An exact solution method for the capacitated item-sharing and crowdshipping problem

Moritz Behrend[a,*], Frank Meisel[a], Kjetil Fagerholt[b], Henrik Andersson[b]

[a] *School of Economics and Business, Kiel University, Germany*
[b] *Department of Industrial Economics and Technology Management, Norwegian University of Science and Technology, Trondheim, Norway*

Abstract

The item-sharing and crowdshipping problem combines two concepts of the sharing economy, namely item-sharing and crowdshipping. Item-sharing is about renting items among members of a sharing community. Crowdshipping addresses the transportation of these items through private people on trips they make anyway. The considered problem is to decide (1.) which request for an item to fulfill through which of the supplied items and (2.) who is doing the transport of rented items from the supply-locations to the request-locations. We generalize this problem with regard to crowdshippers' capacity, meaning that each crowdshipper can transport a given number of items along his/her intended route. This results in a detour routing problem, where crowdshippers are routed through intermediate locations on the way from their actual origin location to their intended destination. We propose an exact solution method based on a set packing formulation for which a label setting procedure generates feasible crowdshipper routes a priori. We also describe how to derive a heuristic from the exact approach. Our experiments identify to what extend higher capacities of crowdshippers lead to more profitable routes and under which conditions a heuristic reduction of the method is required to cope with the complexity of the problem. We also show that the new exact method clearly outperforms procedures that were developed earlier for a setting where each crowdshipper can transport at most one single item.

Keywords: Sharing Economy, Item-sharing, Crowdshipping, Capacity, Label Setting

June 3, 2019

1. Introduction

Item-sharing and crowdshipping are two concepts of the sharing economy. Item-sharing refers to a need-based exchange of requested items among members of a sharing platform like Erento (2019) or Zilok (2019). It offers consumers temporary access to items they do not own and it provides them with the economic advantage of pay-per-use instead of taking the financial risk of ownership. So far, it lies within the consumer's responsibility to get access to the requested item, meaning that he/she must pick up the desired item from the location where it is supplied and must bring it to the location where it is needed. A more service-oriented approach is if the platform performs the transportation of items so that the items are accessible directly at the requesting locations. However, this results in frequent transfers of the same items between (distant) consumer locations which is labor-intensive and time-consuming. In fact, the improved service of delivering items to consumers resembles the often addressed 'last-mile delivery' that is known to be very costly (Cleophas et al., 2019). Item-sharing therefore faces the dilemma of meeting consumer expectations of a fast delivery at low costs. Crowdshipping is a promising approach to provide such fast and cost-efficient deliveries (Archetti et al., 2016). Here, companies outsource delivery jobs to private drivers in return for a small compensation, see the platforms of Amazon Flex (2019), Postmates (2019), and Deliv (2019). Crowdshipping is particularly efficient if private drivers execute delivery jobs along their intended trips as this allows to utilize transportation capacity that exists anyway. For example, Nimber (2019) allows to filter for shipments that have the same origin and destination as the crowdshipper and Jade Zabiore (2019) additionally allows crowdshippers to announce their planned trips in order to receive delivery suggestions. Crowdshipping has recently received much attention from research because of its potential benefits but also because of the challenges of managing a large pool of private drivers (Archetti et al., 2016; Punel and Stathopoulos, 2017).

*corresponding author, tel.: +49 431 880-1532, address: Kiel University, School of Economics and Business, Olshausenstr. 40, 24098 Kiel, Germany

Email addresses: moritz.behrend@bwl.uni-kiel.de (Moritz Behrend), meisel@bwl.uni-kiel.de (Frank Meisel), kjetil.fagerholt@ntnu.no (Kjetil Fagerholt), henrik.andersson@ntnu.no (Henrik Andersson)

Using crowdshipping for fulfilling the transportation needs induced by item-sharing has been investigated recently by Behrend and Meisel (2018). They propose the idea of an integrated item-sharing and crowdshipping platform, where supplies of items, requests for items, and crowdshipping trips are all announced on the same platform. The platform then needs to decide which requests to satisfy using which supplies (so-called *supply-request* matching) and which crowdshippers to assign the job to execute the deliveries along their trips (so-called *trip-delivery* matching). The joint decision making allows to align the supply-request assignment with the available crowdshipper trips and, thus, to execute delivery jobs efficiently with only minor detours from the intended trips. The computational experiments showed that such a platform is more profitable and that it achieves a higher level of service compared with a pure item-sharing platform where consumers need to do the transportation all by themselves (so-called *self-sourcing*). Behrend and Meisel (2018) make a conservative assumption where each crowdshipper accepts at most one delivery job such that the above-mentioned matching decisions become easy-to-solve assignment problems. However, the willingness to execute delivery jobs is highly individual and private drivers may be interested in doing more than one delivery to increase their received compensation. In this paper, we therefore extend the problem investigated by Behrend and Meisel (2018) and consider crowdshippers that also accept more than one delivery job, i.e. that have a carrying *capacity* of more than one item. Larger capacities allow a platform to transfer more items through crowdshipping and, by that, to satisfy additional item-sharing requests. Furthermore, transportation costs may decrease if several delivery jobs can be executed by a same crowdshipper on one route with only little overall detour. However, if crowdshippers can get assigned several delivery jobs, the sharing platform has to solve routing problems instead of assignment problems.

The main contributions of this paper are summarized as follows:

- We generalize the item-sharing and crowdshipping problem with regard to crowdshippers' capacity. This requires to jointly address the assignment of supplies to requests and a corresponding detour-routing of crowdshippers, which is not supported by the previously proposed integrated item-sharing and crowdshipping approaches.

- We propose an exact solution approach for the resulting problem, which is based on a set packing problem and a label setting algorithm. A heuristic is derived by restricting the algorithm's search procedure.

- We conduct extensive experiments and show (i) that the new solution approach delivers exact solutions for the special case of capacity one in significantly less time in comparison to previous approaches, (ii) that the platform's profitability increases substantially with higher crowdshipping capacity, and (iii) that there is a considerable potential for incentivizing crowdshippers to accept multiple deliveries.

- We furthermore analyze by experiment to what extent the integration of item-sharing and crowdshipping on a single platform is useful. To this end, we compare different platform concepts and show how the profit of such platforms depends on the degree of collaboration.

The remainder of the paper is organized as follows. In Section 2, we point out the gap in the related literature. A detailed problem description and the set packing problem formulation is provided in Section 3. In Section 4, we describe the label setting algorithm to generate the columns for the set packing problem. The numerical experiments are the subject of Section 5. Section 6 concludes the paper.

2. Related literature

The two pillars of crowdshipping are private drivers that offer transportation services and senders that ask for non-professionals to fulfill tasks for shipping goods. Numerous empirical studies have shown that a majority of people interviewed are positive about getting involved in crowdshipping for various reasons (Barnes and Mattsson, 2016; Böcker and Meelen, 2017; Marcucci et al., 2017; Miller et al., 2017; Paloheimo et al., 2015). The most relevant reasons are to be (partially) compensated for travel related expenses, to serve the community, or to mitigate the environmental footprint of their individual traveling. The motivations of senders for using crowdshipping are analyzed in Punel and Stathopoulos (2017) and Punel

et al. (2018). A relevant driver is the expectation for a transportation at low cost whereas a strong inhibitor is the concern that an anonymous crowd cannot provide a reliable service. One approach to overcome this shortcoming is to build up trust in the service through tracing technology, direct communication to drivers, or peer-controlled systems that intend to prevent misconduct by publishing each driver's track record (Botsman and Rogers, 2011). Alternatively, Devari et al. (2017) propose to leverage social networks for assigning crowdshipping tasks such that friends or acquaintances conduct the transportation. Eventually, trust in crowdshipping arises from a good expectations management where liabilities and rights but also prospective rewards are clearly communicated between the involved parties, following well-known guidelines from reward management in behavioral research (Armstrong and Murlis, 2007). The operational planning is crucial in this regard to generate offers for senders that comply with crowdshippers' expected extent of involvement, thereby increasing the service's reliability. Our paper contributes to this by focusing on the operational planning with respect to crowdshippers' offered capabilities and compensation-based incentives, which is expected to motivate them to participate in such platforms.

The routing of crowdshippers that is part of the investigated problem links our research to the large body of literature on *pick-up and delivery problems* (Berbeglia et al., 2007, 2010; Parragh et al., 2008) and to *dial-a-ride problems* (Cordeau and Laporte, 2007; Ho et al., 2018). Despite close similarities between these problems and the planning context considered here, the key difference is that the decision making of the considered sharing platform does not involve creating new routes from the scratch. Instead, a *detour routing* is required that determines the sequence of visited pick-up and drop-off locations along existing routes with given origins and destinations with respect to the additional time a crowdshipper is willing to spend on transportation activities.

The only two papers related to detour routing for the transportation of goods seem to be the recent publications on crowdshipping by Arslan et al. (2019) and Chen et al. (2018). The planning problem of Arslan et al. (2019) is to assign a given set of delivery jobs to either crowdshippers or dedicated drivers. The authors' solution approach is to precompute possible routes and then solve a matching problem with side constraints to select those routes that

constitute an optimal solution. The subproblem of precomputing the routes is formulated as a traveling salesman problem with precedence constraints and time windows. It is solved with an exact recursive algorithm in which feasible crowdshipper-specific combinations of delivery jobs are successively extended by inserting new jobs. A heuristic is proposed, as well, where non-promising delivery jobs and routes are neglected at an early stage in the procedure. Crowdshippers' capacity is defined as their willingness for additional stops along their route, the so-called stop willingness. The results show that an increasing stop willingness leads to a more cost efficient execution of delivery jobs. Furthermore, less crowdshippers are involved in transportation activities as multiple jobs can be assigned to a same crowdshipper. The planning problem of Chen et al. (2018) is similar to that of Arslan et al. (2019). Its most salient additional aspect is that parcels can be transshipped between crowdshippers. To this end, the crowdshipping trips must be synchronized as parcels cannot be left unattended during the handover. The solution approach solves a network flow model to match crowdshippers to parcels as well as to obtain their corresponding paths and time schedules. The solution space is reduced by precomputing all edges along which a crowdshipper can travel between his/her origin and destination with respect to the accepted travel time increase due to deliveries. The authors also propose two heuristics, one heuristic with frequent feasibility checks for time compatibility and capacities, and a more efficient time expanded graph heuristic. The results suggest positive effects in terms of cost savings and reduced parcel-miles for a shipping company if crowdshippers offer higher capacities and if the operational planning can utilize this capacity. The integration of item-sharing, i.e. the assignment of supplies to requests, is neither considered in Arslan et al. (2019) nor in Chen et al. (2018).

Detour routing for the transportation of people on existing traffic flows is known from the ride-sharing literature, see Agatz et al. (2012) and Furuhata et al. (2013) for literature surveys. Related papers are those of Kamar and Horvitz (2009) and Herbawi and Weber (2012). Kamar and Horvitz (2009) consider a ride-sharing setting in which every agent can be either a driver that offers rides or a rider. Feasible ride-share matches are precomputed. A set covering problem is solved to determine those matches that lead to the highest cumulative objective value. Herbawi and Weber (2012) formulate the ride-sharing problem as

a network flow problem. Here, it is not required to precompute routes because constraints of the flow model ensure that picked-up passengers are taken to their destination with the same vehicle. The authors also propose a genetic algorithm to solve the model. The mentioned approaches allow for simultaneous transportation where a crowdshipper can execute multiple delivery jobs and a driver can carry multiple riders at a same time.

