Received: 25 April 2021 | Accepted: 28 April 2021 | Published on: 25 May 2021

DOI: 10.1002/net.22045

SPECIAL ISSUE ARTICLE

WILEY

Crowdsourced logistics: The pickup and delivery problem with transshipments and occasional drivers

Stefan Voigt | Heinrich Kuhn

Department of Supply Chain Management & Operations, Catholic University of Eichstätt-Ingolstadt, Ingolstadt, Germany

Correspondence
Heinrich Kuhn, Department of Supply Chain Management & Operations, Catholic University of Eichstätt-Ingolstadt, Auf der Schanz 49, 85049 Ingolstadt, Germany.
Email: heinrich.kuhn@ku.de

Abstract

This article considers a setting in which a courier, express, and parcel service provider operates a fleet of vehicles with regular drivers (RDs) to ship parcels from pickup to delivery points. Additionally, the company uses a platform where occasional drivers (ODs) offer their willingness to take on requests that are on or near the route they had originally planned. There exist transshipment points (TPs) to better integrate these ODs. ODs or RDs may transfer load at these predetermined TPs. The problem is modeled as a mixed-integer programming model and called pickup and delivery problem with transshipments and occasional drivers (PDPTOD). We develop a solution approach based on an adaptive large neighborhood search. The article provides insights on how the number and location of TPs impact the cost advantages achieved by integrating ODs. It also shows that the cost savings are highly sensitive to the assumed flexibility and compensation scheme.

KEYWORDS

adaptive large neighborhood search, city logistics, crowdshipping, last-mile delivery, mixed-integer programming

1 | INTRODUCTION

The growth in online sales during recent decades combined with consumer expectations of a fast, cheap, and on-time delivery service necessitates the development of new and more cost-efficient delivery concepts. The global share of e-commerce in retail sales is predicted to more than double from 7.4% in 2015 to 15.5% in 2021 [32]. At the same time, consumer expectations are rising. According to a study conducted by McKinsey on the future of last-mile delivery, same-day delivery will grow to 25% of B2C and C2C (business-to-consumer and consumer-to-consumer) volume [18]. Nevertheless, they find that only a fraction of customers are willing to pay an appropriate premium for this kind of service. Traditional grocery retailers recognize the growing importance of online sales and have accepted the challenge of creating a fast and inexpensive delivery service for their customers by introducing crowdshipping [16, 17], among other measures.

Crowdshipping applies the concept of crowdsourcing to logistics. Participants are people who become couriers, so-called occasional drivers (ODs). ODs can be commuters or travelers who are already traveling or dedicated nonprofessional or professional drivers [25]. If commuters or travelers offer their services on a crowdshipping platform, the marginal costs, in economic and environmental terms, will be very low. Crowdshipping therefore offers a chance of reducing costs, increasing the speed of delivery and at the same time reducing environmental impact. This makes it a promising concept for mastering some of the challenges that will arise in last-mile logistics.

Crowdsourcing is a participative online activity in which an individual, institution, nonprofit organization, or company proposes to a group of individuals that they voluntarily undertake a task via a flexible, open call. The undertaking of the task should entail mutual benefit [13]. This definition of crowdsourcing implies several characteristics that apply to crowdshipping to the same extent. First, a crowdshipping platform operates in digital form and all participants need access to mobile technology. Furthermore, the undertaking of the task is voluntary and flexible, which leads to challenges for the platform provider in sustaining and guaranteeing a certain level of service quality. As ODs are not obliged to accept requests, solutions may not be robust if many ODs decline the fulfillment of requests. Mechanisms to minimize the number of declined requests are, for example, scores and reviews for ODs that exclude unreliable ODs from the platform, or bonuses that promote reliable ODs. The platform provider supposedly cuts costs, customers may benefit from increasing delivery service, and ODs may benefit from an additional source of income. In addition to monetary compensation for ODs, a variety of possible incentives exists, for example, some ODs are motivated by social or moral considerations, some just want to pass the time [8]. The crowd consists of a group of individuals and is therefore diverse. The platform needs to consider this diversity by offering a certain flexibility in respect of compensation, the number of pickups and dropoffs per trip, the time windows for service, and permitting detouring in order to reach a critical mass of ODs. If the platform wants to allow multiple pickups and dropoffs in a single trip, the routing has to be considered, which makes the problem even more challenging [2]. In addition, it is crucial to generate trust between the participants. Trust-generating mechanisms include, for example, background checks, reviews, and testimonials [12]. Another challenge, yet unsolved, is the possible exploitation of ODs. ODs have the tendency to underestimate the value of their time and the vehicle operating costs [25]. A possible solution is a platform that only allows cost-efficient matchings for both sender and courier by providing fair compensation.

We consider a setting in which a distributor (e.g., a courier, express, and parcel service provider or an omnichannel retailer) operates a fleet of vehicles with regular drivers (RDs) to ship parcels from pickup to delivery points, for example, from retailer to online customers or from one individual to another. Additionally, the distributor uses a platform where ODs offer their willingness to fulfill pickup/delivery tasks. We assume that the platform uses trust-generating mechanisms, scoring, and bonus systems to increase the reliability of ODs. ODs are therefore highly motivated to fulfill the pickup/delivery tasks that have been assigned to them. As a result, the solutions generated are stable, that is, routes for RDs and ODs are likely to be realized. In order to increase flexibility we consider the possibility of transshipments between couriers at certain locations called transshipment points (TPs), multiple tasks in a single trip for ODs, and flexible compensation. At these predetermined TPs, drivers (ODs or RDs) can hand over or take on their freight. Several planning problems arise in this context. On a strategic level the distributor, for example, has to determine the location and capacity of the TPs based on aggregated and estimated demand. On an operational planning level the distributor has to decide which tasks are assigned to RDs, which to ODs, in which sequence the drivers attend to the customers, and whether TPs are used. The present article focuses on the daily operational planning of routes and assumes that TPs are used on a flexible basis depending on demand structure. The distributor aims to minimize the overall costs arising from the distance traveled by RDs and the compensation paid to ODs. ODs receive compensation or payment depending on the additional effort required to fulfill their task(s), that is, their compensation or payment depends on the length of the detour and the number of additional stops. Furthermore, we assume a static setting where requests and offers from ODs are known in advance and are not time critical. This assumption renders the system impractical for same-day delivery, but is reasonable due to the increasing number of possible matches. As mentioned before, the platform needs to reach a critical mass to sustain a certain service level. Especially in the early phase of a platform where the number of couriers is insufficient, that is, critical mass is not reached, lesser limitations regarding time restrictions will produce more matches and hence lead to a better service level and higher utilization of ODs. The lack of time windows for ODs is admittedly a simplifying assumption, but it is nevertheless reasonable to investigate the effects of TPs and the spatial flexibility of ODs on solutions.

This article makes several contributions to the field of research. We introduce a new and challenging routing problem with regular drivers (RDs), occasional drivers (ODs), and transshipment points (TPs). We term this problem "pickup and delivery problem with transshipments and occasional drivers" (PDPTOD). The problem differs from existing modeling approaches in the following respects. (1) The PDPTOD focuses on ODs, that is, drivers who already have an origin and destination. (2) ODs have realistic spatial limitations as their flexibility is expressed by the number of stops and their willingness to make a detour. (3) RDs are employed alongside ODs. (4) TPs are integrated in the delivery process. Additionally, we develop a heuristic solution approach based on an adaptive large neighborhood search (ALNS) in a simulated annealing framework. Furthermore, we conduct extensive numerical experiments, providing insights on how the number and location of TPs impact the cost advantages achieved when ODs are integrated into the delivery process. We show that the cost advantages depend not only on the location and number of TPs but also on the flexibility of ODs, the compensation scheme, and the interaction between the aforementioned aspects. On a more practical note, the model can be used for platforms that have not reached a critical mass of ODs and therefore cannot guarantee to serve all customers with ODs only.

The remainder of the article is organized as follows. Section 2 describes the problem in detail. Section 3 discusses the related literature. Section 4 formulates the associated mixed-integer programming (MIP) model. Section 5 presents the solution

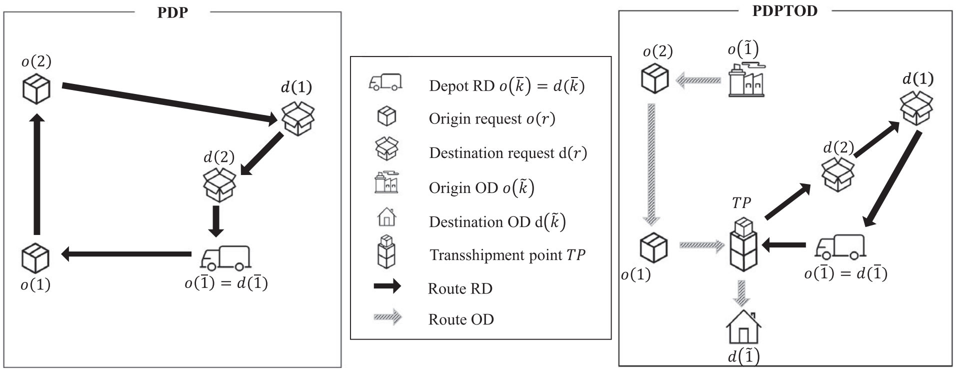


FIGURE 1 Illustrative example of PDP versus PDPTOD solution. OD, occasional driver; PDP, pickup and delivery problem; PDPTOD, pickup and delivery problem with transshipments and occasional drivers; RD, regular driver

approach suggested that is used in Section 6 for numerical experiments. Section 7 summarizes the fundamental findings of the article and provides directions for future research.

2 | PROBLEM DESCRIPTION

We focus on a local distributor (e.g., a courier, express, and parcel [CEP] service provider or an omnichannel retailer) operating its own fleet of vehicles with regular drivers (RDs) $\bar{k} \in \bar{K}$ to ship requests $r \in R$ from pickup $o(r)$ to delivery points $d(r)$, for example, from retailer to online customers or from one individual to another. Every request has to be satisfied. The RDs can execute pickup and delivery tasks when there are not sufficient occasional drivers (ODs) available or if it is cheaper to use RDs, thus enabling the distributor to guarantee a certain service level. RDs start and end at the same depot and cause costs depending on the distance traveled. Besides RDs, the company uses an online crowdsourcing platform to obtain ODs for pickup and delivery tasks. Every OD $\tilde{k} \in \tilde{K}$ starts at its origin $o(\tilde{k})$ and end at its destination $d(\tilde{k})$. The distributor knows the pickup and delivery tasks in advance, for example, one day before the start of the tour. The same holds for ODs, that is, ODs express their willingness to participate the day before. This assumption is reasonable for well-known trips in advance or regular trips, for example, daily trips to work and back home. The platform mainly aims at commuters where the origin and destination are the same every working day. Professional drivers with spare capacity and fixed routes can also participate on the platform. ODs specify the maximum number of stops $\alpha_{\tilde{k}}$, maximum length of detours $\beta_{\tilde{k}}$, and the origins $o(\tilde{k})$ as well as the destinations $d(\tilde{k})$ of the respective trips originally planned. This gives ODs a reasonable amount of control over their tasks while not overtaxing them with too much information. In addition we assume that ODs have sufficient temporal flexibility, that is, ODs are not obliged to indicate time windows, as we assume that all requests can be fulfilled in a reasonable amount of time. Of course, this is a simplifying assumption and may be critical if ODs have no temporal flexibility. However, hard time windows would decrease the number of possible matchings, which is especially unfavorable for platforms in their early stages. ODs and RDs are not obliged to deliver parcels directly, they can transfer parcels at predetermined transshipment points (TPs) $t \in T$ to another OD or RD. A TP can be any suitable location with sufficient capacity, for example, shops or gas stations with a backroom, large parcel lockers, or intermediate depots. We assume that these facilities have a large enough capacity to handle all the parcels that will potentially need to be processed. As parcels may be stored at TPs for a limited time, drivers do not need to wait to hand over parcels to the following driver. The objective from the distributor's viewpoint is to minimize total costs while serving all customers. The costs are composed of routing costs for RDs c_{ij} and costs for ODs that are derived from fixed cost per stop and variable cost depending on the length of a detour compared with the length of their route originally planned $c_{o(\tilde{k})d(\tilde{k})}$. Note that if there are no ODs and no TPs the problem is reduced to the standard pickup and delivery problem (PDP) that typically arise in local area CEP services.

Figure 1 shows an illustrative example with two requests $r = 1$ and $r = 2$. Without ODs, the requests have to be served by one RD. The RD ($\bar{k} = \bar{1}$) starts at the depot $o(\bar{1})$, picks up the requests at $o(1)$ and $o(2)$, delivers the first request 1 to $d(1)$ and then request 2 to $d(2)$. Finally, the RD returns to the depot $d(\bar{1})$. The route of the RD is depicted with black arrows on the left of Figure 1. The solution changes if ODs and TPs are used as displayed on the right of Figure 1. An OD $\tilde{k} = \tilde{1}$ starts at her/his origin $o(\tilde{1})$ (e.g., her/his workplace), picks up both requests, drops off the load at the transshipment point TP, and reaches her/his destination $d(\tilde{1})$ (gray route). The RD starts at the depot $o(\bar{1})$, picks up the dropped load from $\tilde{1}$ at TP, delivers request 2, then request 1, and returns to the depot $d(\bar{1})$.

