4	58.0
	2292	13	0	37.2	2299.0	12.8	0.0	42.4	2282	13	1	41.4	2319.2	12.8	18.2	45.5
	2040	12	39	60.8	2135.6	12.2	17.6	48.4	2005	12	46	60.9	2168.8	12.4	35.0	54.4
	2322	13	38	54.7	2408.6	13.0	9.4	44.0	2400	13	37	52.0	2441.4	13.8	31.4	53.1
	2303	12	0	46.8	2303.0	12.0	0.0	41.9	2299	13	24	74.9	2327.4	13.0	13.2	51.3
	2243	12	0	33.2	2243.0	12.0	0.0	35.2	2243	12	0	37.5	2270.8	12.6	14.8	49.8
	2200	12	0	41.4	2206.0	12.0	0.2	39.7	2207	12	43	69.4	2234.6	12.2	15.0	46.8
	2105	11	1	41.7	2114.6	11.0	2.6	39.8	2098	11	4	32.8	2134.2	11.2	10.4	45.5
	1879	11	44	63.1	1939.2	11.0	11.2	46.2	1945	12	22	54.4	1980.0	11.2	14.4	45.3
	1966	10	0	39.0	1968.6	10.0	0.0	41.2	1966	10	0	43.9	1980.0	11.2	14.4	45.3

Table 13: Results on instances with 50 requests for vehicle shift length $H = 240$ and MD-4T setting

	Proposed ALNS															
	Initial solution without transfer(Ropke)								Initial Solution with transfers							
	Best of 5				Average of 5				Best of 5				Average of 5			
$\mathcal{C}$	Cost	Veh	Trs	CPU(s)	Cost	Veh	Trs	CPU(s)	Cost	Veh	Trs	CPU(s)	Cost	Veh	Trs	CPU(s)
L	2729	15	50	47.7	2877.0	15.6	46.4	56.4	2714	15	47	37.3	2894.2	15.8	46.4	51.3
	2620	14	46	45.6	2852.6	14.6	32.8	44.2	2660	15	48	48.2	2905.2	14.8	31.2	44.4
	2784	15	50	49.2	3135.4	15.4	28.6	46.4	2761	16	48	60.1	2836.8	16.0	47.0	57.7
	2963	16	44	58.9	3282.6	16.4	25.8	50.0	2859	16	50	54.1	2968.4	16.4	45.0	58.3
	2943	18	46	54.6	3422.0	16.8	15.6	46.1	3087	16	42	43.6	3275.4	16.4	33.6	51.0
	2943	16	46	56.6	3139.4	15.8	31.0	51.7	2838	16	46	53.7	2911.2	16.2	46.0	56.6
	2815	15	45	48.3	3085.0	16.4	37.2	52.9	2828	16	47	43.1	2973.0	17.0	45.8	53.3
	3017	17	46	79.4	3214.8	17.2	33.6	62.7	2942	17	49	53.6	3056.6	17.0	46.2	57.0
	3311	18	45	51.1	3440.8	16.6	18.8	44.5	2933	17	48	55.8	3145.6	16.6	39.6	53.1
	2962	17	47	60.1	3143.4	17.6	38.0	53.3	2986	16	48	48.2	3121.2	16.6	40.0	51.3
M	2552	13	6	45.6	2633.8	13.2	10.2	46.2	2321	13	48	47.7	2460.6	13.4	34.8	47.5
	2365	13	47	52.3	2467.0	12.8	23.6	48.1	2366	13	48	32.7	2469.6	13.2	19.8	45.8
	2541	14	46	65.5	2667.6	14.0	13.4	49.5	2485	14	43	55.0	2649.2	14.2	25.4	47.2
	3064	16	11	44.0	3099.0	15.6	8.2	41.8	3018	15	4	41.5	3075.0	15.6	15.4	39.8
	2752	16	46	47.1	2932.8	14.8	17.2	45.5	2473	14	47	51.2	2589.4	15.0	45.6	57.2
	2778	16	46	70.3	3006.8	15.2	12.4	46.7	2656	16	45	49.5	2763.4	15.2	39.2	56.2
	2581	14	38	61.1	2814.4	14.2	10.0	44.3	2762	13	6	45.1	2844.4	14.4	9.8	41.9
	2728	15	38	64.0	2975.0	15.2	14.8	47.8	2704	15	44	52.6	2902.6	15.8	30.4	62.8
	2947	15	7	43.1	2991.6	14.6	8.4	46.0	2848	16	45	50.8	2995.8	15.6	25.2	48.3
	2691	13	7	42.4	2717.8	13.6	7.0	40.8	2744	14	4	31.6	2769.0	13.8	2.8	38.1
S	2213	12	0	33.4	2215.4	12	0.0	38.9	2175	12	8	41.1	2258	12.4	18.4	53.1
	2282	12	1	39.3	2290.4	12.0	0.2	38.1	2292	13	0	40.1	2324	13.4	17.8	55.1
	2092	12	43	89.8	2128.8	11.8	19.2	57.5	2089	12	45	48.8	2122.8	12.6	36.8	56.1
	2399	12	3	38.9	2426.2	12.6	1.8	39.3	2432	13	7	43.9	2459.2	13.2	15.6	48.5
	2283	12	0	41.8	2283.0	12.0	0.0	37.9	2297	13	3	36.2	2331.6	12.8	8.6	39.6
	2243	12	0	39.3	2243.0	12.0	0.0	38.3	2154	11	8	34.3	2250.4	12.0	10.2	40.1
	2200	12	0	40.5	2202.2	12.0	0.6	42.8	2208	12	0	31.1	2285.4	12.6	15.2	47.7
	2078	11	3	34.8	2096.0	11.0	0.8	36.3	2097	11	12	45.5	2126.0	11.0	3.8	40.1
	1958	10	8	36.7	1967.6	10.8	1.6	39.6	1928	11	46	67.1	1969.2	11.0	9.4	45.9
	1966	10	0	41.5	1966.0	10.0	0.0	42.9	1997	10	0	35.5	2067.8	11.0	17.4	46.7