Further approaches in the context of crowdshipping and ride-sharing but without detour routing are investigated in a few papers. Kafle et al. (2017) investigate a two-echelon urban delivery and collection system in which crowdshippers perform the last-mile delivery of freight between relay points and customers, or vice versa. Each crowdshipper starts and ends its route at the same location. Crowdshipping routes are precomputed by enumerating all combinations of customers a crowdshipper can serve in one round trip and then solve the undirected traveling salesman problem for every such combination. An upper travel distance, a maximum number of visited customers, and a total parcel weight limit are respected. A two-echelon urban delivery system is also addressed by Qi et al. (2018). Here, an open vehicle routing problem formulation is used to describe the crowdshipping routes for the transportation of packages between relay points and customers. The authors solve this routing problem together with determining the size of the service zones in which crowdshippers perform the last-mile delivery, the locations of the relay points, and the route of the truck to supply the relay points with goods from a single depot. Furthermore, it is assumed that crowdshippers need to be attracted from the ride-share market so that the availability of shared mobility in the service zones is subject to the offered compensations. In Štiglic et al. (2015), riders with different origins and destinations are grouped and jointly picked up and dropped-off by drivers. To this end, the riders are assigned to so-called meeting point arcs each of which connects a common pick-up point with a common drop-off point. The riders are consequently expected to move from their origin to the pick-up point before their ride and also to move from the drop-off point to their destination afterwards. The problem is modeled as a weighted bipartite matching where the creation of rider combinations and the selection of appropriate meeting points is a precomputed input. Wang et al. (2016) consider a crowdshipping setting in which crowdshippers accept multiple delivery jobs irrespective of

the total route length that is required for their execution. As a consequence, the authors solve an assignment problem instead of a routing problem. The underlying assumptions are that the transportation distances are small due to nearby origin and destination locations and that the crowdshippers do the routing themselves.

The integration of crowdshipping with item-sharing on a single platform was first introduced by Behrend and Meisel (2018). Their results show that such a platform is more profitable and also provides a higher level of service compared with a pure item-sharing platform where consumers pick up items themselves without the support of crowdshippers. To support the operational planning of such an integrated platform, the authors formulate a three-dimensional assignment problem in which each crowdshipper can maximum be assigned one delivery job. Therefore, this formulation is only applicable to the special case with a crowdshipper capacity of one item, which does not require a detour route planning. It is, however, not applicable to the generalized problem addressed in this paper in which crowdshippers can accept multiple deliveries and therefore need to be routed.

We conclude from this literature review that capacities of crowdshippers have not been investigated in an integrated item-sharing and crowdshipping setting so far. However, we know from the literature on pure crowdshipping and pure ride-sharing that the matching rate and overall solution quality increases with such a problem generalization. When crowdshipping is integrated with item-sharing, an even better outcome is expected as supplies and requests can be matched such that the resulting delivery jobs are more likely to be accepted by crowdshippers. This paper therefore proposes a corresponding problem formulation and a new solution method that is based on a label setting algorithm and a set packing problem for solving the generalized item-sharing and crowdshipping problem where crowdshippers have given capacities.

3. The capacitated item-sharing and crowdshipping problem

3.1. Problem definition

The considered problem addresses the operational planning of an integrated platform that consolidates information from an item-sharing platform and a crowdshipping platform. To

this end, we assume that members of an item-sharing community announce their supplies of items and their requests for items to the same platform to which members of a crowdshipping community announce their planned trips to in order to receive delivery jobs. The platform collects the incoming announcements over one period. It then (1.) assigns the supplied items to compatible requests and (2.) coordinates the transfer of items between locations through either consumers or crowdshippers before it responds the announcements at the beginning of the next period. This requires that users of the platform need to wait for a response but, eventually, improves their experience with it since more supplies, requests, and trips can be matched. Consequently, the platform does not need to respond in real-time but, of course, a fast, scalable, and high quality decision making is needed for operating the platform. We further assume that the benefit of a need-based item access is closely linked to the effort that is required to eventually use an item. Consumers are therefore service-oriented. They clearly prefer having their requested items delivered to their homes compared with sourcing them themselves from the supply location, despite a small extra charge for the increased service. The exact timing of an item handover is organized by the involved parties and implies a certain degree of flexibility with regard to time. This is even more so as tight delivery schedules would put a burden on private drivers and may deter them from participating in crowdshipping in the first place. The lack of urgency is essential for item-sharing just as it is for all other concepts of the sharing economy.

For a formal description, let $Sply$ denote the set of supplied items. Every item $i \in Sply$ is associated with its current location. Let Req denote the set of requests for items. Every request $j \in Req$ is associated with the location at which the corresponding consumer wants the item to be delivered to. Despite the wide variety of different product types that are usually exchanged between consumers on such platforms, we consider full compatibility between supplies and requests here, meaning that every supply can satisfy every request. We hereby address the most challenging but also the most profitable scenario as it provides the operator of a platform maximum flexibility in assigning supplies to requests. Heterogeneity of items could easily be incorporated into our models and algorithms but is omitted here for reasons of simplicity. Let K denote the set of all trips announced by crowdshippers. Every

trip $k \in K$ is defined by its origin $o(k)$, its destination $d(k)$, and the travel time t_k to get from $o(k)$ to $d(k)$ on the fastest route. Note that we use the term 'crowdshipper k' to refer to the crowdshipper on trip k. Likewise, we use the term 'consumer j' to refer to the consumer behind request j. Contrasting Behrend and Meisel (2018), crowdshippers agree to execute more than one delivery job in the setting considered here. We denote by ζ the capacity of each single crowdshipper. It corresponds to the number of requests that can maximum be served per trip. We assume that all crowdshippers offer the same capacity ζ.

We consider three transfer modes *self-sourcing*, *home delivery*, and *neighborhood delivery* to transport supplied items from their current locations to their assigned request locations. Self-sourcing means that a consumer $j \in Req$ picks up an item $i \in Sply$ from its current location by himself/herself. A home delivery and a neighborhood delivery involve a crowdshipper $k \in K$ who detours from the direct route between $o(k)$ and $d(k)$ to execute delivery jobs. In a home delivery, crowdshipper k picks up an item at its supply location and delivers it directly to the assigned request location j. In a neighborhood delivery, crowdshipper k drops off previously picked-up items at his/her own destination $d(k)$, from where they are self-sourced by the assigned consumers.

We demonstrate the three transfer modes using the exemplary problem setting in Figure 1a. The example is composed of three supplied items $Sply = \{i_1, i_2, i_3\}$ (symbols $\oplus$), four requests $Req = \{j_1, j_2, j_3, j_4\}$ (symbols $\ominus$), and two trips $K = \{k_1, k_2\}$ (connected rectangles). Figure 1b provides an example for self-sourcing. The platform assigns item i_2 to request j_1 and consumer j_1 picks up the item himself/herself in a round trip (route j_1-i_2-j_1). Figures 1c and 1d provide examples for home deliveries. In Figure 1c, the platform assigns i_2 to j_1 and it assigns crowdshipper k_1 the task to execute the resulting delivery job. The crowdshipper then travels along the route $o(k_1)$-i_2-j_1-$d(k_1)$ to pick up the item at i_2, does a home delivery at the location of request j_1, and then goes to the destination $d(k_1)$. Assuming a crowdshipping capacity $\zeta = 1$ in this example solution, k_1 can only execute one delivery job. In contrast, if the crowdshipping capacity is $\zeta = 2$, the platform can assign k_1 the task to execute two delivery jobs, see Figure 1d. The route $o(k_1)$-i_1-j_3-i_2-j_1-$d(k_1)$ depicted in Figure 1d expresses one out of multiple routes that could be carried out for fulfilling the assigned tasks. For example,

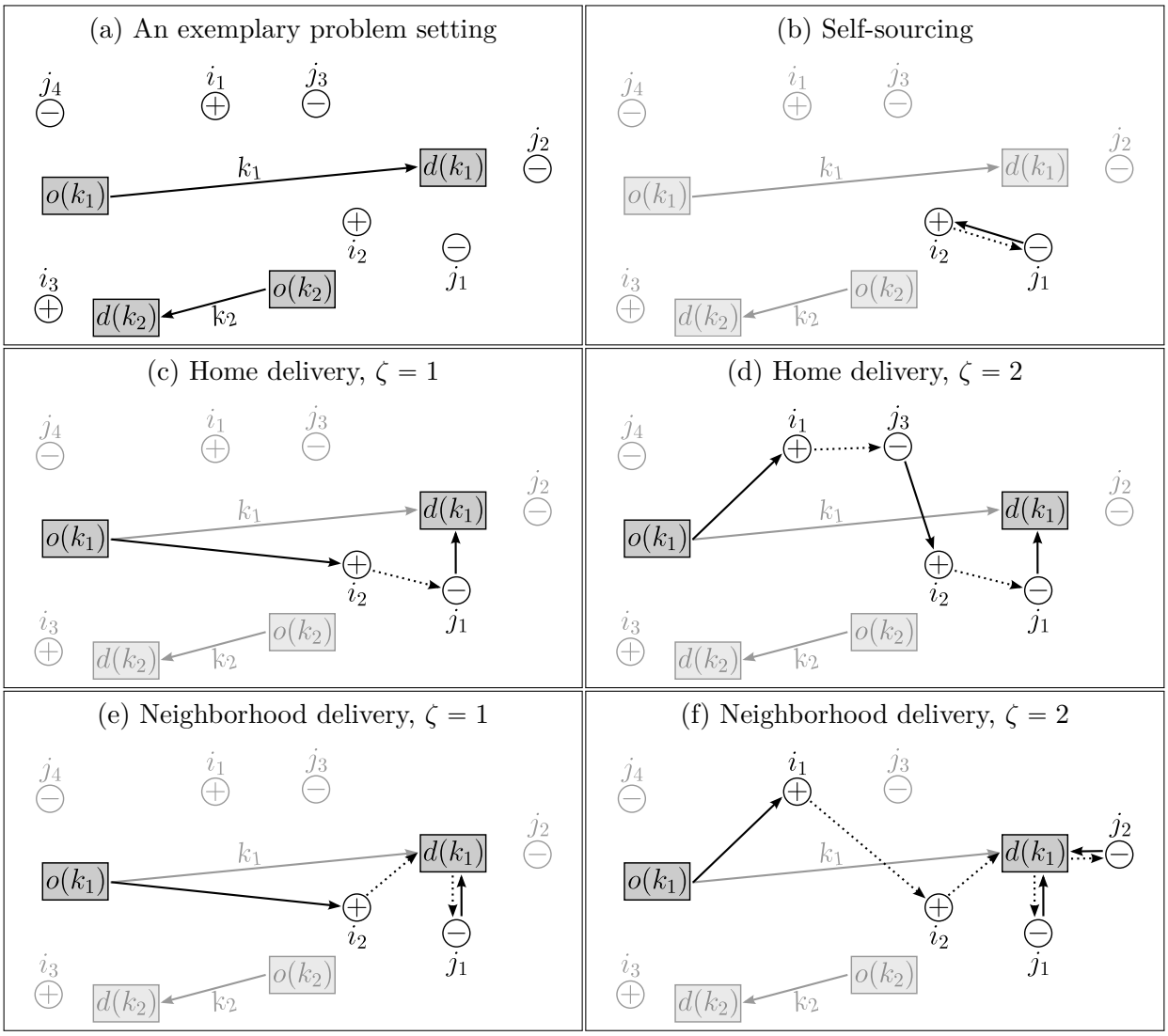


Figure 1: Illustration of the three transfer modes for different capacities ($\zeta = 1$ and $\zeta = 2$).

picking up both items first and then dropping them off at their assigned locations (route $o(k_1)$-i_1-i_2-j_3-j_1-$d(k_1)$) is conceivable too. Figures 1e and 1f provide examples for neighborhood deliveries with a capacity of $\zeta = 1$ and $\zeta = 2$, respectively. Here, crowdshipper k_1 picks up the assigned items, either item i_2 (Figure 1e) or items i_1 and i_2 (Figure 1f), and takes them to destination $d(k_1)$ from where the respective consumers self-source them. Thus, the crowdshipper and the consumers collaboratively coordinate the transport. Again, in the case of $\zeta = 2$ alternative crowdshipping routes exist. In contrast to the depicted crowdshipping route $o(k_1)$-i_1-i_2-$d(k_1)$, item i_2 could also be picked up before item i_1. Note that a crowd-

shipper with $\zeta > 1$ can also execute a mix of home deliveries and neighborhood deliveries.

Executing delivery jobs asks for some flexibility from the community members to spend extra time on traveling. For this, let t_{pq} be the travel time of the direct route between two locations $p, q \in \mathit{Sply} \cup \mathit{Req} \cup \{o(k) \mid k \in K\} \cup \{d(k) \mid k \in K\}$. The maximum time a consumer $j \in \mathit{Req}$ accepts to spend on self-sourcing an item is expressed by the *self-sourcing flexibility* f_j^{ssrc}. Thus, a request j can be satisfied through self-sourcing an item $i \in \mathit{Sply}$ only if the round trip travel time is less or equal to the consumer's self-sourcing flexibility, i.e. if $t_{ji} + t_{ij} \leqslant f_j^{\text{ssrc}}$. Likewise, request j can only be satisfied through a neighborhood delivery if picking up an item at $d(k)$ is within this flexibility, i.e. if $t_{j,d(k)} + t_{d(k),j} \leqslant f_j^{\text{ssrc}}$. The maximum additional travel time by which a crowdshipper k accepts to extend the travel time t_k of his/her intended trip is called *detour flexibility* f_k^{dtr}. Detouring for picking up and dropping off items is accepted as long as the overall travel time does not exceed $t_k + f_k^{\text{dtr}}$. The self-sourcing flexibility of consumers and the detour flexibility of crowdshippers is submitted to the platform along with their announcements.