TABLE 1 Literature overview of crowdshipping approaches

Author	RDs[a]	R[b]	All[c]	TP[d]	C[e]	S[f]
Soto Setzke et al. [35]	n	1	n	n	v	s
Wang et al. [36]	n	≥1	y	n	v	s
Archetti et al. [1]	y	1	y	n	f	s
Macrina et al. [23]	y	≥1	y	n	f	s
Dayarian and Savelsbergh [7]	y	1	y	n	–	d
Arslan et al. [2]	y	≥1	y	n	f, v	d
Kafle et al. [20]	y	≥1	y	y	Bids	s
Chen et al. [5]	(y)	≥1	y	y	f, v	s
Raviv and Tenzer [29]	n	≥1	y	y	v	d
Sampaio Oliveira et al. [33]	n	≥1	y	y	v	s
Own Approach	y	≥1	y	y	f, v	s

[a]Regular drivers: y/n.
[b]Number of requests one OD can serve on a single trip: 1 or ≥1.
[c]All requests have to be served: y/n.
[d]Transshipment points: y/n.
[e]Compensation: Fixed per request (f), variable depending on detour (v), bids, or none (–).
[f]Setting: static (s) or dynamic (d).

3 | RELATED LITERATURE

The present section reviews relevant literature. First, Subsection 3.1 briefly presents literature that discusses general aspects of crowdshipping. Subsection 3.2 then gives an overview on modeling approaches that are closely related to our problem setting. Afterward, Subsection 3.3 reviews exact and heuristic solution approaches for the pickup and delivery problem (PDP) since our modeling and solution approach are based on these approaches. Finally, Subsection 3.4 summarizes the findings from literature and identifies the specific contribution of our work.

3.1 | Crowdshipping: Definition, challenges, and opportunities

Many authors deal with fundamental questions of how crowdshipping can be used in practice, what challenges and opportunities arise, or how to define crowdshipping. Sampaio Oliveira et al. [32] and Buldeo Rai et al. [4] consider crowdshipping as an opportunity for city logistics, especially for the last mile. Doerrzapf et al. [9] see crowdshipping as a chance to strengthen local shops in the competition with online retailers. McKinnon [25] stresses the possible positive environmental impact by cutting down urban traffic levels and therefore emissions. By contrast, Qi et al. [27] find that crowdshipping may increase emissions because occasional drivers (ODs) routes are prolonged and that most cost savings are due to the reducing of the fleet of regular drivers (RDs) rather than cutting operating costs directly. Le et al. [21] present a review of current practice, research, and case studies from a demand, supply, and operations viewpoint, coming to the conclusion that a functioning crowdshipping system has to be fairly complex and needs features such as sophisticated pricing and matching procedures. Similarly, Rouges and Montreuil [31] suggest using automated matching algorithms to increase the efficiency of crowdshipping platforms.

3.2 | Crowdshipping problems

This subsection reviews publications that are directly related to our problem setting as all of them consider ODs within the delivery process of parcels and goods. These papers present modeling and solution approaches that match delivery requests and offers from ODs in a crowdshipping setting. Table 1 lists and classifies these approaches. Approximately half of these approaches consider transshipment points (TPs). Note that we omit literature related to other shared mobility concepts. Interested readers are referred to, for example, Mourad et al. [26].

3.2.1 | Crowdshipping without transshipments

The following publications consider ODs fulfilling delivery requests but omit the option of using TPs. Soto Setzke et al. [35] propose a matching algorithm based on routes and time feasibility modeled as min-cost max-flow model. Wang et al. [36] provide a similar approach with the difference that all parcels have to be delivered and each OD is allowed to make multiple deliveries. Both models are suitable for large-scale data sets, but as a drawback require highly simplifying assumptions and lack the ability to integrate RDs in order to guarantee a certain service level. Archetti et al. [1] develop the vehicle routing problem with occasional drivers (VRPOD) based on the classical capacitated vehicle routing problem (VRP), and propose a multistart heuristic for the solution. In contrast to both modeling approaches mentioned previously, the VRPOD uses one RD as a backup option if no suitable OD is found. The OD can handle only one request per trip and in turn receives compensation depending on the distance between depot and customer, independent of the detour. The contribution of Archetti et al. [1] shows the potential benefits of employing ODs and the importance of choosing an appropriate compensation scheme for ODs. Macrina et al. [23] extend the VRPOD by allowing multiple deliveries, split deliveries, and integrating time windows. Allowing multiple deliveries is one promising possibility to reduce the need for RDs and increase the use of ODs instead. Dayarian and Savelsbergh [7] also set up on the VRPOD. They assume that demand and capacity, that is, the number of ODs, changes over time. ODs are in-store customers willing to deliver at most one online order. They assume in contrast to Archetti et al. [1] that ODs are always cheaper than RDs because their compensation is in the form of store credits rather than actual payments. They point out that a higher number of ODs with higher flexibility has the potential to decrease costs and at the same time improve service quality. Archetti et al. [2] consider a peer-to-peer platform that uses foremost ODs and RDs only as a backup option, similar to Dayarian and Savelsbergh [7]. They assume that ODs can deliver more than one request per job. This means routing is necessary. They show that the use of ODs reduces costs and system-wide distance traveled.

3.2.2 | Crowdshipping with transshipments

All publications mentioned so far come without the possibility of transferring load at TPs and make simplifying assumptions in some cases, such as only one request per OD trip. The following publications are more closely related to our work as they integrate TPs and allow multiple deliveries per OD. Kafle et al. [20] suggest using cyclists and pedestrians as ODs who are close to customers. In the first step, ODs submit bids to a truck carrier and relay parcels at TPs if they win the bid. The ODs can take on several requests and have to make sure that they are able to fulfill these requests on time and therefore organize their routes independently. This assumption leads to high transactional costs for ODs, reducing the attractiveness of offering their services on the platform. The approach reduces the distance the truck travels and cuts costs compared with pure truck delivery. Chen et al. [5] develop the multihop driver-parcel matching problem. This problem is characterized by the possibility of transferring parcels multiple times between ODs. They assume that there is a shipping company with RDs that takes on tasks that cannot be served by ODs at additional cost. However, compared with our setting, they do not consider the routing of these RDs. Raviv and Tenzer [29] develop an innovative approach based on a connected network of automatic service points similar to the approach of Chen et al. [5]. Their work however considers the compensation scheme of ODs and incorporates service-level considerations, that is, parcels that are in the system longer are prioritized. ODs may pick up, drop off, or transfer parcels at service points (SPs) that are along their original route. The ODs are compensated for stopping by these SPs. Similar to our work, ODs are compensated per stop and are allowed to make multiple stops at SPs, as long as they are on their route. In contrast to our work, there are no RDs as backup option, and the routes of ODs are predetermined.

A very closely related problem to our problem setting is developed by Sampaio Oliveira et al. [33]. Similar to our approach, they model the problem as a pickup and delivery problem (PDP) with TPs based on the model of Rais et al. [28], and develop an ALNS to solve it with reasonable computation times. Our problem setting is different in the following respects. Sampaio Oliveira et al. [33] assume that crowdshippers work for a given period of time and are paid on an hourly basis. Note that we particularly use the term "occasional drivers" to distinguish these types of drivers from crowdshippers who work part-time and would otherwise not be traveling. ODs, in contrast, are traveling anyway. In addition, the authors do not incorporate RDs as alternative option to serve requests. The crowdshippers are only constrained by the time window in which they want to work. Their model is therefore fitting for settings in which the crowdshipper works part-time for the service provider. This is the case for companies such as Amazon Flex or Uber-Freight, where crowdshippers indicate the period they are willing to work, but are not limited in any other way. We focus on a courier express and parcel provider that uses fairly constrained ODs (instead of part-time working crowdshippers) as an option, together with RDs. These RDs guarantee to serve all requests, as long as their capacity suffices.

3.3 | Pickup and delivery problem

Pickup and delivery problems (PDPs) describe a class of VRPs, in which vehicles not only deliver goods, but also pick up goods. Some PDPs also permit transshipments within the delivery and/or pickup tours. We briefly review the available modeling and solution approaches for PDP with and without transshipments since the PDPTOD extends these problem classes.

A promising solution approach for the PDP with time windows (PDPTW) is the ALNS developed by Ropke and Pisinger [30]. They applied the heuristic to more than 350 benchmark instances and were able to provide new best solutions for more than 50% of them. More recently, the Lin–Kernighan–Helsgaun (LKH) heuristic that is well known for its good performance on the traveling salesman problem [14] is extended to deal with many variants of VRPs by introducing penalties for constraints [15]. The LKH is able to further improve best-known solutions of instances from the literature.

The literature on PDP with transshipments (PDPT) is rather limited. Drexl [10, 11] presents some practical applications where transshipments are involved. Cortés et al. [6] give a new strict arc-based formulation for the PDPT and present a branch-and-cut solution approach that decreases computation time by approximately 90% compared with a standard branch-and-bound solver. Nevertheless, the instances for which they obtained solutions are remarkably small with six requests, two vehicles and just one TP. Rais et al. [28] present a mixed-integer formulation for the PDPT with a polynomial bounded number of constraints and variables. The MIP uses two distinct flows, that is, (1) the vehicle flow and (2) the request flow, which are matched by linking constraints. The proposed MIP formulation can be extended by simple modifications in order to represent several variants of the problem. The authors prove this capability by showing how to integrate time windows, split deliveries, heterogeneous vehicles with a flexible fleet size and flexible vehicle stops, that is, a vehicle may end its route at a node differing from its original depot. However, their model does not cover the integration of constrained ODs, which can be used alongside RDs. Numerical results are based on small instances with up to 14 nodes and are therefore rather limited, as in Cortés et al. [6].

The publications attempting to solve the PDPT exactly show that exact approaches are only able to solve small instances, so heuristic solution approaches are necessary. A promising heuristic solution approach for the PDPT developed by Masson et al. [24] is an extension of the ALNS for the PDP by Ropke and Pisinger [30]. The publication of Masson et al. [24] is motivated by an application where people with disabilities require transport, for example, from their home to schools. Similar to our work, transfers are possible at designated transfer points, but only RDs starting from a set of depots are considered. They first implement an ALNS for the PDPTW without transfers based on four destroy and two repair operators. This method is then extended by three destroy operators, two repair operators, and heuristics to choose appropriate transfer points to cope with the PDPT. The ALNS is tested on instances from the literature and also on a case study concerning the transport of disabled persons with a maximum of 193 requests and a maximum of 24 transfer points. The authors are able to show that the introduction of transshipments has the potential to reduce costs. However, the cost advantages greatly depend on the geographical characteristics of the instances.

3.4 | Summary

Many publications focus on crowdshipping in general, but few papers deal with optimization models for matching ODs and demand. Early papers make highly simplifying assumptions, for example, one request per trip [35, 36]. Archetti et al. [1] were the first to model the problem as VRP, but still allow only one request per OD trip. The following publications extend the ideas of Archetti et al. [1] by allowing multiple deliveries, split deliveries, and imposing time windows [23], covering the stochastic and dynamic aspects of the problem [7] or turning the problem into a PDP [2]. These publications however neglect the possibility of using TPs to increase the flexibility of the system. Kafle et al. [20] use TPs, but do not cover the routing aspect of the ODs. By contrast, Chen et al. [5] neglect the routing of RDs. The setting of Sampaio Oliveira et al. [33] is the most related to our work, but focuses on crowdshippers instead of ODs, as previously explained.

To sum up, the problem setting considered in our article has not yet been addressed in the literature. We consider ODs who can serve multiple requests, alongside RDs and both type of drivers have the possibility to transfer loads at TPs. The problem enriches the literature on PDPs and crowdshipping by presenting a PDP variant with ODs and TPs. In the course of this we focus on drivers who already have an origin and destination, termed ODs. These ODs have realistic spatial limitations. Additionally, RDs are employed alongside ODs and TPs are integrated into the delivery process to increase the use of ODs and further reduce costs. The combination of these aspects is unique in literature. From the methodological viewpoint, we present a MIP formulation and an extension of the ALNS, which underlines the wide applicability of the ALNS.

4 | THE PICKUP AND DELIVERY PROBLEM WITH TRANSSHIPMENTS AND OCCASIONAL DRIVERS

The pickup and delivery problem with transshipments and occasional drivers (PDPTOD) is defined on a directed graph $G(N, A)$ with node set N and arc set A. The node set N consists of the origins $o(\tilde{k})$ and destinations $d(\tilde{k})$ of occasional drivers (ODs), $\tilde{k} \in \tilde{K}$, the starting $o(\bar{k})$ and ending depots $d(\bar{k})$ of regular drivers (RDs), $\bar{k} \in \bar{K}$, the locations of pickup $o(r)$ and delivery requests $d(r)$, and transshipment points (TPs) $t \in T$. For $i, j \in N$, we denote the arc from i to j as $(i, j) \in A$ and the associated costs using c_{ij}. The arc set consists only of reasonable arcs, for example, there are no outgoing arcs from a destination depot.