Table 14: Results on instances with 50 requests for vehicle shift length $H = 300$ and MD-4T setting

	Proposed ALNS															
	Initial solution without transfer(Ropke)								Initial Solution with transfers							
	Best of 5				Average of 5				Best of 5				Average of 5			
$\mathcal{C}$	Cost	Veh	Trs	CPU(s)	Cost	Veh	Trs	CPU(s)	Cost	Veh	Trs	CPU(s)	Cost	Veh	Trs	CPU(s)
L	3507	22	48	66.2	3717.4	23.8	44.2	68.5	3603	22	44	74.7	3700.4	23.8	44.6	68.9
	3048	19	44	68.3	3110.8	20.0	43.6	60.2	3055	20	47	65.2	3168.8	20.2	43.4	61.0
	3411	21	40	68.4	3480.0	21.6	39.6	63.3	3306	21	46	51.0	3376.2	21.4	45.2	59.4
	3236	20	46	62.8	3331.2	20.8	45.4	57.0	3212	20	47	53.2	3289.2	21.0	47.2	57.9
	3574	22	43	63.8	3671.6	23.4	42.2	61.9	3485	22	44	47.0	3730.6	23.4	38.4	62.1
	3355	21	46	64.5	3391.0	21.4	46.2	62.1	3341	22	47	59.3	3382.2	21.4	44.8	62.0
	3368	22	47	49.2	3463.6	22.2	46.0	57.6	3374	21	46	50.8	3469.6	22.0	46.8	60.0
	3422	22	42	62.9	3517.6	21.8	40.2	63.2	3445	21	43	47.1	3517.2	22.0	43.8	57.0
	3427	21	42	55.6	3563.4	22.8	42.8	58.1	3310	20	41	47.8	3545.8	22.4	41.6	60.4
	3405	21	46	59.4	3481.8	21.8	46.4	57.5	3336	21	47	61.4	3524.6	22.4	44.2	59.9
M	2723	17	35	65.0	2775.6	17.6	39.8	60.4	2574	16	39	58.5	2744.4	17.2	34.6	58.8
	2636	17	31	42.7	2816.2	17.6	23.6	49.1	2651	17	38	50.4	2817.0	18.0	28.8	46.1
	2588	16	46	56.7	2669.8	17.0	37.6	62.0	2595	16	39	55.2	2730.2	17.4	36.8	58.9
	2938	18	36	59.2	2999.4	18.8	35.4	57.0	2950	19	34	55.6	3003.0	19.2	36.6	56.7
	2898	18	35	58.2	2957.6	18.6	37.8	57.5	2886	18	40	53.6	2969.4	18.8	37.0	60.1
	2932	19	38	59.5	3028.2	19.0	38.6	60.3	2973	18	33	62.7	3124.4	19.6	37.8	62.5
	2722	17	43	47.5	2788.2	17.4	37.8	57.0	2749	17	37	52.6	2874.6	18.4	38.2	49.0
	3034	19	34	59.8	3161.0	19.8	33.4	54.0	2949	18	39	62.0	3013.4	18.6	36.4	56.5
	2854	18	37	49.8	2891.2	18.0	38.8	49.8	3067	19	39	42.9	3087.2	18.8	36.4	53.2
	3007	19	39	50.7	3046.2	19.2	34.2	55.4	2826	18	38	64.8	2863.0	18.4	39.4	57.0
S	2210	14	37	54.8	2343.2	14.4	18.2	42.3	2275	15	35	50.8	2351.2	15.0	23.0	47.6
	2402	14	3	40.0	2442.8	14.8	2.6	39.8	2378	15	32	53.4	2436.4	15.2	10.4	40.8
	2148	14	39	50.6	2232.4	14.2	21.2	42.2	2171	13	12	48.1	2265.4	14.4	22.4	44.6
	2624	16	31	50.9	2681.8	16.6	12.6	46.2	2594	16	24	50.9	2643.0	16.6	23.0	56.7
	2493	15	1	34.2	2531.2	15.4	3.4	38.7	2396	15	25	59.8	2492.8	15.6	14.6	45.0
	2375	15	6	36.6	2431.6	15.2	6.2	39.8	2333	15	34	51.2	2431.4	15.2	11.2	43.1
	2329	14	3	27.6	2347.4	14.0	0.6	29.3	2342	14	1	41.7	2390.4	15.0	25.4	48.7
	2402	15	5	27.3	2426.2	14.8	4.6	28.3	2378	15	19	39.7	2409.8	15.2	9.6	42.0
	2150	14	16	33.0	2185.4	13.6	11.4	32.6	2142	13	9	39.1	2173.0	13.0	11.6	41.7
	2263	13	1	27.4	2300.8	13.8	0.4	27.7	2264	15	14	37.3	2299.0	14.4	6.8	34.3