The platform receives revenues for satisfying requests. Let r^{ssrc} denote the revenue if the consumer behind a request needs to self-source an assigned item, be it a direct self-sourcing from a supply location $i \in \mathit{Sply}$ or a collaborative self-sourcing from a location $d(k)$ in a neighborhood delivery. In contrast, a higher service is provided by a home delivery if the requested item is delivered directly to the location where it is needed. We denote the corresponding revenue for a successful home delivery by r^{home} with $r^{\text{home}} \geqslant r^{\text{ssrc}}$. The platform's costs arise from compensating crowdshippers for their detouring. We denote by c^{dtr} the compensation rate per time unit a crowdshipper k travels in excess of t_k. Note that these parameters can be individualized in order to reflect specific preferences of consumers and/or crowdshippers. For example, a price-sensitive consumer j_1 who is not willing to pay extra for a home delivery could be modeled by parameters $r_1^{\text{ssrc}} = r_1^{\text{home}}$. On the contrary, a service sensitive consumer j_2 may not be interested in self-sourcing at all, i.e. $r_2^{\text{ssrc}} = -\infty$. Such individual revenue and cost parameters are, however, not considered in the following for reasons of simplicity. Eventually, the revenue of a platform reflects the number of satisfied consumers and the service provided. Conversely, the cost of a platform reflects the required

Table 1: Notation used to model the integrated item-sharing and crowdshipping problem.

Sets:	
$Sply$	Set of supplied items
Req	Set of requests for items
K	Set of trips of crowdshippers
Parameters:	
$o(k)$	Origin of crowdshipper k
$d(k)$	Destination of crowdshipper k
t_{pq}	Travel time on the direct route between locations p and q
ζ	Capacity per trip
f_j^{ssrc}	Self-sourcing flexibility of consumer j
f_k^{dtr}	Detour flexibility of crowdshipper k
r^{ssrc}	Revenue if self-sourcing is involved
r^{home}	Revenue for a home delivery
c^{dtr}	Compensation rate for crowdshippers

detouring of crowdshippers. Since a high consumer satisfaction and an efficient transfer of items are both goals to strive for, the platform's objective is to maximize profit.

To summarize, the planning problem involves simultaneously assigning supplies to requests to trips together with the routing for crowdshippers who execute multiple delivery jobs. Within the problem, it is also decided on the transfer mode for each supply-request assignment, respecting the detour flexibility of crowdshippers and the self-sourcing flexibility of consumers. A summary of the notation used is provided in Table 1.

3.2. Model formulation

While Behrend and Meisel (2018) formulated the integrated item-sharing and crowdshipping problem as an assignment problem (due to the crowdshipper capacity of one), we propose here a new formulation that can handle the capacitated version of this problem ($\zeta > 1$) with respect to the detour routing for crowdshippers. More precisely, we formulate the integrated item-sharing and crowdshipping problem as a set packing problem, where a *column* defines the set of items that are picked up, the set of requests that are satisfied, and the crowdshipper that does the transportation of these items. We associate a weight with each column that corresponds to the marginal profit of the platform if this column is selected as part of the overall solution. The columns and the associated profits are generated in a preprocessing phase. Eventually, we solve a weighted set packing problem to select a subset of columns

that maximizes a platform's profit. The whole solution procedure that is founded on the set packing problem formulation is denoted by [**spp**] in the following.

The columns that are generated in the preprocessing phase form the set $\mathcal{C}$. Let the binary parameter α_{ic} indicate whether supplied item $i \in Sply$ is assigned to a request in column $c \in \mathcal{C}$. Accordingly, let binary parameter β_{jc} indicate whether request $j \in Req$ is satisfied in column c and let binary parameter γ_{kc} indicate whether or not crowdshipper $k \in K$ carries out the transportation in column c. The parameters are defined as

$$\alpha_{ic} = \begin{cases} 1, \text{ if item } i \text{ is assigned to any request in } c \\ 0, \text{ else} \end{cases} \qquad \forall i \in Sply, c \in \mathcal{C}, \quad (1)$$

$$\beta_{jc} = \begin{cases} 1, \text{ if request } j \text{ is satisfied in } c \\ 0, \text{ else} \end{cases} \qquad \forall j \in Req, c \in \mathcal{C}, \quad (2)$$

$$\gamma_{kc} = \begin{cases} 1, \text{ if crowdshipper } k \text{ carries out the transportation in } c \\ 0, \text{ else} \end{cases} \qquad \forall k \in K, c \in \mathcal{C}. \quad (3)$$

We refer to Figure 1d for an example. Since i_1 and i_2 are assigned to a request whereas i_3 is not, the corresponding α values of a column c are $\alpha_{1c} = 1$, $\alpha_{2c} = 1$, and $\alpha_{3c} = 0$. Moreover, requests j_1 and j_3 are satisfied ($\beta_{1c} = \beta_{3c} = 1,\ \beta_{2c} = \beta_{4c} = 0$) and the transportation is carried out by crowdshipper k_1 ($\gamma_{1c} = 1,\ \gamma_{2c} = 0$).

The profit associated with a column c is denoted p_c. It is composed of the collected revenue that reflects in which transfer modes the requests are satisfied minus the costs that arise from compensating the crowdshipper for his/her detouring. Since both the transfer modes and the crowdshipper's route cannot be inferred from the parameters α, β, and γ, profit p_c contains extra information that is computed at the time the column is created.

Let the binary variable x_c indicate whether column $c \in \mathcal{C}$ is part of the overall solution ($x_c = 1$) or not ($x_c = 0$). The weighted set packing problem then is

$$max \rightarrow P = \sum_{c \in \mathcal{C}} p_c \cdot x_c \quad (4)$$

subject to

$$\sum_{c \in \mathcal{C}} \alpha_{ic} \cdot x_c \leqslant 1 \qquad \forall i \in Sply \tag{5}$$

$$\sum_{c \in \mathcal{C}} \beta_{jc} \cdot x_c \leqslant 1 \qquad \forall j \in Req \tag{6}$$

$$\sum_{c \in \mathcal{C}} \gamma_{kc} \cdot x_c \leqslant 1 \qquad \forall k \in K \tag{7}$$

$$x_c \in \{0, 1\} \quad \forall c \in \mathcal{C}. \tag{8}$$

Objective (4) expresses the profit maximization of the platform. Constraints (5), (6), and (7) ensure that the same item, the same request, or the same crowdshipper cannot appear in more than one selected column of the final solution, respectively. The domain of the decision variables is defined in Constraints (8).

3.3. Generating columns for self-sourcing

Generating those columns in the set packing problem that respect self-sourcing opportunities is straight forward as there is only one supply and one request per column and no crowdshipper. Let A be the set of all supply-request matchings that are feasible through self-sourcing, i.e.

$$A = \{(i, j) \in Sply \times Req \mid t_{ji} + t_{ij} \leqslant f_j^{\text{ssrc}}\}. \tag{9}$$

Every element in A is a feasible self-sourcing option where the consumer behind request j picks up the assigned item i himself/herself. We therefore create for every tuple $(i, j) \in A$ a column c with $\alpha_{ic} = 1$, $\beta_{jc} = 1$, a profit $p_c = r^{\text{ssrc}}$, and all other parameters in that column being set to zero, i.e.

$$\alpha_{i'c} = 0 \quad \forall i' \in Sply, i' \neq i, \tag{10}$$

$$\beta_{j'c} = 0 \quad \forall j' \in Req, j' \neq j, \text{ and} \tag{11}$$

$$\gamma_{kc} = 0 \quad \forall k \in K. \tag{12}$$

Generating the crowdshipping columns in the set packing problem is less straight forward. On the one hand, multiple requests can be involved in the same crowdshipping trip. This requires solving a detour routing problem to determine the compensation, respectively profit p_c, that results from this route subject to detour flexibility constraints. On the other hand, crowdshippers can serve requests either through a home delivery or a neighborhood

delivery. The service is selected for each request individually and it affects a column's profit as both revenue and a crowdshipper's compensation are subject to it. Note that it cannot be determined in a preprocessing phase whether a home delivery or a neighborhood delivery is more profitable for a particular request as it depends on the location of available supplies and the crowdshipping route itself. Therefore, we discuss the procedure to generate crowdshipping columns in the separate Section 4.

4. A label setting algorithm for routing crowdshippers

This section introduces a label setting algorithm that generates crowdshipping columns for the set packing problem. The labels (respectively paths) are generated such that they respect the limited detour flexibilities of crowdshippers. For a thorough description of shortest path problems with resource constraints, see Irnich and Desaulniers (2005). In the following, we first define a *label* and its data, explain how to extend it and how labels can be dominated, and then provide an outline of the algorithm. Since a label is not equivalent to a column in the set packing problem, a post-processing is required to convert labels into columns. The description of the post-processing follows in a separate subsection. The combination of label generation and post-processing is then demonstrated through an illustrative example. Finally, we derive heuristic solution approaches from the exact method.

4.1. Generating labels

The label setting algorithm creates labels for each crowdshipper $k \in K$, one-by-one. Let l be a label of k. A label describes one possible crowdshipping route that starts at $o(k)$ and ends at $d(k)$, and for each label, the following data is stored. The travel time of the route is denoted τ_l. All locations that are visited along the route form the set $\mathcal{V}_l$ and the route's penultimate location, i.e. the location that is visited right before $d(k)$, is denoted δ_l. Furthermore, ϑ_l describes the number of items that crowdshipper k carries to his/her destination $d(k)$ to satisfy requests through a neighborhood delivery. The used notation is summarized in Table 2. For a label l that describes the route of crowdshipper k_1 in Figure 1f, we have $\mathcal{V}_l = \{o(k_1), i_1, i_2, d(k_1)\}$, $\delta_l = i_2$, and $\vartheta_l = 2$.

Table 2: Notation for defining a label l.

τ_l	Travel time of that route
$\mathcal{V}_l$	Set of locations visited on that route
δ_l	Penultimate location on that route
ϑ_l	Number of items the crowdshipper takes to its destination

Since the algorithm is an iterative procedure that creates new labels by extending existing ones, we observe predecessor-successor relations between labels. For the description of the procedure, let l^- denote the unique predecessor label of a label l and let l^+ denote one out of many successor labels of label l. A new label l^+ is created from a label l by inserting exactly one additional location after the current penultimate position into l's route, i.e. right before $d(k)$. Label extensions are valid if the detouring required to visit the newly inserted penultimate location does not violate crowdshipper k's detour flexibility. If the newly inserted location is a supply location, a further criterion for validity is whether crowdshipper k's capacity allows for another pick-up. Those supply locations that are eligible for extending label l form the set $\mathcal{S}_l^{\text{open}}$. The set is defined by

$$\mathcal{S}_l^{\text{open}} = \{i \in Sply \backslash \mathcal{V}_l \mid \tau_l - t_{\delta_l,d(k)} + t_{\delta_l,i} + t_{i,d(k)} \leqslant t_k + f_k^{\text{dtr}}\}. \tag{13}$$

The set of eligible request locations is denoted $\mathcal{R}_l^{\text{open}}$ and defined by

$$\mathcal{R}_l^{\text{open}} = \{j \in Req \backslash \mathcal{V}_l \mid \tau_l - t_{\delta_l,d(k)} + t_{\delta_l,j} + t_{j,d(k)} \leqslant t_k + f_k^{\text{dtr}}\}. \tag{14}$$

Extending a label l by a supply location increases the value of ϑ_l by one and extending it by a request location decreases the value by one. More formally,

$$\vartheta_{l^+} = \begin{cases} \vartheta_l + 1, \text{ if } \delta_{l^+} \in \mathcal{S}_l^{\text{open}} \\ \vartheta_l - 1, \text{ if } \delta_{l^+} \in \mathcal{R}_l^{\text{open}} \end{cases}. \tag{15}$$

Referring to the example in Figure 2, label l^- (route $o(k_1) - i_1 - d(k_1)$, dashed line) is the predecessor label of label l (route $o(k_1) - i_1 - i_2 - d(k_1)$, solid line), from which l was derived by inserting i_2 after the penultimate position of l^-. In contrast, label l^+ (route $o(k_1) - i_1 - i_2 - j_3 - d(k_1)$, dotted line) is a successor label of label l as it extends l's route by the new penultimate location j_3. On the route described by l^-, the crowdshipper k_1 picks up one item without delivering it to a request location ($\vartheta_{l^-} = 1$). Since l extends l^- by another

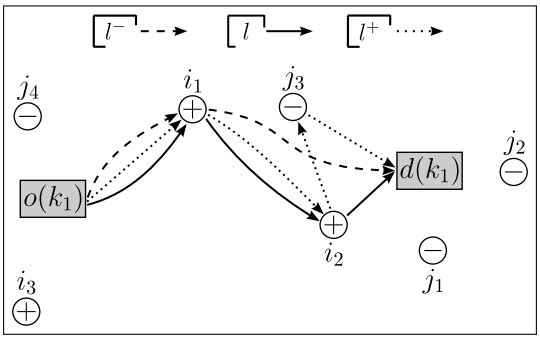


Figure 2: Alternative crowdshipping routes of k_1, represented by the labels l^-, l, and l^+.

supply location, two items are delivered to $d(k)$, i.e. $\vartheta_l = 2$. Finally, since label l^+ extends l by the request location j_3, the crowdshipper drops off an item there so that $\vartheta_{l^+} = 1$ again.