TABLE 2 Notation used to formulate model PDPTOD

Sets	
K	Set of vehicles, $K = \overline{K} \cup \tilde{K}$, $K = \{1, \dots, k, \dots, \lvert\overline{K}\rvert + \lvert\tilde{K}\rvert\}$
$\overline{K}$	Set of regular drivers, start at origin $o(\overline{k})$ and end at destination $d(\overline{k})$, $\overline{K} = \{1, \dots, \overline{k}, \dots, \lvert\overline{K}\rvert\}$
$\tilde{K}$	Set of occasional drivers, start at origin $o(\tilde{k})$ and end at destination $d(\tilde{k})$, $\tilde{K} = \{\lvert\overline{K}\rvert + 1, \dots, \tilde{k}, \dots, \lvert\overline{K}\rvert + \lvert\tilde{K}\rvert\}$
R	Set of requests with origin $o(r)$ and destination $d(r)$, $R = \{1, \dots, r, \dots, \lvert R\rvert\}$
T	Set of transshipment points, $T = \{1, \dots, t, \dots, \lvert T\rvert\}$
Parameters	
$\alpha_{\tilde{k}}$	Maximum number of stops, $\tilde{k} \in \tilde{K}$
$\beta_{\tilde{k}}$	Factor limiting the length of tour, $\tilde{k} \in \tilde{K}$
$\gamma_{\overline{k}}$	Maximum duration of tour, $\overline{k} \in \overline{K}$
c_{ij}	Traveling costs from node i to j, $(i,j) \in A$
M	Sufficiently large number
p^{detour}	Compensation depending on length of detour for ODs
p^{stop}	Compensation per stop for ODs
τ_{ij}	Traveling time from node i to j, $(i,j) \in A$
Decision variables	
s_{jrkl}	Binary variable indicating whether request r is transferred from vehicle k to vehicle l at node j, $s_{jrkl} = 1$ or not $s_{jrkl} = 0$, $j \in N$, $r \in R$, $k, l \in K$, $l \neq k$
t^{arr}_{jk}	Time of arrival of vehicle k at node j, $j \in N$ and $k \in K$
t^{dep}_{jk}	Time of departure of vehicle k at node j, $j \in N$ and $k \in K$
x_{ijk}	Binary variable indicating whether vehicle k uses arc (i,j), $x_{ijk} = 1$ or not $x_{ijk} = 0$, $(i,j) \in A$, $k \in K$
y_{ijkr}	Binary variable indicating whether vehicle k carries request r on arc (i,j), $y_{ijkr} = 1$ or not $y_{ijkr} = 0$, $(i,j) \in A$, $k \in K$, $r \in R$

Abbreviations: OD, occasional driver; PDPTOD, pickup and delivery problem with transshipments and occasional drivers.

The model PDPTOD is based on the PDPT model suggested by Rais et al. [28]. Table 2 summarizes the sets, parameters, and decision variables defined. The PDPTOD is modeled afterward.

Model PDPTOD

$$\text{Minimize} \sum_{k\in\overline{K}}\sum_{(i,j)\in A} c_{ij}x_{ijk} + \sum_{k\in\tilde{K}}\sum_{(i,j)\in A}(p^{\text{detour}}c_{ij} + p^{\text{stop}})x_{ijk} - \sum_{k\in\tilde{K}}\sum_{(o(k),j)\in A}(p^{\text{detour}}c_{o(k)d(k)} + p^{\text{stop}})x_{o(k)jk} \tag{1}$$

s.t.

$$\sum_{(i,j)\in A} x_{ijk} \leq 1 \qquad \forall k \in K,\ i = o(k), \tag{2}$$

$$\sum_{(i,j)\in A} x_{ijk} = \sum_{(j,l)\in A} x_{jlk} \qquad \forall k \in K,\ i = o(k),\ l = d(k), \tag{3}$$

$$\sum_{(i,j)\in A} x_{ijk} - \sum_{(j,i)\in A} x_{jik} = 0 \qquad \forall k \in K,\ \forall i \in N \setminus \{o(k), d(k)\}, \tag{4}$$

$$t^{\text{dep}}_{ik} + \tau_{ij} - t^{\text{arr}}_{jk} \leq M(1 - x_{ijk}) \qquad \forall (i,j) \in A,\ \forall k \in K, \tag{5}$$

$$t^{\text{arr}}_{jk} \leq t^{\text{dep}}_{jk} \qquad \forall j \in N,\ \forall k \in K, \tag{6}$$

$$y_{ijkr} \leq x_{ijk} \qquad \forall (i,j) \in A, \forall k \in K,\ \forall r \in R, \tag{7}$$

$$\sum_{k\in K}\sum_{(i,j)\in A} y_{ijkr} = 1 \qquad \forall r \in R,\ i = o(r), \tag{8}$$

$$\sum_{k\in K}\sum_{(i,j)\in A} y_{ijkr} = 1 \qquad \forall r \in R,\ i = d(r), \tag{9}$$

$$\sum_{k\in K}\sum_{(i,j)\in A} y_{ijkr} - \sum_{k\in K}\sum_{(j,i)\in A} y_{jikr} = 0 \qquad \forall r \in R,\ \forall i \in T, \tag{10}$$

$$\sum_{(i,j)\in A} y_{ijkr} - \sum_{(j,i)\in A} y_{jikr} = 0 \qquad \forall k \in K,\ \forall r \in R,\ \forall i \in N \setminus (T \cup \{o(r), d(r)\}), \tag{11}$$

$$\sum_{(j,i)\in A} y_{jikr} + \sum_{(i,j)\in A} y_{ijlr} \leq s_{jrkl} + 1 \qquad \forall r \in R,\ \forall i \in T,\ \forall k \in K,\ \forall l \in K,\ k \neq l, \tag{12}$$

$$t_{jk}^{\mathrm{arr}} - t_{jl}^{\mathrm{dep}} \leq M(1 - s_{jrkl}) \qquad \forall r \in R,\ \forall j \in T,\ \forall k \in K,\ \forall l \in K,\ k \neq l, \tag{13}$$

$$\sum_{(i,j)\in A} x_{ij\tilde{k}} \leq \alpha_{\tilde{k}} \qquad \forall \tilde{k} \in \tilde{K}, \tag{14}$$

$$\sum_{(i,j)\in A} c_{ij} x_{ij\tilde{k}} \leq \beta_{\tilde{k}} c_{o(\tilde{k})d(\tilde{k})} \qquad \forall \tilde{k} \in \tilde{K}, \tag{15}$$

$$\sum_{(i,j)\in A} \tau_{ij} x_{ij\overline{k}} \leq \gamma_{\overline{k}} \qquad \forall \overline{k} \in \overline{K}, \tag{16}$$

$$x_{ijk} \in \{0,1\} \qquad \forall (i,j) \in A,\ \forall k \in K, \tag{17}$$

$$y_{ijkr} \in \{0,1\} \qquad \forall (i,j) \in A,\ \forall k \in K,\ \forall r \in R, \tag{18}$$

$$s_{jrkl} \in \{0,1\} \qquad \forall j \in N,\ \forall r \in R,\ \forall k \in K,\ \forall l \in K, \tag{19}$$

$$t_{jk}^{\mathrm{dep}} \in \mathbb{R}_0^+ \qquad \forall j \in N,\ \forall k \in K, \tag{20}$$

$$t_{jk}^{\mathrm{arr}} \in \mathbb{R}_0^+ \qquad \forall j \in N,\ \forall k \in K. \tag{21}$$

The objective function (1) quantifies the costs incurred in fulfilling all requests. The costs consist of three terms. The first term quantifies the costs incurred by RDs. The second term quantifies the detour costs and the compensation per stop for ODs. The third term reduces the costs by the length of the route of the OD originally planned and by one stop, because the ODs are only compensated according to the length of their respective detour and the number of additional stops. Therefore, the last two terms reflect the overall compensation for ODs. Note that ODs can go directly to their destination without affecting the objective, because the second and third term cancel each other out.

Constraints (2)–(6) model the vehicle flow, whereas Constraints (8)–(13) model the request flow, and Constraints (7) connect both flows. The following passage describes these constraints in detail. Constraints (2) and (3) ensure that each vehicle starts at most one route from its origin and ends at its destination depot, if it has left its origin depot. Constraints (4) conserve vehicle flows. If a vehicle k uses arc (i, j), Constraints (5) ensure that the arrival time at node j is greater than the departure time at node i plus the traveling time τ_{ij} from node i to j. Constraints (6) ensure that vehicle k does not depart at node j before it has arrived at the same node. Constraints (5) and (6) prevent subtours. Constraints (7) guarantee the connection of the vehicle flow and the request flow. Constraints (8) and (9) ensure that requests are picked up and delivered. Constraints (10) conserve request flow at TPs where requests have to be transported by any one vehicle. Constraints (11) conserve request flow at every other node. In contrast to TPs, the same vehicle bringing the request to that node has to leave with that exact same request. Constraints (12) enforce s_{jrkl} to 1 if there is a transfer between vehicle k and l. Constraints (13) ensure that the vehicle k transferring load to vehicle l arrives before vehicle l departs. Constraints (14) restrict the number of stops of ODs. Constraints (15) limit the length of the detour of ODs, Constraints (16) the duration of the RDs routes. Constraints (17), (18), and (19) define the binary decision variables. Constraints (20) and (21) define the domains of the continuous variables.

The model PDPTOD contains the traveling salesman problem as a special case that is known to be $\mathcal{NP}$-hard. Therefore, the PDPTOD is $\mathcal{NP}$-hard as well. The MIP can only be solved for small instances with a commercial MIP solver, as will be shown in Section 6. Consequently, we suggest developing heuristic solution approaches.

5 | SOLUTION APPROACH

We propose an adaptive large neighborhood search (ALNS) embedded in a simulated annealing framework for the solution of the PDPTOD. As mentioned before, the ALNS has been applied to many variants of the VRP, including the PDPTW [30] and the PDPT [24], which is closely related to our problem. Compared with the ALNS of Masson et al. [24], we do not use operators that directly remove transshipment points (TPs) and therefore requests routed through this TP. Instead, TPs are only removed from a solution if no request is served via this TP. Our ALNS uses no additional heuristic to select a subset of TPs during insertion. The two insertion operators working with TPs may evaluate either all TPs or just one TP at random. Furthermore, operators are adapted to capture and utilize the characteristics of occasional drivers (ODs). The proposed ALNS differs from existing approaches in three further major respects: (1) we suggest using combined operators, that is, different operators could be used during one iteration, (2) the number of requests is sampled from a binomial and not from a uniform distribution, and (3) the initial temperature of the simulated annealing approach is obtained via a sampling procedure. In the following section, we give a high-level introduction to the ALNS scheme. Next, we describe the destroy and repair operators that are specially designed for the PDPTOD in detail.

```
1  CurrentSolution ← InitialSolution
2  BestSolution ← CurrentSolution
3  Temp ← GetInitialTemp()
4  while Temp > TempLow do
5      Choose Repair and Destroy Operator
6      Solution ← Repair(Destroy(CurrentSolution))
7      if f(Solution) < f(BestSolution) then
8          CurrentSolution ← Solution
9          BestSolution ← CurrentSolution
10     else if accept(f(Solution), f(BestSolution), Temp) then
11         CurrentSolution ← Solution
12     end
13     Update Operator Weights
14     Temp ← CoolRate · Temp
15 end
```

FIGURE 2 Adaptive large neighborhood search algorithm in simulated annealing framework

5.1 | Adaptive large neighborhood search

Figure 2 depicts the general scheme of the ALNS proposed. The ALNS extends the large neighborhood search by choosing each destroy and repair operator depending on its past performance during the search. The initial solution required is obtained by randomly applying repair operators until every request is served, that is, until the solution is complete and feasible. We use a simulated annealing acceptance criterion to diversify the search. The probability of accepting a deteriorating solution depends on the difference in costs of the candidate solution $f(Solution)$ and the cost of the best solution obtained so far $f(BestSolution)$, as well as the current temperature *Temp*. The temperature *Temp* is used as stopping criterion (line 4 of Figure 2) and is determined for every instance with $Temp = -\frac{\overline{\Delta E}}{\ln(\chi_0)}$ using the formula from [19] cited in [3], where $\overline{\Delta E}$ is an estimation of the cost increase of strictly positive transitions and χ_0 a parameter expressing the probability of accepting a deteriorating solution. We execute n_0 iterations of the ALNS in order to generate the transitions. The temperature is reduced by *CoolRate* after each iteration (line 14). As long as the temperature is above the minimum temperature, *TempLow* another iteration is executed. At the end of every iteration, the weights are updated according to the performance of the operator. The score of an operator is increased by σ_1 if the operator was involved in producing a new global best solution, σ_2, if a new solution is found that has lower costs than the current solution, or σ_3, if the new solution has higher costs but is accepted. The probability of choosing an operator depends on the scores obtained during the search and is calculated in the same way as in Ropke and Pisinger [30].