Table 15: Results on instances with 50 requests for vehicle shift length $H = 180$ and MD-5TC setting

	Proposed ALNS															
	Initial solution without transfer(Ropke)								Initial Solution with transfers							
	Best of 5				Average of 5				Best of 5				Average of 5			
$\mathcal{C}$	Cost	Veh	Trs	CPU(s)	Cost	Veh	Trs	CPU(s)	Cost	Veh	Trs	CPU(s)	Cost	Veh	Trs	CPU(s)
L	2606	14	48	44.6	2900.6	15.6	41.4	53.4	2727	15	46	70.0	2830	15.6	46.8	58.0
	2538	14	45	50.5	2782.4	15.2	39.4	48.9	2633	15	44	66.8	2783.8	15.4	39.6	56.1
	2810	16	48	52.6	2892.4	16.4	45.2	57.5	2768	16	48	43.2	3011.8	16.6	43.4	53.4
	2890	17	49	51.8	3000.0	16.8	43.6	55.3	2848	16	47	62.0	2948.6	16.4	46.6	59.3
	2851	16	49	53.0	3058.2	16.6	37.8	48.8	2902	16	47	51.9	3026.0	16.6	43.2	59.0
	2370	13	48	73.0	2813.2	15.4	45.4	59.9	2400	13	49	52.9	2542.8	14.4	48.2	60.7
	2797	15	45	53.4	2984.2	16.6	44.8	49.8	2810	15	43	43.6	2910.4	16.8	46.4	53.4
	2988	17	49	45.6	3158.4	17.2	39.8	54.1	2970	18	48	65.2	3020.6	17.2	45.0	61.5
	2859	16	47	65.3	3059.8	17.2	42.0	54.7	2790	16	48	48.3	2914.8	16.6	46.2	53.6
	2908	16	46	53.8	3041.6	17.2	44.6	51.7	2785	15	43	51.5	2895.4	16.4	47.4	61.4
M	2208	12	47	62.9	2534.6	13.4	16.8	46.0	2327	13	47	50.7	2512.6	13.8	31.2	47.5
	2571	14	12	36.7	2592.4	14.0	10.0	40.8	2352	14	42	50.8	2550.2	13.6	17.2	44.5
	2350	13	43	67.3	2629.0	14.2	23.6	51.3	2349	13	45	58.9	2644.0	14.0	22.4	53.2
	2702	15	30	38.4	2919.4	15.0	14.4	43.6	2675	15	40	67.1	2797.0	15.8	35.0	49.1
	2542	16	47	53.1	2701.0	15.6	35.8	55.1	2527	14	41	60.9	2695.4	15.2	37.8	59.4
	2581	14	43	62.0	2915.2	15.2	20.8	49.5	2507	14	39	66.6	2701.2	14.8	37.0	70.0
	2598	14	43	48.6	2820.6	14.8	16.4	42.9	2384	14	42	57.8	2687.4	15.2	33.8	57.0
	2531	14	34	69.6	2759.6	15.2	28.0	53.1	2577	15	39	69.6	2732.6	15.2	32.0	57.4
	2827	16	41	52.0	3000.6	15.8	14.8	47.3	2925	15	12	47.3	2988.2	15.8	21.4	47.7
	2639	13	7	40.7	2717.4	13.8	9.2	40.0	2465	13	37	59.4	2659.0	14.4	23.6	45.4
S	2124	13	43	67.1	2161.4	12.6	31.6	61.7	2111	13	34	80.5	2174.2	12.6	23.4	55.3
	2273	12	3	36.4	2291.6	12.6	0.8	38.2	2301	12	0	39.7	2324.8	12.8	4.2	48.7
	2042	12	44	69.4	2106.8	12	20.2	48.9	2086	12	44	64.7	2149.0	11.8	18.8	49.7
	2320	13	27	65.8	2415.4	12.8	6.6	47.9	2435	13	2	42.2	2437.8	13.2	24.6	57.7
	2303	12	0	37.0	2303.0	12	0.0	38.5	2292	12	3	50.4	2320.8	13.0	16.0	46.4
	2174	12	7	44.1	2221.8	12	8.8	45.0	2243	12	0	35.9	2265.2	12.4	9.6	41.6
	2200	12	0	41.1	2205.6	12	0.0	37.7	2212	12	0	43.9	2257.0	12.6	8.2	39.0
	2103	11	1	42.0	2111.8	11	3.8	41.7	1981	11	20	35.0	2092.8	11.0	5.6	38.7
	1970	11	0	42.3	1970.0	11	0.0	39.1	1894	11	47	73.0	1961.8	11.0	10.4	50.0
	1968	10	0	30.3	1971.2	10	0.0	40.0	1956	10	3	36.9	2062.4	11.2	5.6	44.8