The data of the extended label l^+ is computed through the following resource extension functions. The total travel time of a route is propagated from one label to another by

$$\tau_{l^+} = \tau_l - t_{\delta_l, d(k)} + t_{\delta_l, \delta_{l^+}} + t_{\delta_{l^+}, d(k)}. \tag{16}$$

The set of visited locations $\mathcal{V}_{l^+}$ is composed of the newly inserted penultimate location δ_{l^+} and the set of locations already visited in l, i.e.

$$\mathcal{V}_{l^+} = \delta_{l^+} \cup \mathcal{V}_l \ . \tag{17}$$

The penultimate location, the set of visited vertices, and a label's travel time are used to express a dominance relation between any two labels l and l' by

$$l \succ l' \Leftrightarrow \delta_l = \delta_{l'} \wedge \mathcal{V}_l = \mathcal{V}_{l'} \wedge \tau_l < \tau_{l'}. \tag{18}$$

Referring to Equation (18), l dominates l' if the route of l has the same penultimate location and also includes the same locations as the route of l' but its travel time τ_l is less than the travel time $\tau_{l'}$. In other words, a crowdshipper visits the same locations on both routes and both routes can be extended in the same way but the visiting sequence of l results in less detouring. The platform therefore yields higher profits by choosing the crowdshipping route l over l' as the same revenue is obtained but less compensation is required.

4.2. Outline of the procedure

The label setting algorithm is outlined in Algorithm 1. It takes a problem instance defined by the sets *Sply*, *Req*, and K as an input, computes a set of non-dominated labels $\mathcal{L}$, and, finally,

Algorithm 1: The label setting algorithm for routing crowdshippers.

Input: A problem instance defined by $Sply$, Req, and K;
Output: A set $\mathcal{C}^{\text{crowd}}$ of crowdshipping columns for the set packing problem;

$\mathcal{L} \leftarrow \{\}$; `// set of non-dominated labels generated by the procedure`
for $k \in K$ **do**
  initialize label l_0 ; `// initial label`
  $\tau_{l_0} = t_k$;
  $\mathcal{V}_{l_0} = \{o(k), d(k)\}$;
  $\delta_{l_0} = o(k)$;
  $\vartheta_{l_0} = 0$;
  $\mathcal{L}^+ \leftarrow \{l_0\}$;
  while $\mathcal{L}^+ \neq \varnothing$ **do**
    $l \leftarrow l \in \mathcal{L}^+$;
    $\mathcal{L}^+ \leftarrow \mathcal{L}^+ \backslash \{l\}$;
    if $\nexists l' \in \mathcal{L}, l' \succ l$ **then** `// extend` l `only if it is non-dominated`
      $\mathcal{L} \leftarrow \{l\} \cup (\mathcal{L} \backslash \{l' \in \mathcal{L} \mid l' \prec l\})$; `// add` l `and remove dominated` l'
      `// update sets of eligible locations for a route extension`
      $\mathcal{S}_l^{\text{open}} \leftarrow \{i \in Sply \backslash \mathcal{V}_l \mid \tau_l - t_{\delta_l, d(k)} + t_{\delta_l, i} + t_{i, d(k)} \leqslant t_k + f_k^{\text{dtr}}\}$;
      $\mathcal{R}_l^{\text{open}} \leftarrow \{j \in Req \backslash \mathcal{V}_l \mid \tau_L - t_{\delta_l, d(k)} + t_{\delta_l, j} + t_{j, d(k)} \leqslant t_k + f_k^{\text{dtr}}\}$;
      if $|\mathcal{V}_l \cap Sply| < \zeta$ **then** `// add supply location`
        foreach $i \in \mathcal{S}_l^{\text{open}}$ **do**
          initialize label l^+ ; `// new label`
          $\tau_{l^+} = \tau_l - t_{\delta_l, d(k)} + t_{\delta_l, i} + t_{i, d(k)}$;
          $\mathcal{V}_{l^+} \leftarrow \{i\} \cup \mathcal{V}_l$;
          $\delta_{l^+} = i$;
          $\vartheta_{l^+} = \vartheta_l + 1$;
          $\mathcal{L}^+ \leftarrow \{l^+\} \cup \mathcal{L}^+$;
      if $\vartheta_l > 0$ **then** `// add request location`
        foreach $j \in \mathcal{R}_l^{\text{open}}$ **do**
          initialize label l^+ ; `// new label`
          $\tau_{l^+} = \tau_l - t_{\delta_l, d(k)} + t_{\delta_l, j} + t_{j, d(k)}$;
          $\mathcal{V}_{l^+} \leftarrow \{j\} \cup \mathcal{V}_l$;
          $\delta_{l^+} = j$;
          $\vartheta_{l^+} = \vartheta_l - 1$;
          $\mathcal{L}^+ \leftarrow \{l^+\} \cup \mathcal{L}^+$;
$\mathcal{C}^{\text{crowd}} \leftarrow$ post-processing $(\mathcal{L})$;
return $\mathcal{C}^{\text{crowd}}$;

derives a set of crowdshipping columns $\mathcal{C}^{\text{crowd}}$ for the set packing problem. The algorithm is an iterative procedure in which we create and extend labels for each crowdshipper, one-by-one. Newly created labels temporarily form the set $\mathcal{L}^+$ of successor labels until they are extended themselves. Only non-dominated labels from set $\mathcal{L}^+$ are taken up in set $\mathcal{L}$. Having created labels for all crowdshippers, the procedure terminates and a post-processing translates the labels in $\mathcal{L}$ into columns in $\mathcal{C}^{\text{crowd}}$ for the set packing problem. This step is afterwards described in Subsection 4.3.

The initial label l_0 of a crowdshipper k corresponds to the direct route from the origin $o(k)$ to the destination $d(k)$. Its travel time is t_k, the visited vertices are $o(k)$ and $d(k)$, its penultimate location is $o(k)$, and the crowdshipper carries no items (see lines 3–7). So far, it is the only label in the temporary set $\mathcal{L}^+$, meaning that the label is yet to be extended, but further are about to follow. Each of these labels then passes through the procedure described below. It is first removed from the temporary set $\mathcal{L}^+$ (line 11) and must pass the dominance test (line 12). If the test is passed, there is no other label $l' \in \mathcal{L}$ that allows to satisfy the same requests with the same supplies more efficiently. We therefore include l in the set $\mathcal{L}$ and remove all those labels from set $\mathcal{L}$ that are in turn dominated by l (line 13). If l does not pass the dominance test, we discard l and move on with the next label.

We next identify the supply and request locations that can be used for extending l, see lines 14 and 15 of Algorithm 1. Referring to Equations (13) and (14), only those locations are considered eligible for which a route extension does not exceed k's detour flexibility.

The extension of l by an additional supply location is done in lines 17–24 of Algorithm 1. Such an extension is done only if the crowdshipper's capacity allows for another pick-up. If this holds, we create one new label l^+ per reachable supply location. To this end, we propagate the travel time, update the set of visited locations, define the penultimate location of the new route, and increase the number of carried items by one. Afterwards, the new label l^+ is included in the set $\mathcal{L}^+$. Extending l by a request location is done likewise (see lines 25–32) under the condition that the crowdshipper already carries at least one item. We then create one new label l^+ per reachable request location. Since the crowdshipper performs a home-delivery at the visited request location, he/she drops-off one item there and the num-

ber of carried items decreases by one. Eventually, the algorithm calls the post-processing that converts labels in $\mathcal{L}$ into columns in $\mathcal{C}^{\text{crowd}}$ as described in the next subsection.

4.3. Deriving columns from labels in a post-processing

A label is an incomplete partial solution since we only determine a route at this moment but we do not decide which picked-up item is assigned to which request. More precisely, if the crowdshipper visits a request location, we decrease the quantity of ϑ_l by one but it is not decided which of the carried (homogeneous) items he/she drops off in particular. Moreover, if a crowdshipper carries items to $d(k)$ for a neighborhood delivery ($\vartheta_l > 0$), we cannot infer from the label which requests these items finally fulfill. For example, it is not clear for label l^+ in Figure 2 which of the picked-up items i_1 and i_2 is dropped off at j_3 and which is taken to $d(k_1)$. It is also not decided which of the yet unsatisfied requests j_1 and j_2 is assigned the item that is brought to $d(k_1)$ if both consumers can self-source an item from $d(k_1)$. To resolve this ambiguity, we derive (multiple) columns from a label l. We distinguish two cases: $\vartheta_l = 0$, i.e. the crowdshipper home-delivers all items along the route, and $\vartheta_l > 0$, i.e. the crowdshipper takes at least one item to $d(k)$ for a neighborhood delivery.

In case of $\vartheta_l = 0$, all picked-up items are dropped-off along the route. We can infer any supply-request matching from the routing since we consider full compatibility between supplies and requests and we know from the visited locations which supplies are picked up and which requests are satisfied. Therefore, for every label $l \in \{\mathcal{L} \mid \vartheta_l = 0\}$ we create one column c whose parameters are defined as follows:

$$\alpha_{ic} = \begin{cases} 1, \text{ if } i \in \mathcal{V}_l \\ 0, \text{ else} \end{cases} \qquad \forall i \in Sply, \tag{19}$$

$$\beta_{jc} = \begin{cases} 1, \text{ if } j \in \mathcal{V}_l \\ 0, \text{ else} \end{cases} \qquad \forall j \in Req, \tag{20}$$

$$\gamma_{kc} = \begin{cases} 1, \text{ if } o(k), d(k) \in \mathcal{V}_l \\ 0, \text{ else} \end{cases} \qquad \forall k \in K. \tag{21}$$

An item $i \in Sply$ is assigned to a request and a request $j \in Req$ is satisfied in column c if the associated locations are visited. A crowdshipper $k \in K$ carries out the transportation if it is the same crowdshipper for which label l was created for.