In the following Subsections 5.2 and 5.3, we give a brief overview of the operators implemented. The first three destroy operators as well as the first three repair operators are adapted from Ropke and Pisinger [30] to match our problem setting. If there are no TPs and no ODs, the problem is reduced to a classical PDP and can be solved while using only the first three destroy and repair operators. If there are additionally ODs available, but no TPs, the scenario is denoted as pickup and delivery problem with occasional drivers (PDPOD) and can be solved without both insertion operators for TPs. Table 3 presents an overview of the operators used for solving the three models, that is, PDP, PDPOD, and PDPTOD.

5.2 | Destroy operators

We use five destroy operators that remove q requests (pickup and delivery nodes) from the current solution. The number of requests q to be removed is determined in every iteration and sampled from a binomial distribution with sample size $|R|$ and probability p^{binom}. The idea is to introduce a bias for lower q, but still maintain the possibility of removing a high number of requests. Uniformly distributed q usually focuses on a narrow range and therefore limits the diversification.

Random removal

The random removal operator is the simplest and fastest one. It randomly removes q requests from the solution. If the pickup and delivery of a request are served by two vehicles (i.e., a TP was used), it will be checked whether the TP is used by other requests on the same two routes. If the TP is not used any more, it will be removed from both routes. The same check is implemented for every other destroy operator.

TABLE 3 Overview of operators used

Operator	ALNS-PDP	ALNS-PDPOD	ALNS-PDPTOD
Random removal	✓	✓	✓
Worst removal	✓	✓	✓
Shaw removal	✓	✓	✓
Least efficient removal		✓	✓
Combined removal	✓	✓	✓
Random insertion	✓	✓	✓
Best insertion	✓	✓	✓
Regret-3 insertion	✓	✓	✓
Best-with-random TP insertion			✓
Best-with-best TP insertion			✓
Combined insertion	✓	✓	✓

Abbreviations: ALNS, adaptive large neighborhood search; PDP, pickup and delivery problem; PDPTOD, pickup and delivery problem with transshipments and occasional drivers; TP, transshipment point.

Worst removal

The worst removal operator iterates across all routes of ODs and regular drivers (RDs) to find the request that increases the cost the most and removes this request.

Shaw removal

The shaw removal operator removes requests that are similar to each other. The similarity $Rel(r_1, r_2)$ is measured by the distance D of pickup $o(r_1)$ to $o(r_2)$ and D of delivery $d(r_1)$ to $d(r_2)$ and the number of common ODs that are able to serve both requests. The reasoning behind this is that these requests can potentially be handled by the same OD, leading to reduced detours. The lower the $Rel(r_1, r_2)$, the more related are requests r_1 and r_2.

$$Rel(r_1, r_2) = D_{o(r_1),o(r_2)} + D_{d(r_1),d(r_2)} + \left(1 - \frac{|\tilde{K}_{r_1} \cap \tilde{K}_{r_2}|}{\min(|\tilde{K}_{r_1}|, |\tilde{K}_{r_2}|)}\right). \tag{22}$$

The shaw removal operator chooses the first request to be removed with the random removal operator. The following requests are chosen from a sorted array of all remaining requests, whereas the array R^{Re} is sorted by increasing $Rel(r_1, r_2)$. $\mathrm{uniform}(0, 1)^{p^{\mathrm{Shaw}}} |R^{\mathrm{Re}}|$ determines the index of the request to be removed from the array. $p^{\mathrm{Shaw}} \geq 1$ introduces randomness in selecting the requests.

Least efficient removal

It is only worthwhile using RDs and ODs when they fulfill requests in an efficient way, that is, without long detours. The least efficient removal operator removes a request from the route with the lowest efficiency. The cost per request, $cost_r$ is used as proxy for efficiency. It measures the relative cost by dividing the total cost, $cost_k$ of a route k by the number of requests $|R_k|$ served via this specific route. R_k defines the requests served via route k. The higher the cost per request, the lower the efficiency.

$$cost_r = \frac{cost_k}{|R_k|} \quad \forall r \in R_k, k \in K. \tag{23}$$

Combined removal

The combined removal operator removes a request by randomly applying one of the aforementioned operators. The probability depends on the operator weights and is selected by a roulette wheel selection approach. The operator is chosen for every request and therefore may be different for requests in the same iteration. The combined removal operator fosters diversification.

5.3 | Repair operators

Repair operators insert requests that have previously been removed into the solution. We use six operators to build a new solution, three of which are relatively standard and do not use TPs. The fourth and fifth operators ensure the use of TPs. In some cases these two operators are not able to insert a request while using a TP because, for example, no OD is able to serve the request without exceeding the maximum detour they are willing to make. If the request cannot be inserted using a TP the request is inserted via the best insertion operator.

```
Require: Request r with pickup o(r) and delivery d(r)
Choose random t
for pickup route ∈ potential pickup routes do
    CostPickupRoute ← f(pickuproute)
    if t not in pickup route then
        insert t in pickup route at best position
    end
    if t in pickup route then
        insert o(r) in pickup route at best position before t
    end
    if o(r) in pickup route then
        for delivery route ∈ potential delivery routes do
            CostDeliveryRoute ← f(deliveryroute)
            if pickup route ≠ delivery route then
                if t not in delivery route then
                    insert t in delivery route at best position
                end
                if t in delivery route then
                    insert d(r) in delivery route at best position after t
                    if d(r) in delivery route then
                        Cost ← f(pickup route) + f(delivery route) − CostPickupRoute − CostDeliveryRoute
                    end
                end
            end
        end
    end
end
Determine (pickup route) – (TP) – (delivery route) combination with lowest Cost
Insert request accordingly
```

FIGURE 3 Best-with-random TP insertion operator. TP, transshipment point

Random insertion

The random insertion operator inserts a request randomly into the current solution without using a TP. This means the pickup and delivery of one request has to be served by exactly one vehicle. The operator may use both types of drivers, RDs or ODs. The route in which the request is to be inserted is chosen randomly from the set of possible drivers. The set of possible drivers contains only RDs and ODs for which the request is in reach, that is, pickup and delivery can be served without violating the detour constraint. This set is calculated once and reduces the runtime that would otherwise be wasted on fruitless attempts assigning requests to drivers who are for sure unable to serve these requests. This set becomes considerably smaller as β_k decreases.

Best insertion

Instead of inserting the request randomly, the best insertion operator determines which route and which position increase the costs least. The operator iterates across the set of candidate RDs and ODs and chooses the driver who can accommodate the request with the least additional effort.

Regret-3 insertion

The operator shows some foresight. It calculates the cost difference between the best versus second best and second best versus third best route in which the request can be inserted. The operator iterates across all requests to be inserted and inserts the request with the highest regret value. Before calculating the actual regret values, requests that can be served by fewer than three routes because of the detour constraint are inserted in order of ascending number of potential routes. This means requests with only one potential route are inserted first followed by requests with two potential routes, and so on.

TABLE 4 Regions and locations of transshipment points (TPs) in scenario T4

Region	Location
North	$[\min(x_r) + 0.5 \cdot \text{range}(x_r), \min(y_r) + 0.75 \cdot \text{range}(y_r)]$
East	$[\min(x_r) + 0.75 \cdot \text{range}(x_r), \min(y_r) + 0.5 \cdot \text{range}(y_r)]$
South	$[\min(x_r) + 0.5 \cdot \text{range}(x_r), \min(y_r) + 0.25 \cdot \text{range}(y_r)]$
West	$[\min(x_r) + 0.25 \cdot \text{range}(x_r), \min(y_r) + 0.5 \cdot \text{range}(y_r)]$

Best-with-random TP insertion

This operator enforces the use of a random TP when inserting the request. Figure 3 depicts the procedure. The procedure calculates the costs for all potential pickup and delivery routes when using a random TP. The pickup route is the route where the pickup will be inserted, the delivery route is the route where the delivery of the same request will be inserted. The TP can either be already in the route or has to be inserted. In the second case, it is inserted at its least costly position in this route. If the insertion of the TP was successful, the algorithm proceeds and inserts the pickup before the TP. Again, if the insertion of the pickup was successful, the algorithm will succeed, otherwise it will try the next pickup route. The insertion of the delivery with transshipment is performed in a similar manner. The only difference lies in the position where the delivery has to be inserted. The delivery is inserted after the TP. For all potential (pickup route)–(TP)–(delivery route) combinations the combination with the lowest cost is determined and the request is inserted accordingly.

Best-with-best TP insertion

The second-last repair operator is similar to the best-with-random TP insertion operator. They differ in the fact that the best-with-best TP insertion operator iterates across all possible TPs and uses the (pickup route)–(TP)–(delivery route) combination with the lowest cost. The benefit is the choice of a TP that fits into the routes by means of short detours. The obvious drawback is the high computation time.

Combined insertion

The combined insertion operator inserts a request by randomly applying one of the aforementioned operators. The operator is selected in the same fashion as the combined removal operator. It can be worthwhile inserting just some of the requests with operators that force the use of one TP, and the rest without transshipments.

6 | COMPUTATIONAL EXPERIMENTS

In this section we introduce the instances used, briefly show how we determined the parameters, evaluate the performance of the ALNS, and lastly provide some managerial insights by means of a sensitivity analysis. All experiments were conducted on an AMD Ryzen 7 2700X CPU with eight cores and 16 GB of RAM. We used Gurobi 8.1 as exact solver and LKH-3 (LKH-3 can be downloaded from http://akira.ruc.dk/~keld/research/LKH-3/) as a benchmark heuristic for the larger PDP instances. The ALNS was coded in C++.

6.1 | Instance generation

The pickup and delivery problem with transshipments and occasional drivers (PDPTOD) is a new problem and therefore no benchmark instances exist. We used Solomon instances [34] and Li and Lim instances [22] to generate appropriate instances. All pickups originate from a central depot (VRP-like) for Solomon instances, and every request has a unique origin (PDP-like) for Li and Lim instances. We only used the coordinates for the requests from these instances, as we do not consider capacities or time windows. We randomly generated the origins and destinations of occasional drivers (ODs) in an area with a lower left corner $[0.9 \cdot \min(x_r), 0.9 \cdot \min(y_r)]$ and an upper right corner $[1.1 \cdot \max(x_r), 1.1 \cdot \max(y_r)]$. Note that x_r denotes the x coordinate of requests and y_r the y coordinate. Four transshipment points (TPs) are generated in the same manner for the case where TPs are randomly distributed in the relevant distribution area (scenario TR). Scenario T4 assumes four TPs at the locations specified in Table 4, which locates the TPs according to the requests. Scenario TR and T4 resemble the model PDPTOD. Scenario T0 goes without any TPs and therefore resembles the model without transshipments, that is, pickup and delivery problem with occasional drivers (PDPOD).

TABLE 5 Overview of test instances

Request distribution	#Requests	TP scenario	#ODs
VRP-like			
R101	25	T0, TR, T4	25
	50	T0, TR, T4	50
	100	T0, TR, T4	100
RC101	25	T0, TR, T4	25
	50	T0, TR, T4	50
	100	T0, TR, T4	100
C101	25	T0, TR, T4	25
	50	T0, TR, T4	50
	100	T0, TR, T4	100
PDP-like			
lr201	51	T0, TR, T4	50
lrc201	51	T0, TR, T4	50
lc201	51	T0, TR, T4	50

Abbreviations: OD, occasional driver; PDP, pickup and delivery problem; TP, transshipment point; VRP, vehicle routing problem.

TABLE 6 Parameters and values for ALNS

Parameter	Default value	Range tested	Chosen value
n_0	400	–	400
χ_0	0.25	[0.05, 0.10, …, 0.90, 0.95]	0.35
TempLow	1	[2, 1, 0.5, 0.2, 0.1, 0.05, 0.01, 0.005]	0.5
coolRate	0.9996	[0.999, 0.9991, …, 0.9999]	0.9998
p^{binom}	0.35	[0.05, 0.10, …, 0.55, 0.60]	0.35
p^{shaw}	6	–	6
W_i	1	–	1
σ_1	30	–	30
σ_2	1	–	1
σ_3	10	–	10
Rate	0.1	–	0.1

Abbreviation: ALNS, adaptive large neighborhood search.

There are 12 request scenarios, each with three different TP scenarios and 30 different OD distributions. As naming convention we propose *Request Distribution-R-T-OD-n*. For example the instance *R101-R25-TR-OD25-1* assumes 25 requests distributed according to R101, randomly distributed TPs and 25 ODs with randomly distributed origin and destination. *R101-R25-T4-OD25-1* is the same instance except for the locations of TPs. This leads to 810 VRP-like instances and 270 PDP-like instances. Table 5 presents an overview of all instances tested.

