A column and row generation approach to the crowd-shipping problem with transfers

Stokkink, Patrick; Cordeau, Jean François; Geroliminis, Nikolas

DOI
10.1016/j.omega.2024.103134

Publication date
2024

Document Version
Final published version

Published in
Omega (United Kingdom)

Omega 128 (2024) 103134

Contents lists available at ScienceDirect

Omega

journal homepage: www.elsevier.com/locate/omega

A column and row generation approach to the crowd-shipping problem with transfers☆

Patrick Stokkink [a,c,*], Jean-François Cordeau [b], Nikolas Geroliminis [a]

[a] École Polytechnique Fédérale de Lausanne (EPFL), Urban Transport Systems Laboratory (LUTS), Switzerland
[b] HEC Montréal, Canada
[c] Faculty of Technology, Policy, and Management (TPM), Technical University Delft, Netherlands

ARTICLE INFO

Keywords:
Crowd-shipping
Last-mile delivery
Transfers
Column generation
Row generation

ABSTRACT

Crowd-shipping is a last-mile delivery concept in which commuters pick up and deliver parcels on their pre-existing paths. In urban areas, crowd-shipping circumvents problems that traditional last-mile delivery systems suffer from, such as road congestion and lack of parking spaces, especially if more sustainable modes of transport are utilized, like bikes or e-bikes. Using transfers between crowd-shippers allows for expanding the service area and improving the overall performance. However, as this requires synchronization over space and time, it makes the problem more complex. In this work, we develop a model that can encompass fully heterogeneous crowd-shippers and parcels. Thereby, it allows for both direct time-synchronized transfers as well as intermediate storage at designated parcel lockers. We design a column generation algorithm to solve large-scale realistic instances to optimality. We extend the problem to allow crowd-shippers to carry multiple parcels at the same time and for this, we extend the algorithm to simultaneous column and row generation. We evaluate the performance of our algorithm as well as the potential of crowd-shipping with transfers on a realistic case study of a bike-based crowd-shipping system in Washington DC. Our methods solve realistic instances with 1000 crowd-shippers and 1000 parcels within minutes. The results show that a gain in revenue and service level of 30% can be obtained by allowing transfers. By letting part of the population of crowd-shippers carry two or three parcels at the same time, the revenue and service level can be further increased by 30 to 50%. Maximum locker capacities are shown to be reasonable and are the highest in areas where there is a large gap between the moment when parcels are dropped off and when they are picked up from parcel points, which are mainly in the city center.

1. Introduction

The concept of crowd-shipping as a solution to last-mile delivery problems has drawn a lot of attention in recent years. Especially in urban areas, traditional last-mile delivery using large delivery vehicles is suffering from road congestion [1], lack of parking spaces [2], and restricted access to certain areas due to pollution regulations. Paradoxically, crowd-shipping is most successful in urban areas due to the higher availability of crowd-shippers [3]. Crowd-shipping is seen as a low-cost, flexible, and mostly sustainable alternative to traditional last-mile delivery systems in which large vehicles are responsible for all deliveries. In crowd-shipping, commuters pick up and deliver a parcel on their pre-existing path, possibly making a small detour.

One of the main operational challenges in crowd-shipping is matching crowd-shippers to parcels that need to be delivered [4]. The quality of such a match is influenced by the detour that the crowd-shipper needs to make to pick up and deliver the parcel, as well as potential time windows that need to be satisfied. Especially when the number of parcels and the number of crowd-shippers is high, finding the optimal matching is challenging, yet important to optimize the service level. Another major challenge that can complicate matching problems is stochasticity in demand (i.e., uncertainty in destination, quantity, and time window) as well as supply (the full itinerary of crowd-shippers is uncertain until they communicate it).

When the origins and destinations of parcels are further apart than those of potential crowd-shippers, finding matches that can directly take the parcels from their origin to their destination can be difficult. Especially in bike-based or pedestrian-based crowd-shipping, the two forms that are considered among the least polluting and with the highest potential (i.e. lower value of time), crowd-shipper trips are usually short whereas distances across the city can be long. In

☆ Area: Production Management, Scheduling and Logistics. This manuscript was processed by Associate Editor Yagiura.