In case of $\vartheta_l > 0$, a matching is required where we assign the available items at $d(k)$ to the requests that can be satisfied through a neighborhood delivery. To this end, let $\mathcal{N}_l^{\text{open}}$ be the set of yet unsatisfied requests whose consumers can self-source the items from $d(k)$, i.e

$$\mathcal{N}_l^{\text{open}} = \{j \in Req \backslash \mathcal{V}_l \mid t_{j,d(k)} + t_{d(k),j} \leqslant f_j^{\text{ssrc}}\}. \tag{22}$$

According to Equation (22), we consider only those requests $Req\backslash\mathcal{V}_l$ whose locations have not been visited so far and whose consumers can reach the destination of a trip within their self-sourcing flexibility. The matching is conducted by creating all possible subsets of size ϑ_l from set $\mathcal{N}_l^{\text{open}}$. Each such subset indicates one option for distributing the ϑ_l items to consumers in $\mathcal{N}_l^{\text{open}}$. Note that the cardinality of the subsets must be exactly ϑ_l so that the resulting column is neither invalid (more requests are satisfied than items are available) nor inefficient (less requests are satisfied than items are available). The post-processing enumerates all these subsets to cover all assignment alternatives and then creates a column for the set packing problem for each of them. The number of combinations is equal to the binomial coefficient

$$\binom{|\mathcal{N}_l^{\text{open}}|}{\vartheta_l} = \frac{|\mathcal{N}_l^{\text{open}}|!}{(|\mathcal{N}_l^{\text{open}}| - \vartheta_l)! \cdot \vartheta_l!}\ . \tag{23}$$

More formally, let M_l denote the resulting set of subsets that are derived from a label l. We then create a new column c for every subset $m \in M_l$. Through this column, all those requests are satisfied whose locations are visited along l's route (home deliveries) or which are assigned an item at $d(k)$ and are therefore an element in m (neighborhood deliveries), i.e.

$$\beta_{jc} = \begin{cases} 1, \text{ if } j \in \mathcal{V}_l \cup m \\ \\ 0, \text{ else} \end{cases} \qquad \forall j \in Req. \tag{24}$$

The parameters α_{ic} and γ_{ic} are identical for all columns derived from a same label and set according to (19) and (21).

The platform's profit from choosing column c can then be computed as

$$p_c = |\mathcal{V}_l \cap Req| \cdot r^{\text{home}} + \vartheta_l \cdot r^{\text{ssrc}} - (\tau_l - t_k) \cdot c^{\text{dtr}} \tag{25}$$

where all requests visited en route indicate successful home deliveries with revenue r^{home}, all items carried to $d(k)$ result in successful neighborhood deliveries with revenue r^{ssrc}, and the crowdshipper's compensation corresponds to his/her detour travel time.

Note that the separation of label generation by the label setting algorithm and column generation by the post-processing improves the performance of the overall solution process as the computationally challenging part of extending labels in the algorithm is reduced without loss of optimality. This is because the algorithm can ignore the explicit supply-request matching which means that it does not have to extend equally profitable labels that involve the same supplies, the same requests, and the same crowdshipper. It further reduces the number of created labels by postponing the decision on which requests to satisfy through a neighborhood delivery. This is possible since for each column that is derived from a same label, the same supplies are picked up (α_{ic}) and the same crowdshipper executes the job (γ_{kc}). Even the profit is the same (p_c) as the crowdshipper takes the same route under each such partial solution (yielding the same revenue for home deliveries and the same compensation for the crowdshipper) and the revenue from each of the successful neighborhood deliveries is a constant r^{ssrc}. Only the requests that are satisfied (β_{jc}) differ from one another, which can be resolved through post-processing as described above.

4.4. An illustrative example

The procedure of the label setting algorithm and the corresponding post-processing of labels into columns are illustrated in Figure 3, based on the exemplary problem instance in Figure 1a. We focus in this example on crowdshipper k_1. The crowdshipping routes for crowdshipper k_2 are created accordingly. Note that the locations i_3 and j_4 are ignored here to keep the example brief. Also, it is assumed that there is sufficient detour flexibility to visit all locations except for request j_2 and that the self-sourcing flexibilities of the requests j_1 and j_2 allow for neighborhood deliveries.

The algorithm starts with the initial label l_0 (see Figure 3a). This label represents a route that begins at $o(k_1)$ and ends at $d(k_1)$. It is therefore equivalent to the unmodified route in Figure 1a. No items are delivered to $d(k_1)$ (i.e. $\vartheta_{l_0} = 0$) and no requests are satisfied.

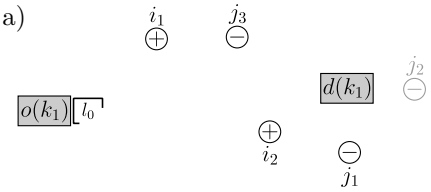
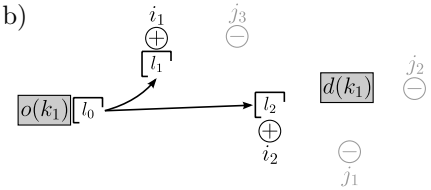
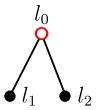
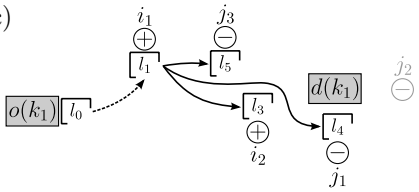
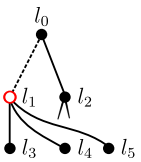
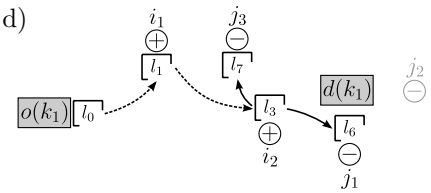
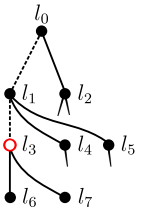
Label propagation	Search tree	New label	Route	ϑ_l	Satisfied requests per column(s)
a) 		l_0	$o(k_1)$-$d(k_1)$	0	$\varnothing$
b) 		l_1 l_2	$o(k_1)$-i_1-$d(k_1)$ $o(k_1)$-i_2-$d(k_1)$	1 1	$\{j_1\}$, $\{j_2\}$ $\{j_1\}$, $\{j_2\}$
c) 		l_3 l_4 l_5	$o(k_1)$-i_1-i_2-$d(k_1)$ $o(k_1)$-i_1-j_1-$d(k_1)$ $o(k_1)$-i_1-j_3-$d(k_1)$	2 0 0	$\{j_1, j_2\}$ $\{j_1\}$ $\{j_3\}$
d) 		l_6 l_7	$o(k_1)$-i_1-i_2-j_1-$d(k_1)$ $o(k_1)$-i_1-i_2-j_3-$d(k_1)$	1 1	$\{j_1, j_2\}$ $\{j_1, j_3\}$, $\{j_2, j_3\}$

Figure 3: An example for propagating labels.

The route of l_0 can only be extended by inserting one of the two pick-up locations i_1 and i_2 ($\mathcal{S}_{l_0}^{\text{open}} = \{i_1, i_2\}$) and, therefore, two new labels l_1 and l_2 are created (see Figure 3b). This results in a search tree, with label l_0 as root. Label l_1 corresponds to the route $o(k_1)$-i_1-$d(k_1)$ and label l_2 corresponds to the route $o(k_1)$-i_2-$d(k_1)$. One item is picked up on both routes and may be taken to $d(k_1)$ for a neighborhood delivery ($\vartheta_{l_1} = \vartheta_{l_2} = 1$).

Label l_1 can be further extended by a supply location ($\mathcal{S}_{l_1}^{\text{open}} = \{i_2\}$) or a request location ($\mathcal{R}^{\text{open}} l_1 = \{j_1, j_3\}$) leading to three new labels l_3, l_4, and l_5 (see Figure 3c). If label l_1 is extended by a supply location, we obtain label l_3, where crowdshipper k_1 takes two items to $d(k_1)$ for neighborhood deliveries ($\vartheta_{l_3} = 2$). If label l_1 is extended by request location j_1,

crowdshipper k_1 performs a home delivery by dropping off the picked-up item i_1 there, see label l_4. If label l_1 is extended by doing a home delivery at j_3, we obtain label l_5. In both cases, no items are taken to $d(k_1)$ ($\vartheta_{l_4} = \vartheta_{l_5} = 0$).

From further extending label l_3, we obtain the labels l_6 and l_7 which are shown in Figure 3d. These labels describe routes on which crowdshipper k_1 performs a home delivery and a neighborhood delivery together. On the route described by label l_6, one of the picked-up items is dropped off at request j_1 and the other is self-sourced from $d(k_1)$ by consumer j_2. Label l_7 describes a route on which one item is home-delivered to request j_3 and the other is taken to $d(k_1)$ to satisfy one out of the two yet unsatisfied requests j_1 and j_2. The algorithm would furthermore extend the not yet processed labels l_2, l_4, and l_5, but we omit this in our example for reasons of simplicity.

The right-hand side of Figure 3 shows the results of the post-processing. We indicate here which subset of requests is satisfied by the corresponding columns in the set packing problem. This set is empty for label l_0 since no requests are satisfied, see Figure 3a. Referring to label l_1 in Figure 3b and Equation (23), $\mathcal{N}_{l_1}^{\text{open}} = \{j_1, j_2\}$ and $\vartheta_{l_1} = 1$ so that we derive two columns from l_1: One in which j_1 is satisfied through a neighborhood delivery using item i_1 and another in which j_2 is satisfied through a neighborhood delivery using item i_1. The resulting subsets of satisfied requests per column are $\{j_1\}$ and $\{j_2\}$, respectively. The columns for the remaining labels are derived likewise.

4.5. Heuristic reductions

The [**spp**] can be turned into a heuristic in various ways by restricting the label setting algorithm in generating labels. This results in fewer crowdshipping columns in the set packing problem and allows for a faster decision making at the expense of a (potentially weaker) solution quality. One of the novel features of our approach is to consider crowdshipper capacities $\zeta > 1$ that allow to assign multiple items to a crowdshipper. Clearly, the capacity parameter ζ restricts the number of deliveries that is maximum assigned to a crowdshipper, see line 16 in Algorithm 1. Therefore, by restricting ζ we can control the depth of the search tree through fathoming branches of possible route extensions, which creates less labels and

yields a heuristic reduction of the search process. We refer to this implementation as $[\mathbf{spp}]_{\zeta}$. For example, the search tree of the label setting algorithm for a crowdshipper with $[\mathbf{spp}]_{\zeta=1}$ can maximum reach two levels in depth. The first level corresponds to a pick-up location and the second to a drop-off location. Referring to Figures 3c and 3d, the labels l_4 and l_5 would be created but the labels l_3, l_6 and l_7 would not. In contrast, the search tree for a crowdshipper with $[\mathbf{spp}]_{\zeta=2}$ can reach up to four levels in depth if all picked-up items are home-delivered. Labels l_3, l_6 and l_7 would be created here, as well. Thus, if we solve a setting with $[\mathbf{spp}]_{\zeta=1}$ instead of $[\mathbf{spp}]_{\zeta=2}$, we truncate the search tree and therefore speed up the algorithm. We do this, however, at the expense of solution quality as less labels are created, regardless of whether they contribute to a profit increase of the platform or not. Note that the heuristic reduction with crowdshipper capacities of $\zeta = 1$ resembles the problem setting investigated in Behrend and Meisel (2018).

Another heuristic solution approach is to limit the number of successor labels a single label can be extended to. This resembles the idea of *beam search*, see Sabuncuoĝlu and Bayiz (1999). To this end, we introduce the integer parameter η. It defines how many reachable supply locations and how many reachable request locations are considered for a label extension, which controls the width of the label setting algorithm's search tree. This variant of the $[\mathbf{spp}]$ is referred to as $[\mathbf{spp}]_{\eta}$. For example, consider a crowdshipper who carries one item and whose route can be extended by three alternative locations, one supply location and two request locations (see l_3, l_4, and l_5 in Figure 3c). With $[\mathbf{spp}]_{\eta=1}$, only two out of the three possible labels are created, one label for a route extension to a supply location and one label for the route extension to one request location. If only a subset of possible extensions is considered, a selection mechanism is required. We choose a greedy approach here and select a route extension that results in minimum travel time. This effects a minimum additional compensation for the crowdshipper. It also allows for higher revenues as more locations can be visited within the given detour flexibility if less detouring is required for a single route extension.

Another idea is to restrict the types of transfer modes to consider in a problem setting. Recall that the post-processing after the label setting algorithm creates crowdshipping

columns to satisfy requests through home deliveries and neighborhood deliveries. In case the problem setting only considers home deliveries, we simply set $\mathcal{N}_l^{\text{open}} = \varnothing$ for all $l \in \mathcal{L}$. We refer to this variant as $[\mathbf{spp}]^{home}$.

The mentioned modifications can be applied in combination. We therefore use the notations $[\mathbf{spp}]_{\zeta,\eta}$ and $[\mathbf{spp}]_{\zeta,\eta}^{home}$ in the following to refer to the different variants of the $[\mathbf{spp}]$.