Table 16: Results on instances with 50 requests for vehicle shift length $H = 240$ and MD-5T setting.

	Proposed ALNS															
	Initial solution without transfer(Ropke)								Initial Solution with transfers							
	Best of 5				Average of 5				Best of 5				Average of 5			
$\mathcal{C}$	Cost	Veh	Trs	CPU(s)	Cost	Veh	Trs	CPU(s)	Cost	Veh	Trs	CPU(s)	Cost	Veh	Trs	CPU(s)
L	2734	15	45	55.7	2805.2	15.6	47.0	54.7	2747	15	50	59.6	2794.8	15.2	46.4	58.5
	2587	14	46	61.5	2855.4	15.0	32.4	49.3	2636	14	48	44.3	2760.2	14.2	39.6	47.6
	2792	15	49	61.2	2909.0	16.2	46.6	55.7	2787	16	47	35.9	3106.6	16.4	38.6	49.9
	2942	16	44	55.7	3012.4	16.6	46.4	55.0	2930	16	40	45.6	3058.2	16.4	37.6	50.8
	2998	17	46	57.6	3228.8	16.6	30.6	51.9	2869	16	46	46.2	2966.4	16.0	45.4	58.0
	2445	14	49	61.9	2733.6	15.0	46.8	61.0	2542	14	50	39.7	2985.2	15.8	38.8	60.9
	2748	15	47	52.2	3170.0	16.8	35.6	50.8	2810	15	48	61.0	3032.0	16.0	40.4	51.4
	2964	17	49	43.5	3015.6	16.8	45.8	55.6	2960	16	46	61.6	3174.6	17.4	42.8	60.1
	3001	16	44	57.9	3166.8	16.4	39.2	54.5	2949	16	45	74.5	3045.2	16.4	45.8	56.6
	2993	18	48	61.5	3172.8	17.4	38.2	53.9	2879	17	50	72.5	3206.0	17.2	33.2	55.9
M	2310	13	44	53.2	2538.2	13.4	15.8	44.1	2504	13	13	46.4	2604.2	13.6	7.2	43.9
	2394	13	47	64.2	2523.6	13.2	23.8	45.2	2486	14	39	51.3	2538.8	13.2	21.4	45.0
	2375	13	42	57.9	2664.2	13.6	12.2	45.2	2424	13	38	65.9	2499.8	14.0	36.6	52.7
	2714	15	39	74.2	2923.8	15.0	20.0	47.7	2642	15	45	66.8	2934.8	16.6	34.8	49.5
	2535	15	46	60.5	2623.4	14.2	44.0	57.3	2466	13	41	60.3	2770.4	15.2	33.8	56.2
	2611	15	45	55.8	2886.0	14.8	22.8	48.6	2583	15	42	63.3	2732.0	14.8	33.8	58.4
	2834	14	6	40.2	2859.6	14.0	3.2	42.7	2586	15	41	56.4	2646.4	14.8	33.0	54.5
	2639	15	43	67.8	2946.2	15.4	16.6	43.7	2614	14	43	67.2	2701.2	14.6	35.6	61.5
	3049	15	10	40.0	3090.4	15.0	6.6	45.1	2579	14	43	38.4	2814.2	15.8	42.6	52.8
	2632	14	13	35.9	2688.0	13.6	9.8	37.7	2393	14	42	79.7	2615.2	14.0	23.2	56.3
S	2182	12	6	44.0	2208.8	11.8	2.8	41.2	2120	12	43	82.0	2210.8	12.8	27.6	61.2
	2280	12	0	37.2	2289.6	12.0	0.2	37.6	2301	13	0	40.7	2373.8	13.0	9.2	55.2