* Corresponding author.
E-mail addresses: patrick.stokkink@epfl.ch (P. Stokkink), jean-francois.cordeau@hec.ca (J.-F. Cordeau), nikolas.geroliminis@epfl.ch (N. Geroliminis).

https://doi.org/10.1016/j.omega.2024.103134
Received 17 October 2023; Accepted 11 June 2024
Available online 20 June 2024

this paper, we consider multi-stage deliveries where parcels can be transported from their origin to their destination in multiple stages and with multiple crowd-shippers. That is, parcels can be stored at intermediate transfer points by a crowd-shipper, from where they can be picked up by another crowd-shipper later. We construct a detailed compensation scheme for crowd-shippers and we consider that crowd-shippers can carry multiple non-identical parcels at the same time. We design a path-based approach that allows us to solve significantly larger instances compared to those considered in the literature. We develop a column-generation approach that allows us to solve realistic instances of the problem in a reasonable amount of time.

Practical applications of this work can be found in urban last-mile delivery systems. Crowd-shippers can access parcel lockers through a mobile application to pick up or drop off a parcel. Through the same mobile application, they can announce their itinerary which a central operator can then use to determine the assignment of parcels to crowd-shippers and with this the routing of parcels from their origin to their destination. Such a system is therefore easily accessible to crowd-shippers, making participation more appealing.

The rest of this paper is organized as follows. In Section 2 we provide an overview of the relevant literature and highlight the contribution of this paper. In Section 3 we provide a formal definition of the problem at hand and in Section 4 we propose a column generation approach to solve the problem. We describe the case study and the experimental results in Section 5 and we conclude the paper in Section 6.

2. Literature review

The literature on crowd-shipping as a last-mile delivery option has been growing rapidly over the last few years. For a review of the current practice, academic research, and empirical case studies, the reader is referred to Le et al. [5]. Pourrahmani and Jaller [4] study the characteristics of crowd-shipping platforms and provide an overview of operational challenges and research opportunities. Alnaggar et al. [6] review management decisions and platform characteristics that are studied in the literature and compare those to real-world applications. Earlier works on crowd-shipping consider direct deliveries from a depot to the customer, often in parallel with regular drivers in traditional delivery vehicles. A common way to model this is by extending the vehicle routing problem with occasional drivers [7,8]. In larger networks and when potential crowd-shippers perform relatively short trips compared to the origin–destination distance of parcels (i.e., in biker-based or pedestrian-based crowd-shipping), direct deliveries can considerably restrict the service level of such a system. Chen et al. [9] show that using relays in a reverse logistics system can substantially increase the number of successful deliveries.

The literature on crowd-shipping with transfers can be roughly divided into two types of transshipments. On the one hand, there are transfers between crowd-shippers and another mode of transport, usually traditional delivery vehicles [10]. Such transfers are commonly modeled as two-echelon systems [11]. Kafle et al. [12] consider crowd-shippers performing first-leg pickups or last-leg deliveries, with relays to trucks performing the middle leg. Several variants of the two-echelon delivery system with crowd-shippers have been introduced, such as mobile satellites [13], parcel lockers [14,15] and delivery options [16]. Others have considered two-echelon systems with transfers to mobile depots [17] and public transport [18] rather than a traditional delivery vehicle.

On the other hand, there are transfers among the crowd-shippers themselves. Our work can be classified in this second type of literature. This can again be divided into two groups of studies. One with transfers taking place at dedicated transfer locations with, for example, parcel lockers [19] and one with time-synchronized transfers, where parcels are transferred directly from one crowd-shipper to another and cannot be left unattended [20]. The latter is highly similar to what is classified by Agatz et al. [21] as multi-hop ride-sharing. Multi-hop ride-sharing has received considerably more attention [22–26]. We also note the similarity with public transport modeling, where passengers can make stops and transfers when traveling through a public transport network [27]. The most important difference between multi-hop ride-sharing and multi-stage crowd-shipping is the fact that passengers incur opportunity costs when making detours and transfers and when they are waiting at transfer points. Parcels, on the other hand, are more flexible and can make large detours with various transfers as long as they arrive on time.