5. Numerical experiments

We first test our solution method against benchmark results to analyze its potentials. We also study the capabilities of the integrated sharing platform in varied settings. In particular, we focus on different aspects of crowdshipper motivation. We assume intrinsically motivated crowdshippers, that are merely compensated for their detouring, in Section 5.3 and discuss the extent by which a platform would benefit from higher capacities or longer detours. In Section 5.4, we investigate by how much a platform's profit increases from higher crowdshipping capacity, which reveals a potential for further (financial) incentives to motivate crowdshippers to participate. In the last Section 5.5, we compare the profitabilities of an integrated platform, a pure item-sharing platform, and cooperating item-sharing and crowdshipping platforms that can draw on each others' services.

5.1. Experimental setting

Our experiments are based on the problem setting used by Behrend and Meisel (2018). It refers to a 30 × 30 km area centered in the city center of Atlanta in the US state of Georgia. The distribution of supply and request locations in the considered area is based on the population density. Origin and destination locations of trips are distributed according to information that is provided by the Atlanta Regional Commission (2019). Behrend and Meisel (2018) created ten test instances with sets of 200 supplies, 200 requests, and 200 trips each. Instances of smaller *density* are derived from this by only considering the first 10, 25, 50, 75, 100, and 150 supplies, requests in each set. Fastest routes between two locations are generated by the Open Source Routing Machine (Geofabrik GmbH, 2018; Open Source Routing Machine, 2019). Routes and travel times are consequently based on the real road

network of Atlanta. The average travel time of the unmodified trips is 15.2 min, with a standard deviation of 5.5 min. We refer to this original problem set from Behrend and Meisel (2018) as *setting 1*.

We further generate instances for a *setting 2* that differs from setting 1 only in the way in which supply and request locations of community members as well as origin and destination locations of trips are distributed in the same area. In setting 2, these locations are distributed uniformly. Travel times and routes are still taken from the Atlanta road network. We include setting 2 for a baseline comparison in which we ignore agglomerations of supplies and requests in certain residential areas of Atlanta. Furthermore, the distribution of trips is independent from the survey of the Atlanta Regional Commission that included high travel activity between busy places such as the city center and the Atlanta airport. The trip duration in setting 2 is on average 23.5 min, with a standard deviation of 9.3 min.

The default values of the further parameters are adopted from Behrend and Meisel (2018) and identical for settings 1 and 2. A self-sourcing flexibility of $f_j^{\text{ssrc}} = 10$ min is assumed for every consumer and the crowdshippers' relative detour flexibility is $\Delta = 20\%$ of the direct trip's duration, i.e. a crowdshipper k accepts to extent his/her travel time by $f_k^{\text{dtr}} = t_k \cdot 0.2$. The revenue from a served consumer is $r^{\text{ssrc}} = \$10$ if the consumer needs to self-source the item. It is $r^{\text{home}} = \$15$ if the item is delivered directly to the consumer. The compensation rate for crowdshippers is set to $c^{\text{dtr}} = \$30$ per hour.

We conduct our computations on a 3.4 GHz machine with four cores and 16 GB RAM. The solution framework is implemented in Java 8. Gurobi 7.5 is used for solving the optimization models. All test instances and solutions of the central experiments are available at `http://dx.doi.org/10.17632/d9zc7knxdz.1`.

5.2. Performance comparison with previous solution approach

The first experiment is a performance comparison between the new [**spp**] approach presented in this paper and those methods that were previously proposed by Behrend and Meisel (2018). Since the methods of Behrend and Meisel (2018) can only handle a capacity of $\zeta = 1$, we consider here the $[\mathbf{spp}]_{\zeta=1}$ reduction in order to conduct a comparison of the

different approaches. More precisely, Behrend and Meisel (2018) formulated two three-dimensional assignment problems, namely [**home**] and [**nbrhd**], and they solve them to optimality using Gurobi. [**home**] serves as a benchmark for the [**spp**] variant $[\mathbf{spp}]^{\text{home}}_{\zeta=1}$ since self-sourcing and home deliveries are the only allowed transfer modes in these two methods. In contrast, [**nbrhd**] includes self-sourcing, home delivery, and neighborhood delivery and therefore represents the benchmark for $[\mathbf{spp}]_{\zeta=1}$. The potentials of capacities $\zeta > 1$, which can only be exploited using our new [**spp**] method, are analyzed in the later experiments.

(a) Setting 1

Density	[**home**]		[**nbrhd**]		$[\mathbf{spp}]^{home}_{\zeta=1}$		$[\mathbf{spp}]_{\zeta=1}$	
	Profit [$]	CPU [s]	Profit [$]	CPU [s]	Profit [$]	CPU [s]	Profit [$]	CPU [s]
10	15	0.01	15	0.01	15	0.01	15	0.01
25	74	0.02	80	0.05	74	0.01	80	0.01
50	220	0.23	233	0.54	220	0.01	233	0.01
75	402	0.90	419	1.97	402	0.01	419	0.01
100	611	2.08	633	4.69	611	0.01	633	0.02
150	1117	7.39	1138	24.61	1117	0.03	1138	0.06
200	1614	45.10	1649	455.83	1614	0.06	1649	0.13
∅	579	7.96	595	69.67	579	0.02	595	0.04

(b) Setting 2

Density	[**home**]		[**nbrhd**]		$[\mathbf{spp}]^{home}_{\zeta=1}$		$[\mathbf{spp}]_{\zeta=1}$	
	Profit [$]	CPU [s]	Profit [$]	CPU [s]	Profit [$]	CPU [s]	Profit [$]	CPU [s]
10	15	0.01	16	0.01	15	0.01	16	0.01
25	100	0.05	105	0.09	100	0.01	105	0.01
50	328	0.35	348	0.73	328	0.02	348	0.02
75	601	1.10	623	2.25	601	0.04	623	0.04
100	896	2.53	920	5.79	896	0.09	920	0.11
150	1531	13.12	1551	33.20	1531	0.47	1551	0.64
200	2207	30.68	2229	770.67	2207	1.68	2229	2.41
∅	811	6.83	827	116.11	811	0.33	827	0.46

Table 3: Comparison of the new solution approaches with benchmarks for capacity $\zeta = 1$.

Tables 3a and 3b show the profits and solution times of all methods for settings 1 and 2, respectively. The reported values are averages over the ten instances per density. The CPU values reported for the [**spp**]-variants include the time needed for label setting (column generation) and solving the set packing problem. Looking at the profits, we see that [**home**] and $[\mathbf{spp}]^{\text{home}}_{\zeta=1}$ as well as [**nbrhd**] and $[\mathbf{spp}]_{\zeta=1}$ deliver the same optimal profits. These results are expected as all methods deliver optimal solutions under the considered combina-

tion of transfer modes and the fixed capacity of $\zeta = 1$. However, we see that the new [**spp**] method solves all instances within negligible solution times whereas the assignment problem formulations of Behrend and Meisel (2018) require substantial CPU times, especially for the larger instances. For example, the most challenging problem in setting 1, with density 200 and all three transfer modes included, is solved in over 400 seconds with [**nbrhd**] whereas $[\mathbf{spp}]_{\zeta=1}$ requires just about one tenth of a second, see Table 3a. For setting 2, we observe slightly higher runtimes for the [**spp**]. This increase can be attributed to a higher absolute detour willingness of crowdshippers in setting 2 (4.7 min vs. 3.0 min on average) and more evenly distributed trips, supplies, and requests, which requires to evaluate more alternative combinations (longer runtimes) but also allows for a better coverage (higher profits). Still, our new method clearly outperforms all competing methods with respect to runtime, even for the most challenging setting.

The [**spp**] approach is also more efficient when it comes to including neighborhood deliveries into the planning. The runtime increase due to a switch from [**home**] to [**nbrhd**] is several minutes whereas it is less than a second if we solve $[\mathbf{spp}]_{\zeta=1}$ instead of $[\mathbf{spp}]^{\text{home}}_{\zeta=1}$. We conclude that the [**spp**] is a much faster solution approach compared with the methods proposed in Behrend and Meisel (2018).

5.3. Analysis of crowdshippers' capabilities and heuristic reductions

Crowdshipper's capabilities are described by their capacity and their detour flexibility. With high capabilities, crowdshippers accept multiple deliveries and long detours and are therefore very useful in supporting item exchanges. We conduct an extensive experimental study in the following to quantify the benefit that stems from higher capabilities.

The experiments are based on setting 2, instances with a moderate density of 100, and all three transfer modes allowed. We consider capacities of up to $\zeta = 3$ for all crowdshippers and we range the relative detour flexibility between $\Delta = 20\%$ and $\Delta = 80\%$. Since both capacities and detour flexibilities are increased compared to the previous experiment, the problems considered here are more challenging for the algorithm and it may become necessary to restrict the search tree to obtain a solution. Recall that ζ controls the depth of the search

Table 4: Profits and solution times for setting 2 and density 100.

Detour flexibility (Δ) and capacity (ζ)		$\eta = 100$ (optimal)		$\eta = 50$		$\eta = 20$		$\eta = 10$		$\eta = 5$		$\eta = 1$	
		Profit	CPU [s]	Profit	CPU [s]	Profit	CPU [s]	Profit	CPU [s]	Profit	CPU [s]	Profit	CPU [s]
$\Delta = 20$ %	$\zeta = 1$	\$920	0.13	0.00%	0.13	−0.45%	0.12	−3.22%	0.06	−8.01%	0.03	−30.16%	0.01
	$\zeta = 2$	1.17%	0.65	1.17%	0.54	0.83%	0.51	−1.74%	0.37	−6.19%	0.13	−29.06%	0.01
	$\zeta = 3$	1.24%	1.01	1.24%	0.96	0.85%	0.87	−1.69%	0.59	−6.12%	0.17	−29.06%	0.01
$\Delta = 30$ %	$\zeta = 1$	11.92%	0.41	11.85%	0.35	10.39%	0.24	6.11%	0.18	−0.76%	0.06	−28.23%	0.01
	$\zeta = 2$	15.13%	5.25	15.09%	5.22	13.84%	3.49	9.97%	1.39	3.36%	0.39	−25.96%	0.01
	$\zeta = 3$	15.20%	27.41	15.16%	26.07	13.90%	15.24	10.07%	4.69	3.53%	1.02	−25.92%	0.01
$\Delta = 40$ %	$\zeta = 1$	17.03%	0.76	16.90%	0.80	15.09%	0.45	10.52%	0.15	3.21%	0.07	−26.80%	0.01
	$\zeta = 2$	22.13%	32.70	22.04%	29.33	20.32%	15.24	16.15%	4.63	9.79%	0.78	−23.53%	0.02
	$\zeta = 3$	n/a	n/a	n/a	n/a	20.64%	136.37	16.44%	26.71	10.17%	3.09	−23.05%	0.03
$\Delta = 60$ %	$\zeta = 1$	20.02%	2.00	19.94%	1.67	18.56%	0.67	14.61%	0.30	6.57%	0.11	−25.42%	0.01
	$\zeta = 2$	n/a	n/a	26.77%	406.27	25.35%	85.20	22.34%	18.30	16.07%	2.39	−20.07%	0.03
	$\zeta = 3$	n/a	n/a	n/a	n/a	n/a	n/a	23.11%	232.06	17.10%	19.43	−19.21%	0.03
$\Delta = 80$ %	$\zeta = 1$	20.49%	4.49	20.46%	2.98	19.21%	1.04	15.89%	0.35	8.03%	0.13	−24.99%	0.02
	$\zeta = 2$	n/a	n/a	n/a	n/a	26.64%	209.90	24.10%	27.22	18.67%	4.13	−18.34%	0.03
	$\zeta = 3$	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	20.90%	37.88	−16.50%	0.05

tree as it restricts the length of the route. η controls the width of the search tree by restricting route extensions to promising locations. For this purpose, η ranges between 100 and 1, where 100 refers to the unmodified width of the search tree and therefore corresponds to the exact method that produces optimal solutions. The search tree thins out with smaller values of η, meaning that an increasing amount of feasible matchings is ignored.