	2107	12	3	43.0	2147.8	12.0	0.6	41.2	2065	12	45	70.3	2114.4	12.2	35.6	60.5
	2358	13	19	46.9	2399.2	12.6	14.6	46.2	2266	12	20	52.9	2393.6	13.0	30.8	60.7
	2283	12	0	40.4	2283.0	12.0	0.0	38.4	2260	13	32	70.7	2333.4	13.2	23.6	57.7
	2236	12	0	33.6	2241.6	12.0	0.0	36.0	2243	12	0	44.2	2255.0	12.2	9.4	46.7
	2189	12	3	42.9	2199.6	12.0	0.6	42.4	2233	12	0	33.8	2284.0	12.4	14.0	42.2
	2063	11	9	40.5	2090.6	11.0	2.6	40.2	2061	11	21	62.3	2113.4	11.4	20.0	50.2
	1950	11	3	44.2	1966.0	11.0	0.6	43.5	1892	11	18	43.1	1940.4	11.0	7.2	40.9
	1966	10	0	50.5	1966.0	10.0	0.0	49.6	1966	10	0	30.6	2062.8	11.0	13.6	41.4

Table 17: Results on instances with 50 requests for vehicle shift length $H = 300$ and MD-5TC setting.

	Proposed ALNS: Initial Solution with transfers							
	Best of 5				Average of 5			
$\mathcal{C}$	Cost	Veh	Trs	CPU(s)	Cost	Veh	Trs	CPU(s)
L	4675	29	68	165.6	4753.6	29.8	67.4	203.2
	4455	28	68	138.0	4580.0	29.4	69.4	159.6
	4619	30	72	158.4	4737.6	30.2	64.8	159.9
	4072	26	74	99.6	4283.6	26.8	70.6	146.3
	5098	32	69	171.4	5224.8	33.0	69.6	182.4
	4439	29	72	127.2	4784.4	30.4	66.4	162.7
	4469	28	69	126.9	4537.4	28.2	67.8	142.9
	4055	26	73	182.7	4241.0	27.2	71.2	147.1
	4412	28	68	134.7	4705.6	29.4	67.8	142.3
	4139	26	75	116.6	4356.0	27.8	69.4	148.8
M	3726	23	60	134.7	3800.6	24.0	60.6	139.4
	3362	20	36	145.4	3489.8	21.4	24.4	125.6
	4125	27	64	126.1	4288.6	27.0	60.4	152.5
	3444	22	63	131.2	3695.4	24.0	53.4	143.7
	3522	23	61	114.0	3705.0	23.4	38.0	131.0
	3572	23	58	133.8	3742.0	24.0	58.2	144.0
	3411	22	56	168.1	3553.8	22.6	52.6	130.6
	3549	23	52	163.9	3681.6	23.8	61.4	155.7
	3428	21	59	144.5	3734.6	23.0	27.8	118.8
	3517	22	61	86.9	3620.0	23.2	59.2	136.9
S	3026	20	42	127.4	3093.6	20.0	42.4	157.5
	2900	19	52	187.4	3059.8	19.0	22.4	123.2
	3214	20	43	163.2	3348.4	21.2	44.6	149.0
	3259	21	58	137.3	3302.8	21.0	47.6	143.5
	3199	21	61	126.0	3313.0	21.2	49.0	138.6
	3070	19	62	97.2	3140.0	19.2	37.8	117.4
	3233	19	2	104.2	3323.4	20.2	23.8	137.9
	3192	21	56	122.5	3299.2	21.0	32.2	120.5
	3149	19	10	100.5	3204.8	19.2	18.6	135.4
	3378	21	39	104.6	3441.8	21.4	48.6	114.5