Chen et al. [20] allow for transfers between crowd-shippers but require time synchronization such that parcels are directly passed on from one to another crowd-shipper. In their approach, a parcel cannot be left unattended. Sampaio et al. [28] consider a crowd-shipping system with a single transfer at a dedicated transfer point, where parcels can be stored temporarily. As their crowd-shippers do not have predetermined paths, their problem is similar to a pickup and delivery problem with transfers [29,30]. The itinerary of crowd-shippers is considered by Voigt and Kuhn [31], but they do not consider time windows for crowd-shippers nor parcels. Such a system is clearly less attractive for potential crowd-shippers that wish to deliver a parcel during their commute, where time windows are imposed. Such a system is considered by Yıldız [32], who develop a dynamic programming approach to solve their problem. The authors later extend this problem by considering stochasticity in demand [33]. Their crowd-shippers are inflexible and do not deviate from their routes. As a result, crowd-shippers are paid a fixed compensation. Raviv and Tenzer [19] consider compensations for stopping and handling. In their work, they assume Poisson arrivals of occasional couriers, that have a predetermined sequence of transfer points that they will visit. Based on this assumption, they use a stochastic dynamic programming algorithm to find an optimal policy. Nieto-Isaza et al. [34] take a strategic perspective and focus on finding the optimal locations for mini-depots that function as transshipment points. Pugliese et al. [35] consider transfers between two types of crowd-shippers: long-distance crowd-shippers and short-distance crowd-shippers in an urban area. Thanks to this classification, they can more easily model transfers. Our work lies at the intersection of these two subgroups, as our methods allow to consider transfers at dedicated transfer locations as well as time-synchronized transfers.

We model our problem using paths through a network. A common approach to solve such a path-based formulation is to use column generation [36]. Column generation has been used in a broad set of applications [37], among which are several variants of the pickup-and-delivery problem [38,39]. In the crowd-shipping literature, column generation has been used by Torres et al. [40] to solve a vehicle routing problem with a stochastic supply of crowd-shippers. The authors also use column generation to solve a crowd-shipping problem with stochastic destinations [41]. This approach allows us to consider only feasible paths and partially relegate the intricate interactions between parcels and crowd-shippers to the subproblem. In this way, our method is highly scalable, allowing us to solve significantly larger problems than those considered in the literature.

In this work, we propose a general framework that allows the incorporation of both time-synchronized transfers as well as transfers with intermediate storage at transfer points. To the best of our knowledge, this is the first model that can capture both types of transfers simultaneously. In addition to this, we consider the original itinerary of crowd-shippers including their departure times, but we consider some flexibility in their routing decisions. This makes crowd-shipping accessible to daily commuters. On top of this, we consider a detailed compensation scheme for crowd-shippers, which includes rewards for stops, detours, and the inconvenience of carrying a parcel for a longer distance. Furthermore, we consider heterogeneous crowd-shippers and parcels. We propose a column-generation approach to solve our problem. Our results are evaluated on a realistic large-scale case study in the city of Washington DC.

3. Problem description and formulation

In Section 3.1 we introduce the main concepts and notation used in the paper before providing a mathematical formulation of the problem in Section 3.2.

3.1. Concepts and notation

We consider a set P of parcels that make up the considered demand requests. Every parcel $p \in P$ has an origin o_p, a destination d_p and a delivery time window $[e_p, l_p]$, where e_p is the earliest delivery time and l_p is the latest. Every delivered parcel p generates revenue, which can vary between parcels, and is denoted by ρ_p. The set C contains all (potential) crowd-shippers. Every crowd-shipper $c \in C$ has an origin o_c, a destination d_c, and a trip starting time at t_c. Crowd-shippers may be willing to deviate from their shortest path with a maximal detour of τ_c. The detour can be measured either in units of distance or units of time.

Definition 1. A **route** is the trajectory a crowd-shipper traverses to get from his/her origin to his/her destination. The route may either be the shortest path between origin and destination or may deviate from this shortest path with a maximal detour of τ_c.

In this work, we consider a static assignment problem where all parcels and crowd-shippers are known when matching and routing decisions are made. Such settings are realistic when considering next-day delivery (such that all parcel requests are known) and if crowd-shippers are required to sign up their commute to the system a day in advance. Dynamic systems where crowd-shippers are able to sign up during the day require (re-)scheduling in a rolling-horizon framework that captures the stochastic arrivals of crowd-shippers. Such a system is outside the scope of this work, but constitutes an important direction of future research.