6.2 | Parameter tuning and analyses of operators

Parameters are tuned using the first instance of the TR, T4, and T0 scenario of R101, RC101, C101 with 100 requests and lr201, lrc201, lc201 with 51 requests (*R101-R100-TR-OD100-1, R101-R-T4-OD100-1, …, lc201-R51-T4-OD50-1*), resulting in 12 tuning instances for the ALNS procedure. Every instance is solved three times. We tune only the parameters that have shown the most impact on the algorithm performance found in preliminary experiments, that is, χ_0, *TempLow*, *coolRate*, and p^{binom}. We set default values to the parameters similar to values found in literature or use values that seemed to deliver the best performance in preliminary experiments. We then fix all parameters except for one that is altered in a specific range. Finally, we decide on a set of parameters that promises a fair compromise of solution quality against runtime. Table 6 summarizes the parameters used by the ALNS procedure independent of the underlying problem situation, whether PDPOD or PDPTOD.

Table 7 shows the performance of the ALNS when removing one operator and using all the remaining operators to solve PDPTOD. Column "Deviation" shows the deviation of the solution quality compared with the solutions obtained by ALNS with all operators. We used the same instances as for parameter tuning. We found that every operator contributes to the overall solution quality. Shaw removal, best insertion, and best-with-best TP insertion operator are especially powerful operators that

TABLE 7 Performance without specific operators

Without operator	Deviation (%)
Random removal	+5.61
Worst removal	+4.46
Shaw removal	+8.15
Least efficient removal	+2.90
Combined removal	+5.35
Random insertion	+3.66
Best insertion	+14.88
Regret-3 insertion	+4.46
Best-with-random TP insertion	+3.76
Best-with-best TP insertion	+12.14
Combined insertion	+6.70

Abbreviation: TP, transshipment point.

TABLE 8 Results Gurobi versus ALNS solution for instances with $|R| \leq 10$, $|T| = 4$, $|\tilde{K}| = 5$

	Gurobi			ALNS			
Instances	Cost	Runtime (s)	Gap (%)	Cost	Runtime (s)	Gap (%)	TP used
R101-R8-RT-OD5	130	682	0	130	<1	0	y
RC101-R8-RT-OD5	59	7200	12	59	<1	12	y
C101-R8-RT-OD5	49	7200	57	36	<1	41	y
lr201-R8-RT-OD5	115	10	0	115	<1	0	y
lrc201-R8-RT-OD5	162	7200	29	162	<1	29	n
lc201-R8-RT-OD5	229	2529	0	229	<1	0	n
R101-R10-RT-OD5	173	579	0	173	<1	0	n
RC101-R10-RT-OD5	124	7200	59	86	<1	41	y
C101-R10-RT-OD5	58	7200	52	55	<1	49	y
lr201-R10-RT-OD5	130	40	0	130	<1	0	y
lrc201-R10-RT-OD5	–	7200	–	171	<1	–	n
lc201-R10-RT-OD5	154	1113	0	155	<1	1	n

Abbreviation: ALNS, adaptive large neighborhood search.

should not be left out. Even the random operators seem to increase the solution quality by introducing some diversification, which ultimately helps finding better solutions.

6.3 | Performance evaluation

The performance of the ALNS is evaluated by comparing its results against the exact solution obtained by solving model PDPTOD with Gurobi. As Gurobi only solves small instances within reasonable times, we evaluate the performance of the ALNS against the heuristic solution of the LKH-3 for larger instances with up to 100 requests. Note that LKH-3 is only able to solve the classical PDP without ODs and without TPs, that is, $|T| = 0$, which is a special case of model PDPTOD. We denote this case as a "baseline scenario." Finally, the solutions of instances with ODs and TPs are compared against the previously solved baseline scenario, that is, the case without ODs and without TPs.

6.3.1 | Benchmark against Gurobi

Twelve small instances with $|R| \leq 10$, $|T| = 4$, $|\tilde{K}| = 5$, $\alpha = 12$, $\beta = 2.0$, $p^{\text{stop}} = 0$, $p^{\text{detour}} = 1.0$ are solved with Gurobi and the ALNS. The time limit for Gurobi was set to 7200 s. Gurobi and the ALNS are run only once. Table 8 presents the results.

Note that the column "Gap" represents the MIP Gap, that is, the difference versus the current lower bound. Gurobi finds the optimal solution ($Gap = 0$) within the time limit for six instances. For five instances it achieves a feasible integer solution, it fails only for one instance. The ALNS achieves the same solutions for five instances. For three instances it finds a better solution within the given time limit of the ALNS procedure. The ALNS achieves slightly worse results only for one instance. The ALNS needs less than 1 s for each instance, while Gurobi needs 4013 s on average. The ALNS achieves comparable results for small instances. Unfortunately, it is not possible to compare the performance against Gurobi for larger instances because the runtime

TABLE 9 Results of LKH-3 versus ALNS for instances with $|T| = 0$, $|\tilde{K}| = 0$

	LKH-3		ALNS		
Instance	Cost	Runtime (s)	Cost	Runtime (s)	Gap (%)
R101-R25-T0-OD0	311	<1	311	<1	0
RC101-R25-T0-OD0	226	<1	226	<1	0
C101-R25-T0-OD0	133	<1	133	<1	0
R101-R50-T0-OD0	455	<1	455	3	0
RC101-R50-T0-OD0	370	<1	370	2	0
C101-R50-T0-OD0	242	<1	242	2	0
lr201-R51-T0-OD0	709	35	723	4	2.0
lrc201-R51-T0-OD0	743	11	737	4	-0.8
lc201-R51-T0-OD0	543	<1	543	2	0
R101-R100-T0-OD0	629	17	629	10	0
RC101-R100-T0-OD0	637	29	637	10	0
C101-R100-T0-OD0	501	17	501	8	0

Abbreviation: ALNS, adaptive large neighborhood search; LKH, Lin-Kernighan-Helsgaun.

TABLE 10 Results achieved by approach PDPOD (T0) versus approach PDPTOD (TR, T4) for instances with $|R|, |\tilde{K}| \leq 100$

		Cost (%)			Hypotheses[a]			Average runtime (s)		
		T0	TR	T4	T0 = TR	T0 = T4	TR = T4	T0	TR	T4
	R25, OD25	64.60[b]	54.38	50.75	***	***	**	1.10	3.19	3.84
R101	R50, OD50	55.86	51.64	46.92	***	***	***	3.63	14.13	19.14
	R100, OD100	42.22	41.32	39.01	o	**	**	11.85	115.63	161.02
	R25, OD25	81.59	61.64	54.53	***	***	***	1.09	3.06	3.37
RC101	R50, OD50	67.80	56.65	51.63	***	***	**	3.53	15.49	18.57
	R100, OD100	50.82	49.68	46.79	o	***	*	13.25	113.55	168.93
	R25, OD25	98.02	82.31	75.76	***	***	***	0.46	2.18	2.35
C101	R50, OD50	80.29	68.35	59.82	***	***	***	3.52	11.49	15.10
	R100, OD100	48.32	49.56	44.25	o	***	**	14.09	113.97	145.89

Abbreviations: PDPOD, pickup and delivery problem with occasional drivers; PDPTOD, pickup and delivery problem with transshipments and occasional drivers.
[a]Shows if the hypothesis can be rejected and the according p-value. ***$p \leq 0.001$, **$p \leq 0.01$, *$p \leq 0.05$, $^{o}p > 0.05$.
[b]Example calculation: $Cost\ (\%) = \frac{cost^{\text{av}}}{cost^{\text{base}}} \cdot 100 = 64.60$ with $cost^{\text{av}} = \frac{\sum_{i=1}^{30} f(\text{R101-R25-T0-OD25-i})}{30} = 200.91$ and $cost^{\text{base}} = 311$.

becomes prohibitive. However, we analyze the quality of the ALNS procedure by comparing the solutions achieved against the LKH-3 procedure, assuming the baseline scenario where ODs and TPs are not present, that is, the PDP modeling approach.

6.3.2 | Benchmark against LKH-3

This section compares the results of the ALNS approach with the LKH-3 approach, assuming scenarios without ODs and TPs. The LKH-3 and the ALNS are both run three times, the best solution found, and the average runtime is reported in Table 9. The difference in terms of costs is reported in column Gap (%), where $Gap = \frac{Cost_{\text{ALNS}} - Cost_{\text{LKH-3}}}{Cost_{\text{LKH-3}}}$.

Both heuristics achieve the same results for 10 out of 12 instances. In one case the ALNS produces slightly worse results, in one case slightly better. The LKH-3 needs 10 s on average, the ALNS only 4 s. The ALNS produces high-quality solutions for cases with up to 100 requests, assuming no TPs and no ODs, within a short computation time. Note that the results for this baseline scenario serve as an upper bound for the following experiments. The objective value cannot get worse if TPs or ODs are available because it is always possible to only use regular drivers (RDs) without deploying ODs and without fulfilling requests via TPs.

6.3.3 | Comparison with baseline scenario

Table 10 shows the results obtained by solving instances with up to 100 requests, 100 ODs, $\alpha = 12$, $\beta = 2.0$, $p^{\text{stop}} = 0$, and $p^{\text{detour}} = 1.0$. Cost (%) is a standardized value. It is calculated by dividing the average cost, $cost^{\text{av}}$, by the best known cost of the baseline scenario, $cost^{\text{base}}$, without TPs and without ODs, as denoted in Table 9.

A Wilcoxon test on the paired samples is executed to check the following three hypotheses.

- Column T0 = TR: "Scenarios T0 and TR lead to the same solution quality"
- Column T0 = T4: "Scenarios T0 and T4 lead to the same solution quality"
- Column TR = T4: "Scenarios TR and T4 lead to the same solution quality"

The PDPTOD modeling approach usually leads to solutions with significantly lower costs compared with the approach of neglecting TPs, that is, the PDPOD modeling approach. We are only unable to show a significant difference in the solution quality of scenario T0 and TR for instances with 100 requests. In general, the costs achieved by approach PDPTOD (TR, T4) should not be higher than the costs achieved by approach PDPOD (T0) because TPs are only used if cost advantages can be realized. The ALNS procedure successfully achieves cost advantages in all but one case (C101-R100-TR-OD100). We can also reject the hypotheses that scenarios TR and T4 lead to the same solution quality, that is, the evidence provided in Table 10 shows that a reasonable placement of TPs has the potential to decrease costs.

The ALNS procedure scales well with the instance size. Small instances with R25, OD25 are solved in few seconds. Larger instances are solved in less than 20 s, except for instances with R100, OD100, and TR or T4, which take considerably more time because of the increasing number of possible solutions when TPs are available.

6.4 | Sensitivity analysis

Since crowdshipping is a relatively new delivery concept, it is important to understand how the characteristics of planning scenarios impact the economic advantages of this concept. The following investigation may therefore be of interest for companies who are experimenting with crowdshipping. We analyze the impact of the number and flexibility of ODs, the number and location of TPs, and the compensation scheme for ODs on diverse dependent variables. Dependent variables are total costs, the share of ODs used, and the share of requests served via TPs. We use randomly distributed and clustered VRP-like instances with $|R| = 50$ (R101-R50 and C101-R50) to illustrate these effects. Please note that because of space limitations we do not present the results for the additional instances mentioned in Table 5 in detail as long as these instances result in equivalent effects.

We assume three flexibility levels: low ($\alpha = 4, \beta = 0.2$), medium ($\alpha = 8, \beta = 0.6$), and high ($\alpha = 12, \beta = 1.0$). As a compensation scheme we assume $p^{\text{detour}} = 1$ and $p^{\text{stop}} = 0$, that is, ODs receive the same compensation as RDs. In this case, costs are synonymous with the total distance traveled. The number of available ODs varies within the following range: $|\tilde{K}| = [10, 20, 30, 40, 50]$.

6.4.1 | Analyzing total costs

Figures 4 and 5 show the impact of the number of ODs, the flexibility level of ODs, and the transshipment scenario chosen on the objective value achieved. The horizontal and vertical axes denote the number of ODs and the standardized objective value (%), respectively. The objective value quantifies the percentage of the total costs achieved compared with the total costs in a scenario without any ODs. Three different lines (light-gray dashed line, gray dot-dashed line, black dotted line) represent the transshipment scenario (T0, TR, T4).

It is possible to achieve significant cost savings if more ODs are available. However, the cost savings strongly depend on the flexibility of ODs and how the requests are distributed. The cost savings per additional OD are higher if the requests are randomly distributed and the flexibility of the ODs is high (see Figure 4(B)). Minor cost effects can be observed on clustered instances with less flexible ODs, especially if no transshipment points (T0) are available (see Figure 5(A)). In the event of clustered requests, the ODs should offer a certain degree of flexibility in order to be able to serve an entire cluster with a limited number of ODs, since an entire cluster can also be efficiently served by a single RD (see Figures 5(A) vs. (B)). Thus, if the flexibility of ODs is limited, it is more cost efficient to serve the entire cluster with only one RD and not with a combination of ODs and RDs. In addition, the impact of TPs on the objective value is greater for clustered instances (C101) than for random instances (R101). This becomes especially visible if the TPs are placed in a somewhat reasonable manner, for example, scenario T4 (see black dotted line in Figure 5(B)). The average effect of TPs on the objective value is relatively small in our numerical experiments. This confirms the findings of Sampaio Oliveira et al. [33]. They found small positive cost effects if the driver shift length is long and the distance between pickup and delivery points is short. However, we observe additionally to their findings that the impact of TPs on the objective value strongly depends on the distribution of requests.