Table 18: Results on instances with 75 requests for vehicle shift length $H = 180$ and MD-5TC setting.

	Proposed ALNS: Initial Solution with transfers							
	Best of 5				Average of 5			
$\mathcal{C}$	Cost	Veh	Trs	CPU(s)	Cost	Veh	Trs	CPU(s)
L	3543	20	70	170.7	3614.0	20.4	72.0	174.3
	3648	20	73	164.9	3819.0	21.2	70.6	159.0
	3735	21	74	143.5	3845.4	22.0	71.2	169.4
	3498	19	72	112.0	3821.2	20.8	62.0	173.7
	4121	24	74	163.1	4380.6	24.4	67.6	164.2
	3753	21	72	220.1	3819.8	21.2	71.8	173.9
	3559	20	74	189.3	3875.2	21.0	60.8	154.3
	3594	20	68	115.7	3677.0	21.2	70.6	140.5
	3773	21	67	178.4	3882.2	21.8	68.4	145.8
	3377	19	70	189.6	3517.4	19.6	71.6	168.0
M	3245	18	67	222.3	3334.0	18.8	66.0	167.7
	3097	18	68	99.6	3250.8	18.4	51.6	140.0
	3523	20	70	184.8	3597.8	20.4	66.8	178.9
	3087	18	62	159.1	3192.0	18.6	64.0	151.1
	3390	19	23	107.0	3525.8	19.6	31.8	135.5
	3198	18	67	190.3	3470.4	18.8	45.8	149.5
	3067	18	65	189.4	3235.4	18.2	52.6	150.2
	3264	20	71	105.9	3630.2	19.2	20.2	118.4
	3167	18	63	108.9	3415.4	17.6	15.2	114.2
	3087	18	69	198.5	3293.6	17.8	46.6	154.6
S	2842	16	0	103.2	2942.0	17.4	50.8	192.7
	2789	15	2	84.5	2843.4	16.0	15.4	102.5
	3124	19	55	158.0	3226.6	18.4	53.8	145.2
	3069	18	69	205.0	3151.4	18.4	64.4	202.6
	2956	16	6	106.0	3047.0	16.8	27.6	124.2
	2869	17	67	177.4	2975.6	17.4	44.8	151.7
	2981	17	0	102.9	3062.0	17.8	38.8	157.9
	2983	16	0	105.0	3024.4	17.0	40.6	142.2
	2811	15	0	74.1	2867.2	15.6	0.6	97.2
	3073	18	65	159.8	3146.8	18.2	42.8	118.5

Table 19: Results on instances with 75 requests for vehicle shift length $H = 240$ and MD-5TC setting.