Based on their route, a crowd-shipper c is able to execute a set of delivery segments S_c. Fig. 1 illustrates a network with a crowd-shipper traveling from A to D with its original path, marked in green, being $A \rightarrow B \rightarrow D$. The crowd-shipper can also travel through the blue path $A \rightarrow B \rightarrow C \rightarrow D$ within his maximum detour. Based on these two paths, the list of segments for this crowd-shipper is: $[AB, AC, AD, BC, BD, CD]$. Based on the crowd-shipper's start time, we can compute the time at which the crowd-shipper starts the segment, which is given by t_s. A segment also has an origin o_s and a destination d_s. A crowd-shipper $c \in C$ is rewarded w_{cs} for traversing a segment $s \in S_c$. This cost is made up of three components:

1. A fixed compensation α_c^1 for the inconvenience of pickup and delivery;
2. A variable compensation based on the detour crowd-shipper $c \in C$ makes to perform the delivery on segment $s \in S_c$, denoted by $\alpha_c^2 \cdot det_{cs}$;
3. A variable compensation based on the time/distance spent carrying the parcel, which is equal to the length of the segment and denoted by $\alpha_c^3 \cdot len_s$.

Using this compensation scheme, a crowd-shipper receives a fixed compensation for every parcel carried, and a variable compensation proportional to the effort they put in. This depends on the detour they make from their original path and the total time spent carrying this parcel. If the same crowd-shipper carries more than one parcel, they receive full compensation for each of these parcels.

Definition 2. A **segment** is a part of a crowd-shipper's route between two nodes in the network during which he/she can carry a parcel. Every segment corresponds to a unique crowd-shipper and has an origin and destination node and a start time at which the crowd-shipper commences with traversing the segment. The origin and destination of the segment may differ from the origin and destination of the crowd-shipper.

A parcel can be transferred between crowd-shippers at a set H of transfer points or transfer hubs. After the parcel is dropped off at the transfer point by a crowd-shipper, the next crowd-shipper can pick up the parcel at least Δ^{min} time units later (a safety margin) and at most Δ^{max} time units later (to avoid the parcel staying at the hub for too long). We note that by choosing the set H of points to be arbitrarily large and Δ^{max} arbitrarily small, this corresponds to direct transfers where parcels are handed directly from one crowd-shipper to another. Otherwise, parcel lockers need to be present at transfer hubs for crowd-shippers to temporarily store the parcels. Generally, this may differ across transfer points $h \in H$ and we allow Δ_h^{min} and Δ_h^{max} to vary.

The objective is to maximize the profit consisting of the revenue for delivered parcels minus the costs of paying crowd-shippers. For this, we determine the optimal matching of parcels to crowd-shippers. Specifically, for the multi-stage delivery problem, we determine the exact path a parcel traverses from its origin to its destination. This path may be direct or through transfer points and by using multiple crowd-shippers. To this end, we define the concept of a parcel path.

Definition 3. A **parcel path** is the trajectory a parcel traverses to get from its origin to its destination. A parcel path is made up of one or more segments that a parcel travels with a crowd-shipper. Between segments, a parcel is stored at a transfer point.

In the next section, we give a formulation of the problem based on this concept of parcel paths. The approach we take to solve the problem is described in Section 4.

3.2. Mathematical formulation

We first give a full formulation of the problem described above. This is a path-based formulation that maximizes the revenue collected by parcel deliveries minus the costs of crowd-shipper compensation. The full set of parcel paths is denoted by K, where K_p is the set of parcel paths that correspond to parcel $p \in P$. Only feasible parcel paths (i.e., paths that are fully connected and time-synchronized, and for which the time windows of the delivery are satisfied) are included in the set K. The binary decision variable x_k is equal to 1 if parcel path $k \in K$ is selected and 0, otherwise. We define a_{ck} as a binary parameter that is equal to 1 if crowd-shipper $c \in C$ is involved in parcel path $k \in K$. For completeness, we also introduce binary parameter b_{csk}, which is equal to 1 if crowd-shipper $c \in C$ contributes to parcel path $k \in K$ by performing segment $s \in S_c$ and 0 otherwise. Although this parameter is only indirectly part of the formulation of the problem through the defined profit of a parcel, it is required for the solution approach. Clearly, following the definition of a segment, $a_{cs} = \sum_{s \in S_c} b_{csk}$.