6.4.2 | Analyzing the share of ODs used

We assume that the number of ODs used is important to achieve long-term stability. Low utilization may lead to OD dissatisfaction. If ODs do not receive tasks repeatedly, it is likely that they will no longer offer their services on the platform and

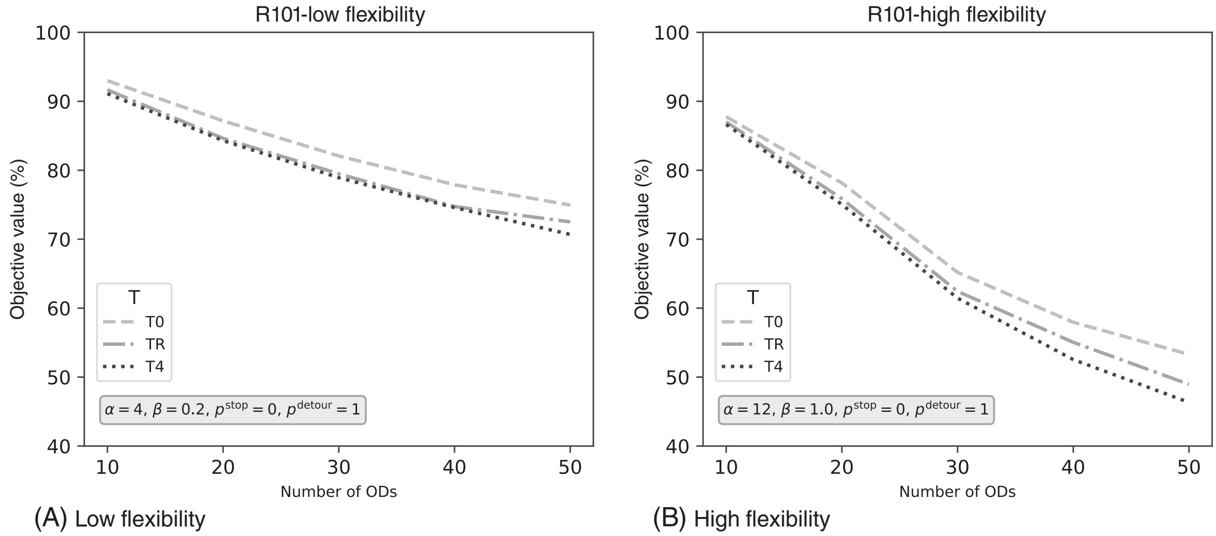


FIGURE 4 Objective value depending on the number of available ODs—random instances, R101-R50. OD, occasional driver

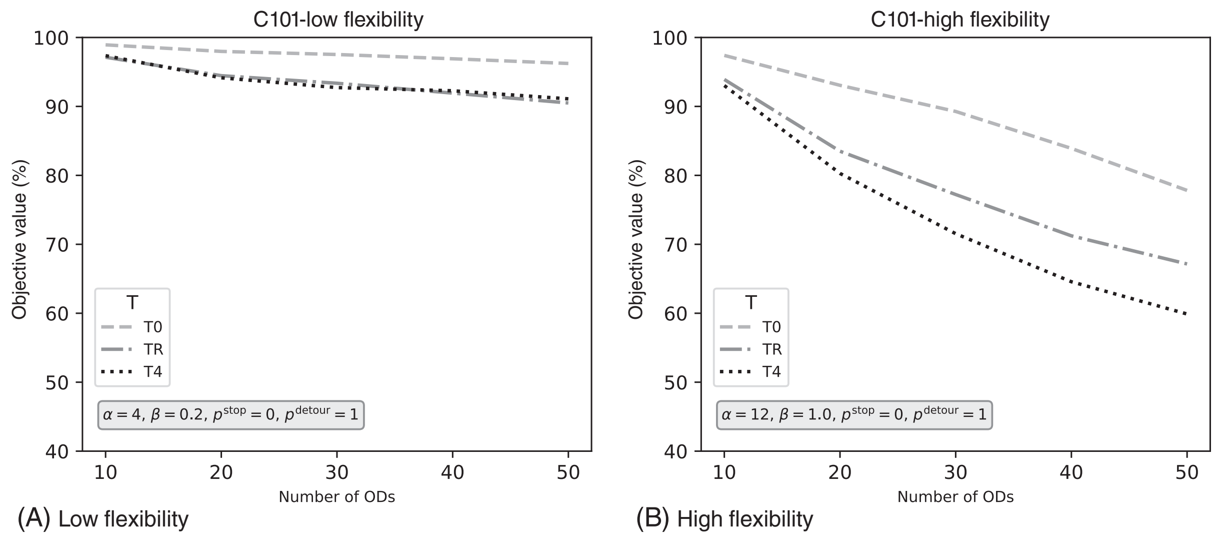


FIGURE 5 Objective value depending on the number of available ODs—clustered instances, C101-R50. OD, occasional driver

henceforth fewer ODs will be available to fulfill the requests. We therefore analyze this issue in the present subsection. Figures 6 and 7 show the impact of the number of available ODs, the flexibility, and the transshipment scenario on the share of the entire set of available ODs used to fulfill the requests. Note that the share of ODs used quantifies the average number of ODs deployed compared with the total number of available ODs.

Obviously, the deployment of an OD greatly depends on her/his willingness to undertake detours, that is, her/his flexibility to make extra trips. Thus, an OD would never be deployed for a trip that is located outside her/his direct range. In the instances analyzed, only a limited share of ODs could directly serve a request. We visualize this share of ODs by adding a solid black line into the charts of Figures 6 and 7. If a request is located outside the direct range of an OD, the OD can only be deployed during a fulfillment process if the request is served via a TP. This means that in cases without TPs (T0, light-gray dashed line), the share of ODs deployed cannot be higher than the average share of ODs who are in the direct range of a request (solid black line). Figures 6 and 7 also show that the share of ODs used greatly depends on the request distribution, but only slightly depends on the number of available ODs. This means that the absolute number of ODs used increases with the number of available ODs. Therefore it is not against the ODs' interests to have more ODs to compete with, in some cases it is even beneficial for the ODs.

Higher flexibility increases utilization. The platform provider should therefore ensure that the ODs are flexible, that is, willing to deviate from the route originally planned, and to accept additional stops. The share of ODs used is higher for instances where requests are randomly distributed (see Figure 6). In these cases, many ODs get to serve only a small number of requests per OD. By contrast, for clustered instances the utilization of ODs is relatively low (see Figure 7) because clusters are better

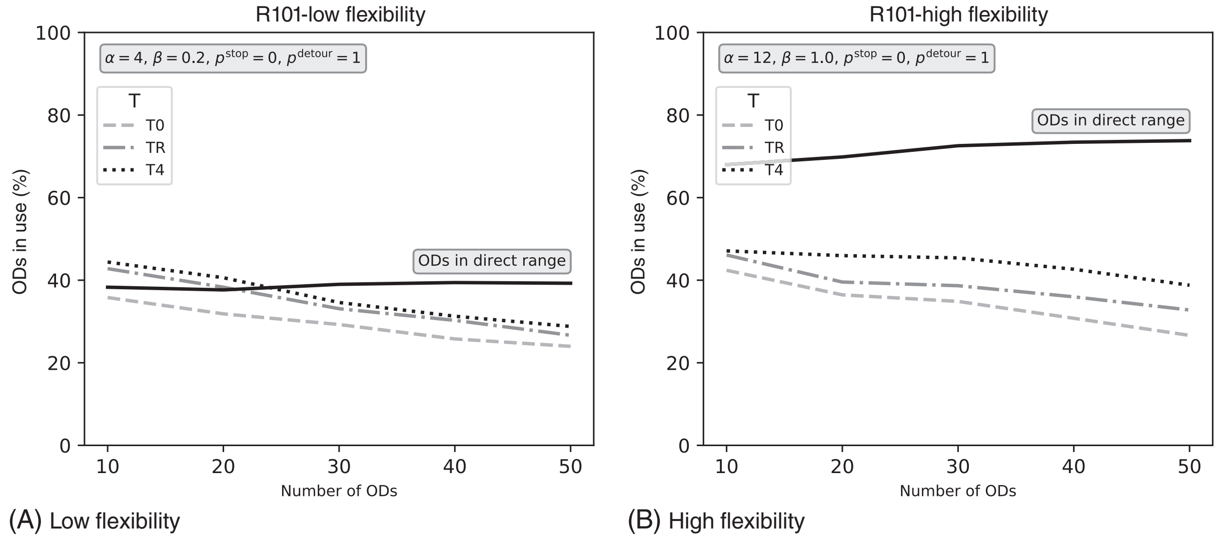


FIGURE 6 Share of ODs used depending on the number of available ODs—random instances, R101-R50. OD, occasional driver

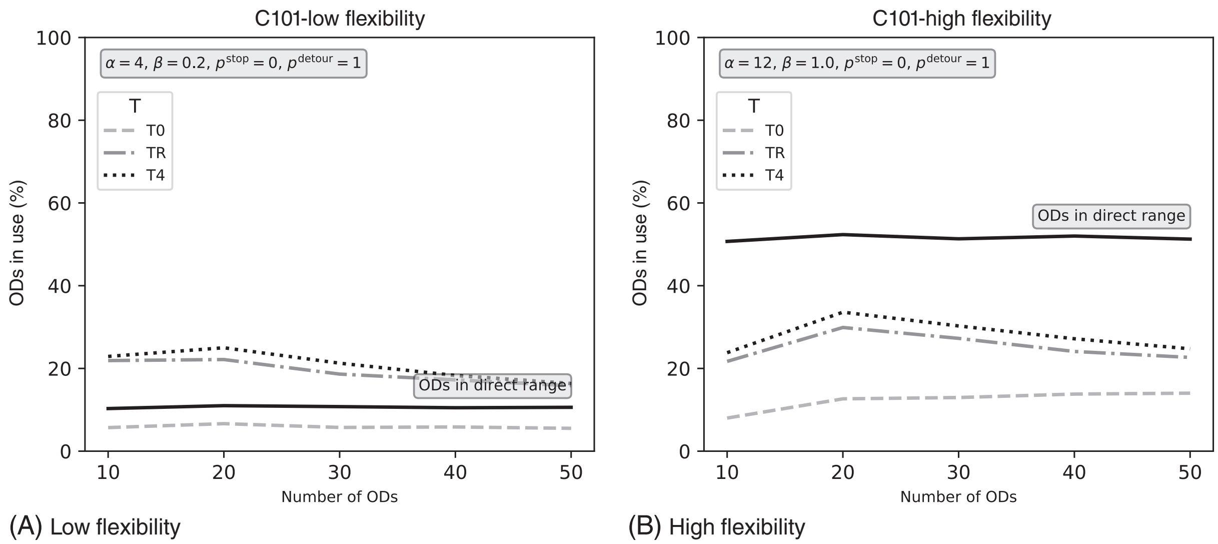


FIGURE 7 Share of ODs used depending on the number of available ODs—clustered instances, C101-R50. OD, occasional driver

served with just a few ODs. Figure 8 shows an illustrative example where the upper left cluster—far away from the origin of requests—is served by one OD only, who—of course—offers high flexibility.

TPs improve utilization. ODs who were not able to serve any request without TPs because of the detour constraint now find themselves in the range of some requests delivered via TPs and can also be deployed. This effect becomes more pronounced as flexibility decreases. In some cases (e.g., Figure 7(A)), the number of ODs used is even higher than the average share of ODs located in the direct range of at least one request (solid black line vs. TR and T4 scenarios).

The results shown focus on VRP-like instances. The share of ODs deployed, however, is much higher for PDP-like instances, which are not presented here in detail. In those cases, the deployment of ODs only slightly depends on the flexibility offered by ODs, the availability of TPs, and the total number of available ODs. A platform operating in PDP-like circumstances can therefore guarantee higher and more stable utilization of ODs than a platform operating in VRP-like circumstances.

6.4.3 | Analyzing the share of requests served via TPs

Figures 9 and 10 show the impact of the number of ODs, the flexibility level, and the transshipment scenario on the share of requests served via TPs.

Note that the charts do not show results for the instances that exclude TPs, that is, scenario T0. Obviously, in those cases no request can be fulfilled via TPs.