	Proposed ALNS: Initial Solution with transfers							
	Best of 5				Average of 5			
$\mathcal{C}$	Cost	Veh	Trs	CPU(s)	Cost	Veh	Trs	CPU(s)
L	3590	19	70	139.2	3828.2	20.2	59.2	158.4
	3694	21	73	267.3	3919.4	21.8	71.6	181.7
	3923	20	71	108.3	4372.8	21.8	49.4	146.5
	3453	19	73	216.3	3560.8	19.4	72.0	188.9
	4049	23	72	139.8	4290.0	22.8	61.2	160.5
	3797	20	73	220.8	3916.4	21.8	72.2	160.2
	3715	21	72	157.8	4122.8	21.2	48.8	135.2
	3454	18	70	200.0	3596.4	20.0	70.8	169.4
	3764	20	70	169.8	4058.6	22.4	68.0	136.0
	3373	18	74	150.3	3711.8	19.2	62.0	151.3
M	3301	19	63	124.1	3644.4	19.8	43.6	144.1
	2976	16	60	246.1	3150.0	17.4	55.6	184.7
	3476	19	67	221.2	3552.4	20	65.8	195.0
	3128	17	67	135.1	3296.0	18	53.4	160.5
	3370	19	62	125.1	3500.6	18.4	23.8	116.4
	3301	18	66	196.2	3365.6	18.8	64.8	160.3
	3049	17	63	162.5	3508.2	18.4	26.6	139.4
	3436	20	64	152.1	3588.0	19.2	40.8	137.3
	3059	18	69	144.1	3255.2	17.8	43.6	147.6
	3121	18	66	297.6	3222.8	17.8	53.8	229.8
S	2836	17	66	158.0	3006.8	17.4	62.4	171.3
	2784	15	0	87.3	2887.0	17.2	41.8	138.2
	2958	18	71	90.0	3103.0	18.2	67.0	156.9
	2994	17	64	129.8	3115.0	17.6	53.6	156.9
	2973	17	65	309.8	3065.8	17.4	50.8	149.4
	2833	15	0	105.0	2986.6	17.0	53.2	148.2
	2934	16	62	140.3	3131.0	17.8	51.2	159.4
	2999	18	67	123.9	3123.6	17.8	47.6	130.4
	2846	15	42	86.4	2859.6	15.6	9.8	80.3
	3129	18	64	131.4	3282.2	18.2	60.6	122.6

Table 20: Results on instances with 75 requests for vehicle shift length $H = 300$ and MD-5TC setting.

	Proposed ALNS: Initial Solution with transfers							
	Best of 5				Average of 5			
$\mathcal{C}$	Cost	Veh	Trs	CPU(s)	Cost	Veh	Trs	CPU(s)
L	5419	33	96	244.4	5503.8	35.2	96.0	265.4
	5332	33	96	232.7	5541.0	34.6	94.2	309.0
	5399	33	98	271.0	5615.8	35.2	96.2	268.2
	5052	33	100	275.3	5168.6	33.8	98.0	300.5
	5708	38	98	271.0	5823.8	38.0	98.0	279.4
	5613	36	96	281.5	5922.0	37.6	92.4	296.1
	5913	37	96	312.0	5989.8	38.0	96.0	281.5
	5444	35	99	394.6	5709.2	36.4	92.4	362.9
	5401	35	97	268.3	5948.8	37.0	92.6	338.2
	5199	32	100	224.9	5546.8	35.0	98.4	300.4
M	5419	33	96	244.4	5503.8	35.2	96.0	265.4
	5332	33	96	232.7	5541.0	34.6	94.2	309.0
	4514	29	94	243.0	4863.6	30.2	87.4	289.0
	4687	30	84	254.2	4821.4	30.6	87.6	259.3
	4580	29	88	318.5	4827.2	30.2	86.2	317.7
	4700	30	90	205.1	4822.0	30.6	87.0	235.4
	4469	29	89	315.4	4586.2	29.4	86.4	293.4
	4499	28	86	227.8	4638.6	29.2	86.4	300.3
	4693	30	89	285.6	4818.0	31.4	89.8	288.7
	4343	28	94	285.4	4505.6	29.2	91.2	295.4
S	3760	24	96	198.8	3918.4	24.2	49.2	283.7
	3795	24	91	335.8	3911.8	25.2	84.0	277.3
	3663	23	88	372.2	3831.4	24.0	72.8	333.0
	3813	26	87	306.5	3944.8	25.8	68.4	290.7
	3790	24	92	285.1	4003.2	24.6	45.4	244.9
	3860	24	31	202.7	3934.2	23.6	8.8	214.1
	4008	24	7	211.1	4123.2	25.0	7.6	223.2
	4022	24	13	235.7	4155.4	25.2	36.6	294.3
	3854	25	89	280.2	4134.0	25.6	62.8	343.6
	3752	24	85	374.5	3795.2	24.0	70.8	303.5

Table 21: Results on instances with 100 requests for vehicle shift length $H = 180$ and MD-5TC setting.