The profit of a parcel path $k \in K_p$ is defined as π_k and is defined as follows:

$$\pi_k = \rho_p - \left[\sum_{c \in C} a_{ck} \alpha_c^1 + \sum_{c \in C} \sum_{s \in S_c} b_{csk} (\alpha_c^2 det_{cs} + \alpha_c^3 len_s) \right]. \tag{1}$$

Here, the first term captures the revenue obtained by delivering the parcel p corresponding to the column $k \in K_p$. We emphasize that incorporating the cost of unsatisfied demand is equivalent to the lost revenue of delivery. The second and third term capture the compensation scheme defined in Section 3.1. The second term is the fixed price paid to a crowd-shipper for making a delivery. This does not depend on the segment and therefore only uses parameter a_{ck}. The third term is a variable cost paid to a crowd-shipper which depends on the segment and is therefore based on b_{csk}. This term captures the cost per unit of detour and cost per unit traveled with a parcel.

The formulation of the problem is as follows:

$$\max \sum_{p \in P} \sum_{k \in K_p} \pi_k x_k \tag{2}$$

$$\sum_{k \in K_p} x_k \leq 1 \qquad \forall p \in P \tag{3}$$

Fig. 1. Illustration of a crowd-shipper traveling from A to D that can perform segments: AB, AC, AD, BC, BD, CD. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$$\sum_{k \in K} a_{ck} x_k \leq 1 \qquad \forall c \in C \tag{4}$$

$$x_k \in \mathbb{B} \qquad \forall k \in K. \tag{5}$$

The objective (2) is to maximize the total profit. By substituting Eq. (1) we observe the dependency on parameters a_{cs} and b_{csk}. Constraints (3) ensure that every parcel is delivered at most once and therefore only one parcel path can be selected among those associated with that parcel. Constraints (4) ensure that a crowd-shipper is used at most once.

4. Methodology

We solve the problem using a column generation approach, where every column is a unique parcel path. A column generation approach is commonly used for routing and path-planning problems where the total number of routes or paths is exponentially large. For example, Archetti et al. [42] use a column generation approach for the split-delivery vehicle routing problem, Faiz et al. [43] use a column generation approach for vehicle scheduling and routing problems, and Borndörfer et al. [44] use a column generation approach for line planning in a public transport system. In the master problem, parcel paths from the current set of columns $\bar{K}$ are selected to maximize revenue and minimize operational costs, by solving the LP relaxation of the Restricted Master Problem (RMP). In the pricing problem, new columns are generated that improve the current solution, based on the dual variables of the constraints of the last iteration of the LP. Finally, when no more columns with positive reduced cost are found the optimal solution to the LP is obtained. We then obtain an integer solution by solving the IP with the last set of obtained columns. We note that this does not guarantee the optimality of the IP solution. An exact method would require embedding the column generation in a branch-and-price framework. However, in our computational experiments, the optimality gap of the IP and LP objectives indicates that the obtained solutions are (near) optimal.

The master problem is described in Section 4.1 and the pricing problem is described in Section 4.2. The shortest path problem that is used to solve the pricing problem is described in Section 4.3.

4.1. Master problem

The formulation of the master problem closely resembles the formulation in Section 3.2. In the master problem, we select the best columns from the current set $\bar{K}$ that maximize the obtained revenue from delivering parcels minus the costs of crowd-shippers. In addition to the total set of columns, we define $\bar{K}_p$ as the set of columns that correspond to parcel paths of parcel $p \in P$. It follows that $\bigcup_{p \in P} \bar{K}_p = \bar{K}$. The formulation of the master problem is as follows, with the dual variables of the constraints in parentheses.