Table 4 shows the relative improvement of profits due to a variation of parameters and the corresponding runtime in seconds, both averaged over 10 instances. The reference value for the profits is the optimal profit of \$920 for setting 2 with the limited capacity $\zeta = 1$ and the default detour flexibility of $\Delta = 20\%$, see Table 3b. Positive percentage values in Table 4 indicate a relative improvement of profits whereas negative values indicate a loss of profit. Entries 'n/a' indicate out-of-memory situations due to a too large search tree in which case the problem cannot be solved by the [**spp**] approach. The arrangement of values allows for various ceteris paribus analyses. Row-wise comparisons reflect the change in solution quality and runtime when crowdshippers offer higher capacities. This effect can be analyzed for different levels of detour flexibility. Column-wise comparisons address the trade-off between lower profits and runtime improvements due to an increasingly restricted search tree.

The results of Table 4 confirm the expectations that a platform is more profitable if crowdshippers offer higher capacities. The magnitude of the benefit does, however, strongly

Table 5: Number of generated columns and generation time for setting 2 and density 100.

Detour flexibility (Δ) and capacity (ζ)		$\eta = 100$ (optimal)		$\eta = 50$		$\eta = 20$		$\eta = 10$		$\eta = 5$		$\eta = 1$	
		Columns	CPU [s]	Columns	CPU [s]	Columns	CPU [s]	Columns	CPU [s]	Columns	CPU [s]	Columns	CPU [s]
$\Delta = 20$ %	$\zeta = 1$	3239	0.02	3227	0.02	2838	0.02	2025	0.01	1141	0.01	242	0.01
	$\zeta = 2$	11451	0.09	11431	0.08	10216	0.06	6941	0.04	3322	0.02	314	0.01
	$\zeta = 3$	17317	0.16	17297	0.15	15663	0.12	10478	0.07	4558	0.03	328	0.01
$\Delta = 30$ %	$\zeta = 1$	9883	0.10	9574	0.07	6798	0.05	3742	0.03	1691	0.01	264	0.01
	$\zeta = 2$	83149	0.57	81658	0.69	58217	0.39	28015	0.18	8907	0.05	384	0.01
	$\zeta = 3$	221992	5.53	219608	5.39	157336	1.92	71027	0.58	18356	0.14	427	0.01
$\Delta = 40$ %	$\zeta = 1$	21390	0.20	19773	0.16	11317	0.10	5234	0.04	2068	0.03	274	0.01
	$\zeta = 2$	382701	3.97	360943	3.29	198274	2.06	71530	0.49	16837	0.11	434	0.01
	$\zeta = 3$	n/a	n/a	n/a	n/a	948624	21.54	306223	3.05	52863	0.41	507	0.01
$\Delta = 60$ %	$\zeta = 1$	58130	0.70	47312	0.55	19450	0.22	7375	0.10	2536	0.04	283	0.01
	$\zeta = 2$	n/a	n/a	2733364	139.93	939439	8.68	224425	1.76	36989	0.36	517	0.02
	$\zeta = 3$	n/a	n/a	n/a	n/a	n/a	n/a	2179454	32.41	216502	2.05	666	0.01
$\Delta = 80$ %	$\zeta = 1$	104933	1.88	76925	1.11	25588	0.39	8715	0.14	2777	0.08	287	0.01
	$\zeta = 2$	n/a	n/a	n/a	n/a	2412261	32.24	438604	4.70	57837	0.64	568	0.01
	$\zeta = 3$	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	532566	6.67	799	0.02

depend on the detour flexibility of crowdshippers. Referring to the optimal values (column $\eta = 100$), a higher capacity of $\zeta = 2$ results in a profit increase of only 1.17 percentage points under the default detour flexibility of $\Delta = 20\%$ but of 5.10 percentage points under $\Delta = 40\%$ (17.03% for $\zeta = 1$ vs. 22.13% for $\zeta = 2$). Crowdshippers' detour flexibility must consequently be sufficiently large to exploit higher capacities.

When crowdshippers offer larger capacities and accept long detours, it results in higher profits but it also significantly increases the planning complexity. With capacity $\zeta = 1$, we obtain optimal solutions in a reasonable amount of time, irrespective of the detour flexibility (see the CPU times in rows belonging to '$\zeta = 1$' under $\eta = 100$ in Table 4). If we solve the same instances with capacity $\zeta = 2$, the computation times grow much faster and we even run out of memory for detour flexibilities of $\Delta = 60\%$ and above (see n/a entries under $\eta = 100$ in Table 4). This is in line with Table 5 which shows the strong increase of the number of generated columns and the corresponding CPU times of the label setting with increasing capacity and detour flexibility. The platform consequently faces a trade-off between what crowdshippers offer to do and what the decision making is capable to handle.

An alternative approach is to obtain heuristic solutions by restricting the width of the algorithm's search tree using parameter η. We observe from Table 4 that decreasing η speeds up the solution process drastically at the cost of a decrease of solution quality. It also helps to solve instances feasibly for which the exact solution approach ([**spp**] with $\eta = 100$) runs

out of memory. For example, referring to a setting with a detour flexibility of $\Delta = 60\%$ and capacity $\zeta = 2$, we obtain a feasible solution with $\eta = 50$ and below. Note that the associated profit increase of 26.77% is larger than what the platform could optimally obtain if less capacity (20.02%) or less detour flexibility (22.13%) is considered instead. Turning the [**spp**] into a heuristic by decreasing η can consequently help to increase the achieved solution quality. It should be noted, however, that the results are sensitive to changes of η, especially if η is very small. Then, the solution quality may be less than what could have been obtained by considering smaller capacities or less detour flexibility but higher values of η. This becomes obvious in the column for $\eta = 1$ of Table 4, where the profit obtained from the heuristic is consistently lower than the reference value, despite higher capabilities.

We conclude from this analysis that a platform's profitability benefits from high capacities and large detour flexibilities of crowdshippers. However, this results in a more complex problem that may be very challenging to solve to optimality. In such cases, the [**spp**] approach allows to heuristically obtain a solution by considering less capacity, less detour flexibility, or fewer feasible matchings. The highest profit increase of 26.77% in Table 4 is obtained with a combination of all three (detour flexibility of $\Delta = 60\%$, capacity of $\zeta = 2$, and $\eta = 50$). Thus, the option to turn the [**spp**] into a heuristic in three different ways makes it versatile applicable and, therefore, powerful.

5.4. Incentivizing high crowdshipping capacity

The fulfillment of more than one delivery on a single trip requires additional effort from the crowdshipper as more stops and coordination with consumers are required. However, crowdshippers are not compensated for this effort in the compensation scheme that is considered here so that offering higher capacities only pays off if the fulfillment of additional deliveries results in more detouring. In contrast, the platform benefits from higher capacities in terms of additional profit and, therefore, may have an interest in further incentivizing crowdshippers to make multiple deliveries.

Against this background, we analyze the marginal profit due to higher capacities as this profit could be shared among the platform and the crowdshippers to foster win-win situa-

Table 6: Scope for incentivizing crowdshippers based on marginal profit per extra capacity.

Number of crowdshippers ($\|K\|$) and capacity (ζ)		**Transfer modes**				**Marginal profit per extra capacity unit used** [$]
		Self-sourcings	Home deliveries	Neighborhood deliveries	$\sum$	
$\|K\| = 0$		46.9	0.0	0.0	46.9	0.0
$\|K\| = 25$	$\zeta = 1$	40.9	19.9	1.4	62.2	0.0
	$\zeta = 2$	35.4	26.0	3.7	65.1	4.9
	$\zeta = 3$	35.0	26.2	4.3	65.5	4.9
$\|K\| = 50$	$\zeta = 1$	33.7	39.3	3.1	76.1	0.0
	$\zeta = 2$	25.6	48.1	5.0	78.7	4.2
	$\zeta = 3$	24.5	48.3	6.0	78.8	4.1
$\|K\| = 100$	$\zeta = 1$	15.6	70.5	3.6	89.7	0.0
	$\zeta = 2$	10.9	75.6	3.9	90.4	2.1
	$\zeta = 3$	10.6	75.5	4.3	90.4	2.0
$\|K\| = 150$	$\zeta = 1$	6.4	86.3	2.3	95.0	0.0
	$\zeta = 2$	4.2	88.3	2.7	95.2	1.5
	$\zeta = 3$	3.9	88.7	2.6	95.2	1.5
$\|K\| = 200$	$\zeta = 1$	1.9	93.3	1.6	96.8	0.0
	$\zeta = 2$	1.8	93.4	1.8	97.0	1.0
	$\zeta = 3$	1.7	93.5	1.8	97.0	1.0

tions. We compute such marginal profits as follows. For capacity $\zeta = 2$, we subtract from the profit obtained from $[\mathbf{spp}]_{\zeta=2}$ the profit obtained from $[\mathbf{spp}]_{\zeta=1}$ and divide this value by the number of items that are delivered using the second capacity unit of a crowdshipper. For example, if the profit is \$100 under capacity $\zeta = 1$ and \$145 under capacity $\zeta = 2$ where 9 crowdshippers deliver two items each (i.e. 9 items exploit the second capacity unit), the marginal profit is $(\$145 - \$100)/9 = \$5$ per extra capacity unit used. For capacity $\zeta = 3$, we use the profit obtained from $[\mathbf{spp}]_{\zeta=3}$ and count all those items that are carried using the second or third capacity unit of a crowdshipper for computing the marginal profit rate.

Table 6 shows results for setting 2, density 100, and detour flexibility of 30% under varied numbers of crowdshippers of $|K| = 0$ to $|K| = 200$ as well as varied capacities of $\zeta = 1$ to $\zeta = 3$. The table shows the frequency of selected transfer modes in the profit maximizing solutions and the marginal profit per extra capacity unit used.

We see that the platform's scope for incentivizing multiple deliveries is especially high if crowdshippers are scarce. For example, if there are only $|K| = 25$ crowdshippers but each of them offers capacity $\zeta = 2$, the platform could pay up to \$4.9 to every crowdshipper who executes two deliveries without being worse off than under $\zeta = 1$. Such a compensation is equivalent to almost 10 minutes of detouring with the employed compensation scheme. The

considerable potential for incentivizing crowdshippers stems from new options to replace self-sourcings with home deliveries, to conduct crowdshipping more efficiently, and to serve so far unserved requests, see the split of transfer modes in Table 6. If crowdshippers are less scarce, the marginal profit rate decreases as expected. However, it remains above zero even under $|K| = 200$ as it is still profitable to have at least some crowdshippers execute two or more deliveries. We observe that the marginal profit rates for a given value $|K|$ are almost identical for $\zeta = 2$ and $\zeta = 3$ meaning that the platform could pay a crowdshipper almost the same for the second and third capacity unit used.

The frequency of selected transfer modes in Table 6 suggests a shift from self-sourcings to home deliveries as more crowdshippers become available. If no crowdshippers are available at all ($|K| = 0$), self-sourcing is the only option where less than half of all requests can be satisfied. In contrast, in the scenario with 200 crowdshippers, 97% of requests can be satisfied, where less than 2% are self-sourcings and 95% are served by crowdshippers. We see that home deliveries are the predominant transfer mode for crowdshipping. The platform prefers home deliveries because of high revenues and small compensations as a result of a good fit between trips and delivery jobs. Neighborhood deliveries are the least profitable mode and are only used to enable further matchings that cannot be established otherwise. A capacity increase to $\zeta = 2$ is particularly useful when only few crowdshippers are available. For instance, roughly 6 additional home deliveries and 2 additional neighborhood deliveries are possible if 25 crowdshippers are willing to conduct two deliveries instead of just one. This benefit fades out with more crowdshippers being available where additional crowdshipping capacity is rarely required anymore. A capacity increase to $\zeta = 3$ still allows for further item exchanges through crowdshipping. However, the overall margin is low since either the limited detour flexibility makes it difficult to fulfill three deliveries per trip or the extra capacity is not required if there are sufficiently many other crowdshippers.

5.5. Item-sharing and crowdshipping on separate platforms

Item-sharing and crowdshipping are so far dealt with in practice on separate platforms that operate in complete isolation. An intermediate step between the status quo and our proposed

concept of a fully integrated platform is a partial interaction (cooperation) of item-sharing and crowdshipping platforms. We analyze in the following the potential of two such interacting platforms based on a conceivable cooperation scheme that could be implemented in practice relatively easily. More precisely, we assume that an item-sharing platform first conducts a supply-request matching and then announces the resulting delivery jobs on a crowdshipping platform. The crowdshipping platform then searches for profitable home delivery opportunities by matching the announced delivery jobs to the available crowdshipping trips. We assume that neighborhood deliveries are not supported here as this transfer mode requires coordination of a consumer and a crowdshipper who, in this setting, belong to different platforms. Finally, the item-sharing platform decides on the transfer mode that is eventually used per transaction based on revenues and compensation. Clearly, many alternative coordination schemes are conceivable but we restrict ourselves to this single scheme for reasons of brevity.