	Proposed ALNS: Initial Solution with transfers							
	Best of 5				Average of 5			
$\mathcal{C}$	Cost	Veh	Trs	CPU(s)	Cost	Veh	Trs	CPU(s)
L	4389	25	95	206.8	4483.8	25.4	95.8	294.1
	4316	24	99	235.7	4720.8	26.0	82.8	308.9
	4467	25	98	314.2	4585.0	26.4	96.6	261.8
	4059	23	98	335.7	4622.0	26.4	97.4	284.6
	4514	26	99	247.8	4714.4	26.6	97.6	256.6
	4664	26	92	403.7	4796.8	26.4	94.4	323.3
	4829	27	95	266.9	4905.0	27.6	94.4	275.7
	4523	25	92	293.3	4796.0	26.6	93.6	312.1
	4500	25	98	296.1	4589.4	25.6	98.0	280.4
	4545	27	98	354.5	4899.6	27.6	93.2	310.5
M	4034	24	93	506.1	4364.6	24.2	78.2	351.3
	4032	23	95	333.8	4146.6	23.8	92.8	312.9
	4085	24	97	310.6	4322.6	24.6	92.8	326.5
	4215	24	91	203.1	4559.6	25.0	66.2	269.9
	3909	23	91	199.4	4019.0	22.6	92.2	310.9
	4122	23	84	247.2	4466.8	25.0	79.8	335.9
	3882	22	97	362.2	3979.0	23.2	93.6	377.3
	3847	22	94	365.4	3995.0	23.0	93.4	326.4
	4177	23	91	188.9	4282.0	24.0	91.2	325.7
	3840	23	97	390.8	4091.2	23.0	78.6	284.0
S	3514	19	0	200.9	3544.2	19.4	14.6	209.6
	3516	19	2	232.7	3665.8	21.8	73.4	321.6
	3364	21	99	207.1	3570.8	20.0	39.8	242.3
	3561	22	88	366.0	3688.6	21.8	71.8	317.0
	3653	21	82	514.2	3723.6	20.6	43.2	382.6
	3491	21	94	569.0	3534.6	19.2	19.6	287.1
	3698	21	85	182.1	3755.4	20.6	35.6	254.8
	3516	21	95	299.9	3626.8	20.4	57.8	263.5
	3418	20	92	228.4	3567.2	21.2	83.4	312.4
	3386	20	96	179.9	3523.8	20.4	93.8	357.3

Table 22: Results on instances with 100 requests for vehicle shift length $H = 240$ and MD-5TC setting.

	Proposed ALNS: Initial Solution with transfers							
	Best of 5				Average of 5			
$\mathcal{C}$	Cost	Veh	Trs	CPU(s)	Cost	Veh	Trs	CPU(s)
L	4398	24	100	251.1	4602.2	25.4	97.8	363.0
	4405	26	99	427.3	4732.4	26.2	83.2	339.8
	4699	27	99	358.1	4802.4	26.8	97.6	259.5
	4614	25	95	398.3	5035.6	27.4	93.0	363.5
	4633	24	98	305.2	4753.8	26.0	97.6	322.1
	4806	26	96	445.4	4909.8	26.8	94.6	313.3
	4834	27	100	285.5	5058.6	27.8	94.4	309.8
	4394	24	95	301.0	4829.2	26.4	90.8	332.9
	4491	25	99	338.8	4580.4	25.6	97.8	351.1
	4525	26	99	216.8	4784.0	26.6	94.2	265.3
M	4281	24	94	611.8	4545.8	23.8	66.6	301.6
	3966	23	96	273.5	4302.6	23.2	77.6	313.1
	4326	24	96	373.5	4616.0	25.0	73.4	341.1
	4102	24	93	295.1	4321.2	25.0	90.8	341.8
	3943	22	90	485.6	4184.0	23.0	78.0	364.4
	4259	24	89	284.6	4619.4	24.6	57.8	324.9
	3948	23	95	370.1	4102.0	23.4	91.0	324.6
	3858	21	93	247.6	4231.4	23.4	76.0	302.6
	4323	25	94	567.5	4675.2	24.2	62.0	385.5
	3948	23	97	352.7	4233.8	22.6	61.8	267.2
S	3438	21	96	407.1	3536.4	20.0	58.2	317.2
	3499	19	0	227.3	3652.0	20.6	55.2	310.6
	3315	20	93	343.0	3533.0	19.6	38.8	268.1
	3626	22	87	345.8	3810.2	22.2	88.2	354.0
	3691	21	43	549.9	3763.8	20.4	8.6	276.6
	3412	21	96	352.0	3498.8	19.6	38.2	294.8
	3676	20	0	206.5	3755.4	20.6	11.0	256.7
	3616	19	0	170.4	3632.6	19.0	8.2	230.9
	3632	22	89	250.9	3737.0	21.2	55.0	230.7
	3546	20	93	318.2	3704.4	21.2	88.6	345.9

Table 23: Results on instances with 100 requests for vehicle shift length $H = 300$ and MD-5TC setting.