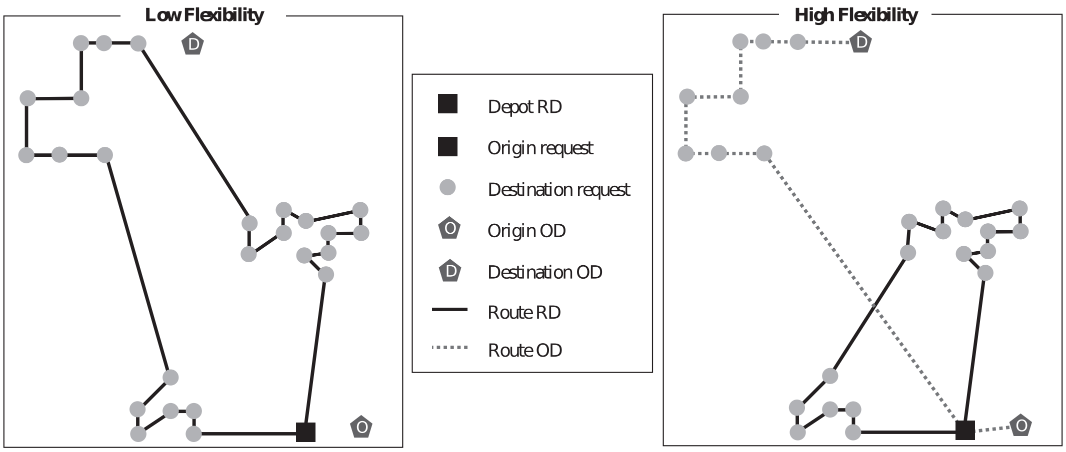


FIGURE 8 Illustrative example of a cluster served by only one OD. OD, occasional driver; RD, regular driver

The share of requests served via transshipment points considerably depends on the flexibility level offered by the ODs, and greatly depends on where the TPs are located in the distribution area. In cases where ODs only offer low flexibility, TPs are rarely used (see Figures 9(A) and 10(A)). Nevertheless, the availability of TPs increases the deployment of ODs, as demonstrated in the previous subsection. However, in cases where ODs offer high flexibility, the utilization of TPs substantially increases with the increasing number of available ODs (see Figures 9(B) and 10(B)). The T4 scenario (black dotted line), in addition, reveals the relevance of placing TPs appropriately in the distribution area. In this case the TPs are located in accordance with where requests occur. Thus, almost every request is served via a TP. The solution for clustered instances results in a two-stage distribution structure for 64% of the requests, on average, where the TPs (first stage) are served via RDs and via ODs in the second stage. ODs additionally take over the first stage of the transport for randomly distributed instances. TPs are at most used for 48% of requests in the case of PDP-like instances (randomly generated instances, scenario T4 with 50 OD), and ODs usually serve the first and second stage. A two-stage distribution structure is therefore not obvious for PDP-like instances. Please note that a distribution structure larger than two stages cannot occur since the ALNS procedure developed only allows using a maximum of one TP when fulfilling a delivery request.

Summarizing the observations above, it is highly important to locate the TPs in accordance with the demand arising, and not just randomly. In addition, a platform provider should preferably set up a two-stage structure if ODs offer high flexibility and clustered requests originate mostly from the depot.

6.4.4 | Analyzing the compensation per additional stop

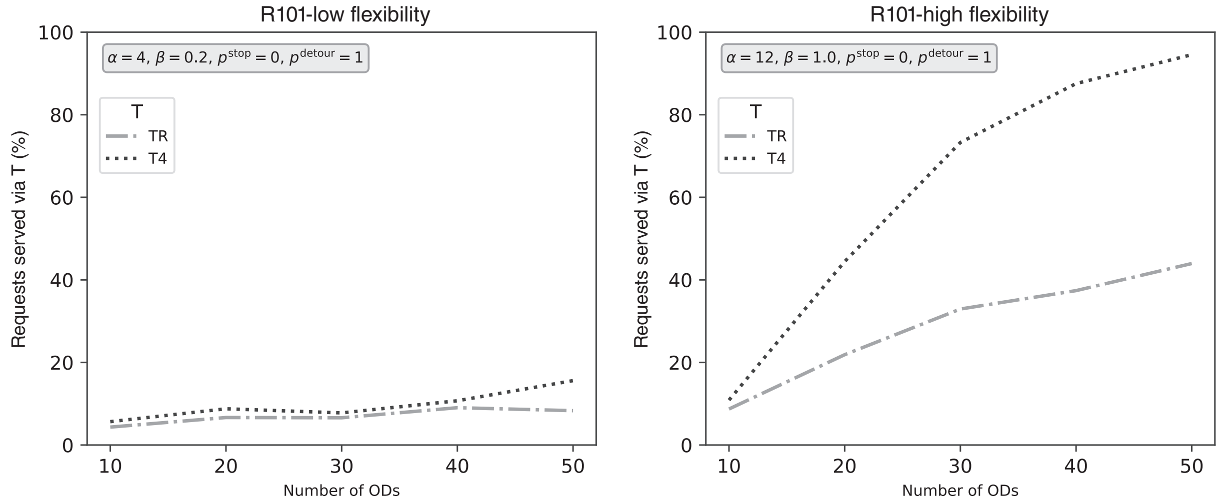
Figure 11 shows the impact of the compensation paid to ODs per additional stop, p^{stop}, on the objective value (Figure 11(A)), on the share of ODs used (Figure 11(B)), and on the share of requests served via TPs (Figure 11(C)). The C101 instance type with medium flexibility serves here to illustrate the effect, as it is very pronounced in this case. Nevertheless, the effects can also be observed in other instances, albeit in a less pronounced form.

As expected, cost savings decrease as the compensation per stop increases. The share of ODs used also decreases gradually with increasing p^{stop}. For Figure 11(A,B), it can be observed that there seems to be a critical compensation per stop, when the usage of TPs is no longer beneficial. This critical compensation can be determined from Figure 11(C). The share of requests served via TPs for the T4 scenario drops from a stable level of around 90% if $p^{\text{stop}} < 0.7$ to under 20% if $p^{\text{stop}} > 1.4$.

6.4.5 | Analyzing the influence of temporal flexibility

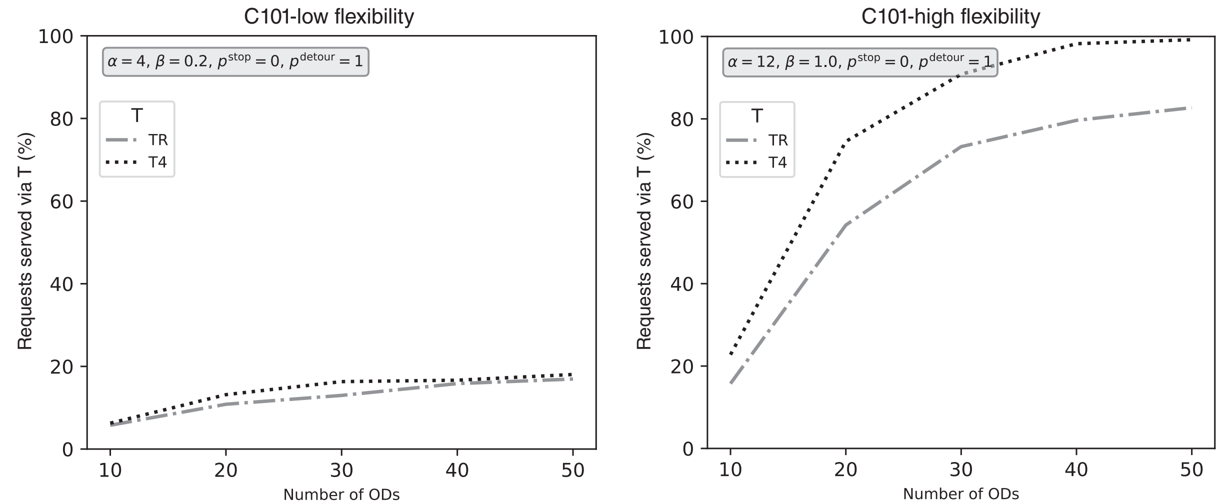
The model formulation principally assumes sufficiently high temporal flexibility of ODs such that delivery schedules that result from the associated task assignments to ODs can always be realized. However, drivers generally have limited temporal flexibility, for example, commuters should arrive at their working place at a specific time and cannot leave their working place before the end of work. They cannot generally organize their private schedule in order to fulfill all tasks that might be realizable related to their spatial degree of freedom. The duration of an OD's route is not explicitly limited within our model formulation. The model formulation only limits the number of stops and the length of the detour for each individual OD's offer. This however implicitly limits the duration of the OD's routes generated. Nevertheless, the duration of a route can become unacceptably long if an OD has to wait at a TP for a parcel that is not yet available at their desired departure time.

Feasible OD schedules can only be assured for all instances if the modeling approach contains concrete time windows for all trips and the associated detours offered by an OD. This would however substantially increase model complexity and would



(A) Low flexibility (B) High flexibility

FIGURE 9 Share of requests served via TPs depending on the number of available ODs—random instances, R101-R50. OD, occasional driver; TP, transshipment point



(A) Low flexibility (B) High flexibility

FIGURE 10 Share of requests served via TPs depending on the number of available ODs—clustered instances, C101-R50. OD, occasional driver; TP, transshipment point

also possibly cause some waiting times for ODs who may have to wait at TPs, as traveling times may be stochastic in reality. Nevertheless, we cope with this issue by letting ODs indicate whether their trip is planned before midday or after. ODs only deliver parcels to TPs but do not gather parcels there if they offer their services before midday. By contrast, ODs gather parcels from TPs but do not deliver to TPs if they offer their services after midday. The direct delivery of parcels is not affected by these assumptions. The commitments of the ODs guarantees that all parcels that are not directly delivered are available at TPs before or at midday at the latest. Table 11 shows the results obtained by solving instances, assuming half of the ODs are available before midday and the other half after midday.

The results are generated in the same manner as the results displayed in Table 10. In that case, the potential TPs' cost savings are obviously lower than when relaxing ODs' potential temporal constraints. Nevertheless, the savings are still significant depending on the type of instance. Instances *C101*, for example, allows for about 10% additional cost savings when TPs are available, that is, 80.29% for scenario T0 and 70.32% for scenario T4 with limited temporal flexibility. The gap between the limited and unlimited TR scenarios is more pronounced for clustered instances (C, RC) because TPs are more frequently used compared with random scenarios (R); see Figures 9(B) and 10(B).

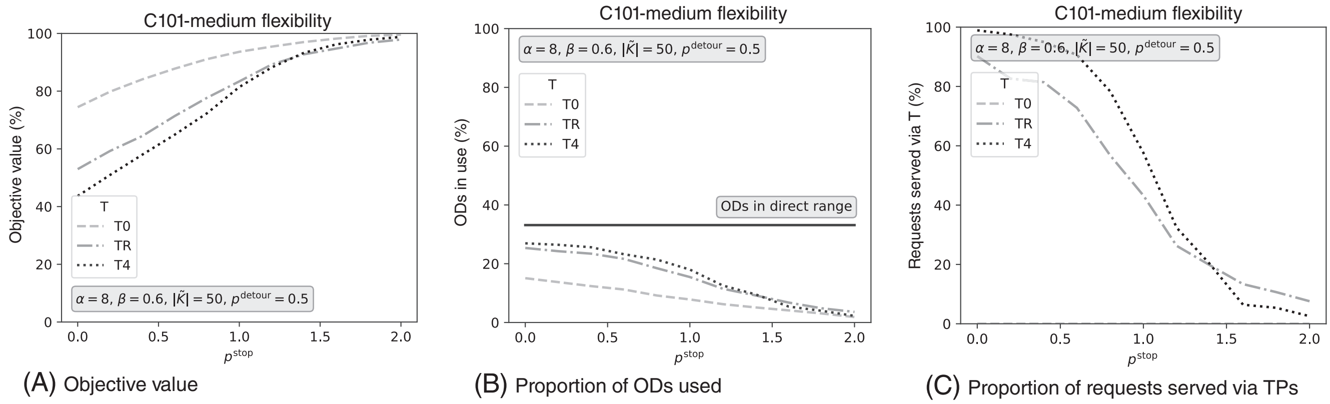


FIGURE 11 Impact of compensation rate p^{stop} on the results—clustered instances, C101-R50, medium flexibility. OD, occasional driver; TP, transshipment point

TABLE 11 Results achieved by approach PDPOD (T0) versus approach PDPTOD (TR, T4) for instances with (un)limited temporal flexibility

Instances		Costs (%)				
			TR		T4	
		T0	Limited	Unlimited	Limited	Unlimited
R101	R50, OD50	55.86	52.12	51.64	52.12	46.92
RC101	R50, OD50	67.80	66.63	56.65	63.99	51.63
C101	R50, OD50	80.29	72.78	68.35	70.32	59.82

Abbreviations: PDPOD, pickup and delivery problem with occasional drivers; PDPTOD, pickup and delivery problem with transshipments and occasional drivers.

6.4.6 | Managerial insights

This section summarizes the findings of the sensitivity analysis and gives some advice on how to include crowdshipping in the last-mile delivery process. The use of ODs has the potential for cost savings independent of the request distribution (R, RC, C) or fulfillment structure (VRP-like vs. PDP-like), but is dependent on the flexibility offered by ODs. The platform provider should therefore create incentives for ODs that guarantee at least a moderate level of OD flexibility.

The benefits from TPs depend on the request distribution. If requests are evenly distributed in the area, savings will be high and the utilization of ODs will be acceptable, even without TPs. However, for clustered or semiclustered (C, RC) request distributions, TPs reduce costs and increase the utilization of ODs, which would otherwise be rather low. The location of TPs is of considerable importance and should be carefully determined according to the distribution of requests. As soon as a critical mass of ODs is available, the platform provider should consider establishing a two-stage distribution system where RDs serve only TPs and ODs take over the last-mile delivery from TPs to the destination of requests. The fact that TPs potentially increase the number of ODs deployed is an especially valid reason for introducing TPs.