More technically, we implement such an interaction in three steps. First, the item-sharing platform matches supplies and requests only based on self-sourcing with the objective to maximize profit. To do so, we generate self-sourcing columns as described in Subsection 3.3 and solve the set packing problem. Feasible matches directly translate into delivery jobs. Furthermore, the item-sharing platform matches remaining supplies and requests such that the transportation distance is minimized. These matches cannot be fulfilled in self-sourcing but they are promising opportunities for crowdshipping with low compensation costs. This problem is a simple two-dimensional assignment problem that is solved here using the exact Kuhn-Munkres algorithm (Kuhn, 1955). Next, the crowdshipping platform attempts to find for each of the delivery jobs an eligible crowdshipper, which is done here by creating columns for all feasible trip-delivery combinations and solving the set packing problem. Eventually, the item-sharing platform solves a further set packing problem to determine for each request whether to use self-sourcing or the home delivery option offered by the crowdshipping platform.

Table 7 shows the profits and frequencies of used transfer modes in the considered platform concepts at different density levels. These results are for setting 2, the default relative

Table 7: Comparison of different platform concepts.

Density	Isolated item-sharing		Interacting platforms			Fully integrated platform			
	Profit [$]	Self-sourcings	Rel. profit increase	Self-sourcings	Home deliveries	Rel. profit increase	Self-sourcings	Home deliveries	Neighborh. deliveries
10	10	1.0	40.00%	0.9	0.4	60.00%	0.8	0.5	0.2
25	49	4.9	46.94%	3.2	3.3	114.29%	2.6	5.8	0.7
50	169	16.9	38.46%	7.5	12.5	105.92%	8.6	18.2	3.8
75	302	30.2	35.10%	12.6	21.9	106.29%	14.3	33.9	6.1
100	469	46.9	36.46%	18.7	35.3	96.16%	21.9	50.7	6.9
150	846	84.6	33.81%	29.1	65.4	83.33%	31.7	89.4	10.6
200	1286	128.6	31.96%	39.4	100.6	73.33%	38.8	135.9	10.5

detour flexibility of 20%, and the default capacity $\zeta = 1$. The columns labeled 'Isolated item-sharing' refer to a pure item-sharing platform that does not rely on a crowdshipping platform at all. This concept serves as a benchmark for the other platform concepts. The columns 'Interacting platforms' refer to a cooperation of an item-sharing and a crowdshipping platform using the procedure that was described before. The columns 'Fully integrated platform' refers to the concept that was investigated throughout this paper. The results reported for this concept stem from our [**spp**]-method.

We observe that the isolated item-sharing platform can use self-sourcing only, which drastically limits its profitability. If the item-sharing platform interacts with a crowdshipping platform, the profit can be increased consistently by about 30% to 50%. The outsourcing of delivery jobs to crowdshippers is very effective here. In almost all densities, the number of home deliveries is substantially higher than the number of remaining self-sourcings. This increases profits as home deliveries have higher revenues than self-sourcings and because additional requests can be satisfied if the item-sharing platform cooperates with a crowdshipping platform. For instance, the item-sharing platform satisfies 128.6 requests at density 200 through self-sourcing. If it interacts with a crowdshipping platform, merely 39.4 of these requests are still self-sourcings whereas another 100.6 requests are home-delivered. The overall number of satisfied requests increases from 128.6 to 140, meaning that 11.4 additional requests are satisfied because of the cooperation of the platforms. However, if we consider the performance of the fully integrated platform, we see even higher profits as this platform takes more advantage of crowdshipping. Such a platform can obtain profits that are about 60% to 106% higher than the profits of the isolated item-sharing platform, which

clearly outperforms the concept of interacting platforms too.

6. Conclusions

We have presented a new exact solution procedure for the capacitated item-sharing and crowdshipping problem, where each crowdshipper can transport a given number of items. The decisions in this problem address the supply-request matching that is part of the item-sharing together with the transportation planning for the shared items. For the latter, we consider self-sourcing of items through the requesting consumers as well as home-deliveries and neighborhood-deliveries through crowdshippers. We have formulated a corresponding set packing problem where columns represent feasible self-sourcings and crowdshipper routes. While it is trivial to generate the self-sourcing columns, the columns for crowdshipper routes demand a dedicated label setting procedure. We have described this procedure in detail, including the label setting itself as well as a post-processing that derives columns from the labels. We also described how to restrict the depth and width of the label setting search process to obtain a scalable heuristic procedure.

We have conducted extensive experiments based on a problem setting for the region of Atlanta, Georgia, that uses travel data surveyed by the Atlanta Regional Commission. A second problem setting has been generated using artificial travel data. Our results show that the profitability of the sharing platform can improve if crowdshippers offer to transport more than one item on their trips. Anyhow, high detour flexibilities of crowdshippers are even more important for the profitability than high capacities. This finding holds for all settings analyzed in our experiments. Since higher capacities and higher detour flexibilities make the label setting more difficult, the proposed heuristic mechanisms have to be applied if the search space becomes too large. Our experiments show that the heuristic can be very well adapted through the included control parameters. We have also shown that our new exact method is significantly faster than other exact methods that have been proposed in previous research for a simpler problem where each crowdshipper can transport at most one item. Finally, a series of experiments shows that a fully integrated platform clearly outperforms a less deep cooperation of independent item-sharing and crowdshipping platforms. Only then

can crowdshippers' capabilities be used to the full extent and, for example, higher incentives can be paid for conducting multiple delivery jobs on their trips. This provides practitioners guidance for the design of such a platform. We therefore recommend to strive for a deepest possible collaboration among item-sharing and crowdshipping platforms to achieve a highest possible profitability.

Future research might transfer the problem into a multi-period version where items are handed over from one request location to the next or where returns of items to the supply-locations are captured too. Furthermore, the analysis of the environmental impact of such sharing systems could also be handled.

Acknowledgments

We are grateful for the comments of two anonymous reviewers that helped us to considerably improve the quality of the paper.

Bibliography

Agatz, N., Erera, A., Savelsbergh, M., Wang, X., 2012. Optimization for dynamic ride-sharing: A review. European Journal of Operational Research 223, 295–303.

Amazon Flex, 2019. `https://flex.amazon.com` (Last accessed on 2019-03-04).

Archetti, C., Savelsbergh, M., Speranza, M., 2016. The vehicle routing problem with occasional drivers. European Journal of Operational Research 254, 472–480.

Armstrong, M., Murlis, H., 2007. Reward Management: A Handbook of Remuneration Strategy and Practice. Kogan Page.

Arslan, A. M., Agatz, N., Kroon, L., Zuidwijk, R., 2019. Crowdsourced delivery – a dynamic pickup and delivery problem with ad hoc drivers. Transportation Science 53, 222–235.

Atlanta Regional Commission, 2019. `http://atlantaregional.org` (As of 2017-09-27, last accessed on 2019-03-04).

Barnes, B. J., Mattsson, J., 2016. Understanding current and future issues in collaborative consumption: A four-stage delphi study. Technological Forecasting and Social Change 104, 200–211.

Böcker, L., Meelen, T., 2017. Sharing for people, planet or profit? Analysing motivations for intended sharing economy participation. Environmental Innovation and Societal Transitions 23, 28–39.

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

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

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

Botsman, R., Rogers, R., 2011. What's Mine Is Yours. HarperBusiness, USA.

Chen, W., Mes, M., Schutten, M., Sep 2018. Multi-hop driver-parcel matching problem with time windows. Flexible Services and Manufacturing Journal 30, 517–553.

Cleophas, C., Cottrill, C., Ehmke, J. F., Tierney, K., 2019. Collaborative urban transportation: Recent advances in theory and practice. European Journal of Operational Research 273, 801–816.

Cordeau, J.-F., Laporte, G., 2007. The dial-a-ride problem: Models and algorithms. Annals of Operations Research 153, 29–46.

Deliv, 2019. `https://www.deliv.co` (Last accessed on 2019-03-04).

Devari, A., Nikolaev, A. G., He, Q., 2017. Crowdsourcing the last mile delivery of online orders by exploiting the social networks of retail store customers. Transportation Research Part E: Logistics and Transportation Review 105, 105–122.

Erento, 2019. `http://www.erento.com` (Last accessed on 2019-03-04).

Furuhata, M., Dessouky, M., Ordóñez, F., Brunet, M.-E., Wang, X., Koenig, S., 2013. Ridesharing: The state-of-the-art and future directions. Transportation Research Part B: Methodological 57, 28–46.

Geofabrik GmbH, 2018. Openstreetmap data extracts. `http://download.geofabrik.de` (As of 2017-02-16, last accessed on 2019-03-04).

Herbawi, W., Weber, M., 2012. The ridematching problem with time windows in dynamic ridesharing: A model and a genetic algorithm. In: IEEE Congress on Evolutionary Computation. IEEE, pp. 1–8.

Ho, S. C., Szeto, W., Kuo, Y.-H., Leung, J. M., Petering, M., Tou, T. W., 2018. A survey of dial-a-ride problems: Literature review and recent developments. Transportation Research Part B: Methodological 111, 395 – 421.

Irnich, S., Desaulniers, G., 2005. Shortest path problems with resource constraints. In: Desaulniers, G., Desrosiers, J., Solomon, M. M. (Eds.), Column Generation. Springer US, Boston, MA, pp. 33–65.

Jade Zabiore, 2019. `https://jadezabiore.pl/en` (Last accessed on 2019-03-04).

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

Kamar, E., Horvitz, E., 2009. Collaboration and shared plans in the open world: Studies of ridesharing. In: Proceedings of the 21st International Jont Conference on Artifical Intelligence. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 187–194.

Kuhn, H. W., 1955. The hungarian method for the assignment problem. Naval Research Logistics Quarterly 2, 83–97.

Marcucci, E., Pira, M. L., Carrocci, C. S., Gatta, V., Pieralice, E., 2017. Connected shared mobility for passengers and freight: Investigating the potential of crowdshipping in urban areas. In: 5th IEEE Inter-

national Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS). IEEE, pp. 839–843.

Miller, J., Nie, Y. M., Stathopoulos, A., 2017. Crowdsourced urban package delivery: Modeling traveler willingness to work as crowdshippers. Transportation Research Record: Journal of the Transportation Research Board 2610, 67–75.

Nimber, 2019. `https://www.nimber.com` (Last accessed on 2019-03-04).

Open Source Routing Machine, 2019. `http://project-osrm.org` (As of 2017-02-17, last accessed on 2019-03-04).

Paloheimo, H., Lettenmeier, M., Waris, H., 2015. Transport reduction by crowdsourced deliveries – a library case in Finland. Journal of Cleaner Production 132, 240–251.

Parragh, S., Doerner, K., Hartl, R., 2008. A survey on pickup and delivery problems. Management Review Quarterly 58, 81–117.

Postmates, 2019. `https://postmates.com` (Last accessed on 2019-03-04).

Punel, A., Ermagun, A., Stathopoulos, A., 2018. Studying determinants of crowd-shipping use. Travel Behaviour and Society 12, 30–40.

Punel, A., Stathopoulos, A., 2017. Modeling the acceptability of crowdsourced goods deliveries: Role of context and experience effects. Transportation Research Part E: Logistics and Transportation Review 105, 18–38.

Qi, W., Li, L., Liu, S., Shen, Z.-J. M., 2018. Shared mobility for last-mile delivery: Design, operational prescriptions, and environmental impact. Manufacturing & Service Operations Management 20, 737–751.

Sabuncuoğlu, I., Bayiz, M., 1999. Job shop scheduling with beam search. European Journal of Operational Research 118, 390 – 412.

Štiglic, M., Agatz, N., Savelsbergh, M., Gradišar, M., 2015. The benefits of meeting points in ride-sharing systems. Transportation Research Part B: Methodological 82, 36–53.

Wang, Y., Zhang, D., Liu, Q., Shen, F., Lee, L. H., 2016. Towards enhancing the last-mile delivery: An effective crowd-tasking model with scalable solutions. Transportation Research Part E: Logistics and Transportation Review 93, 279–293.

Zilok, 2019. `http://us.zilok.com` (Last accessed on 2019-03-04).