$$\max \sum_{p \in P} \sum_{k \in \bar{K}_p} \pi_k x_k \tag{6}$$

$$\sum_{k \in \bar{K}_p} x_k \leq 1 \qquad \forall p \in P \quad (v_p) \tag{7}$$

$$\sum_{k \in \bar{K}} a_{ck} x_k \leq 1 \qquad \forall c \in C \quad (u_c) \tag{8}$$

$$x_k \in \mathbb{B} \qquad \forall k \in \bar{K}. \tag{9}$$

4.2. Pricing problem

We extend the set of columns in the RMP by finding columns with positive reduced cost. The reduced cost for a new column $k \in K \setminus \bar{K}$ is defined as r_k and it can be computed as:

$$r_k = \pi_k - v_p - \sum_{c \in C} u_c a_{ck}. \tag{10}$$

We can rewrite this by substituting π_k from Eq. (1), as follows:

$$r_k = \rho_p - \sum_{c \in C} a_{ck} \alpha_c^1 - \sum_{c \in C} \sum_{s \in S_c} b_{csk} (\alpha_c^2 det_{cs} + \alpha_c^3 len_s) - v_p - \sum_{c \in C} u_c a_{ck}. \tag{11}$$

We can then rewrite this by grouping together similar terms:

$$r_k = (\rho_p - v_p) - \sum_{c \in C} a_{ck} (u_c + \alpha_c^1) - \sum_{c \in C} \sum_{s \in S_c} b_{csk} (\alpha_c^2 det_{cs} + \alpha_c^3 len_s). \tag{12}$$

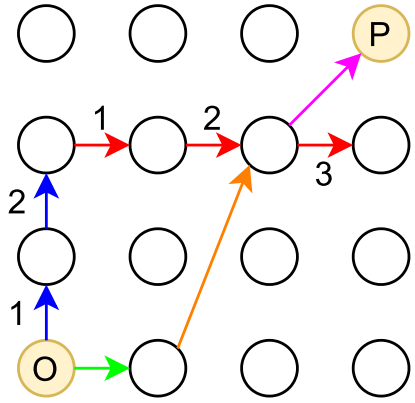
Recall that $a_{cs} = \sum_{s \in S_c} b_{csk}$ and that the total compensation paid to a crowd-shipper is denoted by w_{cs}. We can further simplify the definition of the reduced cost as follows:

$$r_k = (\rho_p - v_p) - \sum_{c \in C} \sum_{s \in S_c} b_{csk} (u_c + \alpha_c^1 + \alpha_c^2 det_{cs} + \alpha_c^3 len_s) \tag{13}$$

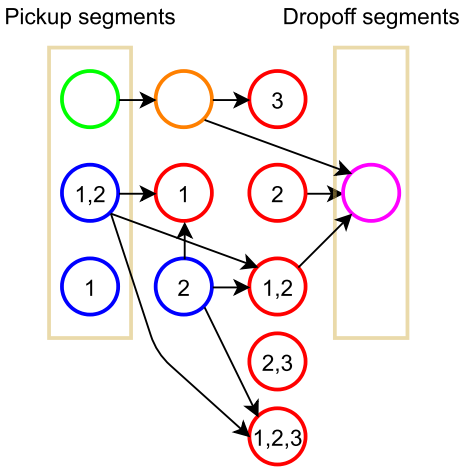
$$r_k = (\rho_p - v_p) - \sum_{c \in C} \sum_{s \in S_c} b_{csk} (u_c + w_{cs}). \tag{14}$$

From Eq. (14) it is clear that finding a column with positive reduced cost can be decomposed over the parcels. For every parcel, we search the parcel path with the highest reduced cost (if any column with positive reduced cost exists). This is done by finding the best crowd-shippers and segments to constitute a feasible path from origin to destination. This path has to satisfy basic flow constraints as well as timing restrictions to ensure that a parcel can only be picked up after it is delivered. As the problem is separated over parcels, the term $\rho_p - v_p$ is fixed. Finding a path with maximal reduced costs is then equivalent to minimizing $\sum_{c \in C} \sum_{s \in S_c} b_{csk} (u_c + w_{cs})$. This means that finding the positive reduced cost path is equivalent to solving the shortest path problem.