Customer requests fulfilled late or even unfulfilled lead to dissatisfied customers, which in the end also reduces the number of requests. Future initiatives to set up crowdshipping platforms should therefore carefully address the questions of how to attract a critical mass of ODs and how to guarantee a high service level in all circumstances. The analyses presented here reveal some valuable insights related to these questions. Above all, a business model that deploys a sufficient number of RDs alongside ODs increases the level of service, that is, fulfills a high share of requests, thus maintaining high customer satisfaction, which is important to keep customers using the platform. In addition to this general delivery framework, offering fair compensation for ODs, as assumed in our analyses, is obviously a relevant factor attracting an adequate number of ODs. However, fair payment on a one-off basis is insufficient to keep ODs on the platform in the long term if ODs are only rarely deployed. TPs increase the deployment of ODs in all settings analyzed, but this in turn depends on the compensation per stop for ODs. As a result, the compensation per additional stop, p^{stop}, has to be carefully quantified in order to benefit from operating TPs. On the one hand, almost all delivery requests are assigned by the platform provider to RDs—for economic reasons—if the cost rate is set too high. In that case, TPs remain unused and ODs are also not deployed although they would be highly motivated to offer their services. On the other hand, ODs will most likely not participate if the cost rate is set too low. In that case, a compensation solely for the detour would generally be insufficient to motivate potential ODs.

Summarizing these thoughts, setting up appropriate TPs increases the utilization of ODs. The employment of RDs ensures a high customer satisfaction, so customers keep on using the platform. A crowdshipping business model operating TPs and employing RDs, defines suitable circumstances for analyzing and setting up an appropriate compensation scheme for ODs. Then, in a later phase of business activity, that is, as soon as a critical mass of requests and ODs is available, the platform provider may further develop their business concept. For example, they may only use ODs without RDs, or apply the aforementioned two-stage distribution approach.

7 | CONCLUSIONS AND FUTURE AREAS OF RESEARCH

7.1 | Conclusions

This article introduces a variant of the static pickup and delivery problem with transshipments and occasional drivers, termed PDPTOD. In this setting, occasional and regular drivers (ODs and RDs) may pick up parcels at the origin, deliver these directly, or transfer them to other drivers at dedicated transshipment points (TPs). The platform provider can choose between RDs and rather constrained ODs who are willing to deviate from the route they had originally planned and take on the fulfillment of one or more requests. We develop a novel MIP model (PDPTOD) that we use to solve small instances. In order to solve larger instances with up to 100 requests, we develop a heuristic solution approach based on an adaptive large neighborhood search (ALNS) for the optimization problem (PDPTOD) formulated. The ALNS procedure produces equivalent results for small instances, compared with the exact solution obtained by Gurobi, and for larger instances compared with results obtained by LKH heuristics. The ALNS scales well with the problem size. The numerical experiments based on known instances from literature extended by ODs and TPs show that the usage of both significantly reduces total costs. The cost savings potential highly depends on the assumed flexibility of ODs. The cost savings are only moderate if ODs are unwilling to make extra trips. By contrast, substantial cost savings are achievable if ODs are willing to make additional trips. In addition, setting up TPs reduces total costs as well. This occurs in both cases, that is, where requests are distributed randomly in the delivery area, and where requests predominantly emerge in specific regions of the delivery area. The availability of TPs will certainly increase the utilization of ODs and will lead to a more equal distribution of workload between RDs and ODs. This is a very relevant aspect when setting up a crowdshipping platform. ODs are more likely to offer their services on a platform if they are consistently assigned requests to serve. To get the most out of setting up TPs, it is important to locate them in accordance with the demand arising, and not just randomly. The scenario where we positioned TPs in accordance with demand achieves superior results in almost all cases compared with the scenario where TPs are randomly distributed in the delivery area.

7.2 | Future areas of research

The suggested modeling and solution approach includes specific—possibly restricted—assumptions that offer avenues for future research.

(a) ODs are only restricted by spatial limitations; strict temporal restrictions are not included in our setting. Time windows for ODs are a realistic assumption and should be included in further research. The effect of the length of time windows on the critical mass can be examined, for example.
(b) The present research neglects delivery time windows, too. Approaches that assume customer-preferred delivery time windows or time windows with higher customer attendance rates will certainly increase the probability of successful deliveries.
(c) The timeline of the modeling approach assumes that all data are known when the planning takes place. Customer requests and offers from ODs may however arrive on a dynamic and/or stochastic basis. Specialized modeling and solution approaches are required for those planning situations.
(d) The locations of the TPs in the scenario analyzed are determined via a simple heuristic. It can be assumed that a more elaborate model and solution procedure for planning the location of TPs will impact the results, and will also increase the use of ODs.
(e) A two-stage approach where RDs only serve the first stage and ODs the second stage, that is, the very last mile to customers, could be a viable option for established platforms with many registered ODs, and therefore offers an additional avenue for further research.
(f) The compensation for ODs has to be determined carefully because it is probably the most important reason for ODs to participate in the long run. Studies dealing with fair compensation for ODs and determining appropriate types of compensation schemes would be of great value to further encourage crowdsourced logistics in the future.

(g) ODs are not obliged to accept the delivery of requests. Solutions may therefore not be robust if many ODs decline requests, or do not fulfill requests even if they have pledged to do so. Empirical research is needed to quantify the risk of unfulfilled requests and identify measures to motivate ODs to accept requests.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers and the guest editors of the special issue for their valuable recommendations, which have significantly improved the paper. Open Access funding enabled and organized by Projekt DEAL.

ORCID

Heinrich Kuhn https://orcid.org/0000-0003-3704-4042

REFERENCES

[1] C. Archetti, M. Savelsbergh, and M. G. Speranza, *The vehicle routing problem with occasional drivers*, Eur. J. Oper. Res. **254** (2016), 472–480.
[2] A. M. Arslan, N. Agatz, L. Kroon, and R. Zuidwijk, *Crowdsourced delivery—a dynamic pickup and delivery problem with ad hoc drivers*, Transp. Sci. **53** (2018), 222–235.
[3] W. Ben-Ameur, *Computing the initial temperature of simulated annealing*, Comput. Optim. Appl. **29** (2004), 369–385.
[4] H. Buldeo Rai, S. Verlinde, J. Merckx, and C. Macharis, *Crowd logistics: An opportunity for more sustainable urban freight transport?* Eur. Transp. Res. Rev. **9** (2017), 39.
[5] W. Chen, M. Mes, and M. Schutten, *Multi-hop driver-parcel matching problem with time windows*, Flex. Serv. Manuf. J. **30** (2017), 517–553.
[6] C. E. Cortés, M. Matamala, and C. Contardo, *The pickup and delivery problem with transfers: Formulation and a branch-and-cut solution method*, Eur. J. Oper. Res. **200** (2010), 711–724.
[7] I. Dayarian and M. Savelsbergh, *Crowdshipping and same-day delivery: Employing in-store customers to deliver online orders*, Prod. Oper. Manag. **29** (2020), 2153–2174.
[8] A. Devari, A. G. Nikolaev, and Q. He, *Crowdsourcing the last mile delivery of online orders by exploiting the social networks of retail store customers*, Transp. Res. E Logist. Transp. Rev. **105** (2017), 105–122.
[9] L. Doerrzapf, M. Berger, G. Breitfuss, and E. Remele, *Crowd Delivery als neues Lieferkonzept zur Stärkung des lokalen Marktplatzes.*, REAL CORP 2016 Proceedings, 2016, pp. 197–206.
[10] M. Drexl, *Synchronization in vehicle routing—a survey of VRPs with multiple synchronization constraints*, Transp. Sci. **46** (2012), 297–316.
[11] M. Drexl, *Applications of the vehicle routing problem with trailers and transshipments*, Eur. J. Oper. Res. **227** (2013), 275–283.
[12] L. Einav, C. Farronato, and J. Levin, *Peer to peer markets*, Ann. Rev. Econ. **8** (2016), 615–635.
[13] E. Estellés-Arolas and F. G. L. de Guevara, *Towards an integrated crowdsourcing definition*, J. Inf. Sci. **38** (2012), 189–200.
[14] K. Helsgaun, *An effective implementation of the Lin—Kernighan traveling salesman heuristic*, Eur. J. Oper. Res. **126** (2000), 106–130.
[15] K. Helsgaun, *An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems: Technical report*, Roskilde: Roskilde Universitet, 2017.
[16] A. Hübner, A. Holzapfel, H. Kuhn, and E. Obermair, *Distribution in Omni-channel grocery retailing: An analysis of concepts realized*, S. Gallino, and A. Moreno, *Operations in an Omnichannel World*, Springer Series in Supply Chain Management, Springer, New York, 2019, 283–309.
[17] A. Hübner, H. Kuhn, and J. Wollenburg, *Last mile fulfilment and distribution in omni-channel grocery retailing: A strategic planning framework*, Int. J. Retail Distrib. Manage. **44** (2016), 228–247.
[18] M. Joerss, J. Schroeder, F. Neuhaus, C. Klink, and F. Mann, *Parcel delivery — The future of last mile*, McKinsey & Company — Travel, Transport and Logistics, 2016.
[19] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon, *Optimization by simulated annealing: An experimental evaluation; Part I, Graph partitioning*, Oper. Res. **37** (1989), 865–892.
[20] N. Kafle, B. Zou, and J. Lin, *Design and modeling of a crowdsource-enabled system for urban parcel relay and delivery*, Transp. Res. B Methodol. **99** (2017), 62–82.
[21] T. V. Le, A. Stathopoulos, T. V. Woensel, and S. V. Ukkusuri, *Supply, demand, operations, and management of crowd-shipping services: A review and empirical evidence*, Transp. Res. C Emerg. Technol. **103** (2019), 83–103.
[22] H. Li and A. Lim, *A metaheuristic for the pickup and delivery problem with time windows*, Int. J. Artif. Intell. Tools **12** (2003), 173–186.
[23] G. Macrina, L. Di Puglia Pugliese, F. Guerriero, and D. Laganà, *The vehicle routing problem with occasional drivers and time windows*, in *Optimization and decision science: Methodologies and applications*, Springer International Publishing, Cham, Switzerland, 2017, 577–587.
[24] R. Masson, F. Lehuede, and O. Peton, *An adaptive large neighborhood search for the pickup and delivery problem with transfers*, Transp. Sci. **47** (2013), 344–355.
[25] A. McKinnon, *Crowdshipping: A communal approach to reducing urban traffic levels? Working paper*, Hamburg: Kühne Logistics University, 2016.
[26] A. Mourad, J. Puchinger, and C. Chu, *A survey of models and algorithms for optimizing shared mobility*, Transp. Res. B Methodol **123** (2019), 323–346.
[27] W. Qi, L. Li, S. Liu, and Z. J. M. Shen, *Shared mobility for last-mile delivery: Design, operational prescriptions, and environmental impact*, Manuf. Serv. Oper. Manag. **20** (2018), 737–751.
[28] A. Rais, F. Alvelos, and M. Carvalho, *New mixed integer-programming model for the pickup-and-delivery problem with transshipment*, Eur. J. Oper. Res. **235** (2014), 530–539.
[29] T. Raviv and E. Z. Tenzer, *Crowd-shipping of small parcels in a physical internet. Working paper*, Tel Aviv: Tel Aviv University, 2018.
[30] S. Ropke and D. Pisinger, *An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows*, Transp. Sci. **40** (2006), 455–472.
[31] J. F. Rouges and B. Montreuil, *Crowdsourcing delivery: New hyperconnected business models to reinvent delivery*, Proceedings of the 1st Physical Internet Conference IPIC, Québec, Canada, 2014.
[32] A. Sampaio Oliveira, M. Savelsbergh, L. Veelenturf, and T. van Woensel, *Crowd-based city logistics*, in *Sustainable transportation and smart logistics*, Elsevier, Amsterdam, Netherland, 2019, 381–400.

[33] A. Sampaio Oliveira, M. Savelsbergh, L. P. Veelenturf, and T. Van Woensel, *Delivery systems with crowd-sourced drivers: A pickup and delivery problem with transfers*, Networks **76** (2020), 232–255.
[34] M. M. Solomon, *Algorithms for the vehicle routing and scheduling problems with time window constraints*, Oper. Res. **35** (1987), 254–265.
[35] D. Soto Setzke, C. Pfluegler, M. Schreieck, S. Froehlich, M. Wiesche, and H. Krcmar, *Matching drivers and transportation requests in crowdsourced delivery systems*, Proceedings of the 23rd Americas Conference on Information Systems, Boston, MA, 2017.
[36] Y. Wang, D. Zhang, Q. Liu, F. Shen, and L. H. Lee, *Towards enhancing the last-mile delivery: An effective crowd-tasking model with scalable solutions*, Transp. Res. E Logist. Transp. Rev. **93** (2016), 279–293.

How to cite this article: Voigt S, Kuhn H. Crowdsourced logistics: The pickup and delivery problem with transshipments and occasional drivers. *Networks.* 2022;79:403–426. https://doi.org/10.1002/net.22045