We consider a layered procedure for the pricing problem where direct, indirect paths with a single transfer, and indirect paths with multiple transfers are considered separately. This procedure is presented in Algorithm 1. First, direct paths are generated. Direct paths constitute a simple match of a crowd-shipper to a parcel. Here, the feasibility with respect to time windows and location needs to be verified and the costs are computed. Thereafter, indirect paths are generated. Although slightly more difficult due to time and location synchronization at the transfer, this can still be done by simply enumerating for every parcel all crowd-shippers that can pick up and all crowd-shippers that can deliver the parcel. Finally, we consider multi-stage deliveries by solving a shortest-path problem. As the number of transfers is not fixed, this is more complicated and discussed in detail in the remainder of this section.



(a) Network where O is the origin of the parcel, P is the parcel destination, colours indicate a crowd-shipper and numbers are used to identify parts of the route.



(b) Graph for the pricing problem corresponding to the network in (a). Nodes correspond to segments and arcs connect segments if one can be feasibly executed after the other (timing constraints are ignored in this example). Numbers refer to parts of the route that together form a segment.

Fig. 2. Conversion from network with 1 parcel and 5 crowd-shippers, each with a maximum detour of 0, to a graph for the pricing problem. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

This layered procedure has two main benefits. First, solving the pricing problem for direct delivery and indirect delivery with one transfer is computationally much faster. For a direct delivery, finding a column with a positive reduced cost only requires going over all feasible matches of crowd-shippers and parcels, which can be done in $\mathcal{O}(|P||C|)$. For an indirect delivery with one transfer, a similar approach is used where every crowd-shipper is considered twice (once for pickup and once for delivery), which can be done in $\mathcal{O}(|P||C|^2)$. Therefore, the column generation algorithm can be warm-started first for direct deliveries and then also for indirect deliveries with one transfer, before considering the computationally more expensive multi-stage deliveries. The second benefit is that, by considering multi-stage deliveries separately, the shortest path problem and the corresponding graph can be fully adapted to this type of delivery and therefore improve the speed of the algorithm.

Algorithm 1: Layered procedure for pricing problem

```
for every parcel p ∈ P do
    Generate a direct path with positive reduced costs.
Compute r̄; the maximum reduced cost across all generated paths
if r̄ ≤ 0 then
    for every parcel p ∈ P and every pickup segment s ∈ N_p do
        Generate an indirect path with one transfer with positive reduced costs.
    Compute r̄; the maximum reduced cost across all generated paths
    if r̄ ≤ 0 then
        for every parcel p ∈ P and every pickup segment s ∈ N_p do
            Generate an indirect path with positive reduced costs, by solving the shortest path problem.
Add all generated paths with positive reduced costs to K̄
```

4.3. Shortest path algorithm — Graph construction

To solve the shortest path problem, a graph is constructed based on the movement of crowd-shippers through the road network. An example of such a graph is given in Fig. 2 and will be described below. The road network with five potential crowd-shippers and one parcel is depicted in Fig. 2(a) and the corresponding graph is depicted in Fig. 2(b). The shortest path problem is solved on a directed graph where nodes correspond to segments. Whenever a node is part of the shortest path, the variable b_{csk} is equal to 1 and it is equal to 0 otherwise. The cost of such a node is equal to $u_c + w_{cs}$, such that the length of the shortest path corresponds to the second term of the reduced cost in Eq. (14). An arc between two nodes exists if the two segments are compatible, in the sense that one segment can be executed right after the other. In Fig. 2(b), such an arc is depicted by a black arrow. An arc between two nodes n_1 and n_2 exists if all of the following conditions hold:

- The crowd-shipper of node n_1 is different from the crowd-shipper of node n_2.
- The segment of node n_1 ends at the same transfer point where the segment of node n_2 starts.
- The segment of node n_1 finishes at least Δ^{min} time units before and at most Δ^{max} time units after the segment of node n_2 starts.

All existing arcs have a cost of zero, which means that the only cost components are on the nodes. For the multi-stage delivery problem we consider three types of nodes each corresponding to a type of segment: pickup nodes/segments, dropoff nodes/segments and transfer nodes/segments. We describe the properties of these nodes in detail below, with the set of nodes of each type in parentheses. A feasible parcel path starts with a pickup segment and ends with a dropoff segment, possibly with one or more transfer segments in between. A segment describes a part of the parcel path for which the parcel is traveling with the same crowd-shipper.

1. **Pickup nodes/segments (N_P):** A pickup segment represents the initial pickup of the parcel from the origin location and its delivery to a transfer point. A pickup node exists if the origin of the segment coincides with the origin of the parcel and the destination of the segment coincides with a transfer point. Thereby, it only exists if the start time of the segment is later

than the earliest availability time of the parcel. A pickup node has no incoming arcs. In the example in Fig. 2(b), the three left-most nodes correspond to pickup nodes.

2. **Dropoff nodes/segments (N_D):** A dropoff segment represents the final delivery of the parcel from the last transfer point to the destination of the parcel. A delivery node exists if the destination of the segment coincides with the destination of the parcel and the origin of the segment coincides with a transfer point. Thereby, it only exists if the time window of the parcel is satisfied. A dropoff node has no outgoing arcs. In the example in Fig. 2(b), the single right-most node corresponds to a dropoff node.
3. **Transfer nodes/segments (N_T):** A transfer segment represents the transfer of any parcel from one transfer point to another. There are no restrictions on location or time for the existence of a transfer node.

We emphasize that although pickup and dropoff nodes are parcel-specific, due to origins, destinations, and time windows, transfer nodes are not. Therefore, transfer nodes are only added once, whereas pickup and dropoff segments may be repeated for multiple parcels that are similar.

4.4. Modified Dijkstra's algorithm

To find the column to add to the master problem for every parcel, we aim to find the shortest path between any pickup segment and any dropoff segment. We do this by applying a modified version of Dijkstra's shortest path algorithm tailored to fit well the specifics of our problem. As Dijkstra's algorithm can find the shortest path from a source node to any node in the graph, we apply the shortest path problem $|N_P|$ times. The column with the highest reduced cost (if any column with positive reduced cost exists) is added to the master problem and this is repeated for every parcel.

Dijkstra's algorithm takes as an input a set of nodes and an adjacency matrix which defines the arcs between the nodes. We observe that the full graph does not change between iterations and can therefore be pre-computed once. Then, at each call to the pricing problem, only the costs on the nodes are updated according to the dual variables. The details on the algorithm are described in Algorithm 2. The algorithm enforces all constraints that hold between nodes through the adjacency matrix, as these constraints are transitive. The only exception to this is that two segments belonging to the same crowd-shipper may be included in the shortest path, as long as at least one other segment is in between. This constraint is not enforced as a hard constraint as this would make the problem resource-constrained. However, by construction of our problem, such paths are never feasible if $\Delta^{min} > 0$. As a crowd-shipper will leave directly after dropping off the parcel, whereas a parcel can only be transferred after Δ^{min} time units, the crowd-shipper will arrive at the next transfer point at least Δ^{min} time units before the parcel arrives with another crowd-shipper. As crowd-shippers never wait for a parcel to become available in our framework, these paths are implicitly eliminated.

The algorithm is initialized by marking all nodes as unvisited (Line 5) and setting the distance to each node at infinity, except the source node (Line 6). The next node to be visited is the one with minimal distance (Line 8). In our modified version of Dijkstra's algorithm we have an additional termination criterion. In Line 11, the algorithm is terminated because there exist no remaining *unvisited* nodes that can be visited through a feasible path from the source node. In Line 13, we skip the for-loop in Lines 14–16 whenever the current node is in N_D as this is by definition the last node on a path and therefore cannot be on the shortest path to another node. Otherwise, the distances to all unvisited nodes are updated in Lines 14–16.

In addition to the modifications to Dijkstra's algorithm, more computational enhancements are made to improve the speed of the algorithm. We consider three enhancements that allow to retain the optimality of the algorithm and one enhancement that does not guarantee optimality.

Algorithm 2: Modified Dijkstra's Algorithm

1 **Input:** A set of nodes $N = N_P \cup N_D \cup N_T$ with their costs $c(n)$ for all $n \in N$
  For every node $n \in N$, a set of neighboring nodes $A(n)$
  A source node s
4 **Output:** A set of shortest paths from source node $s \in N_P$ to all nodes in N_D
5 Mark all nodes as *unvisited*: $visit(n) \rightarrow$ **false**
6 Set the shortest distance to each node at $dist(n) \rightarrow \infty$, except for the source node which is set to $dist(s) \rightarrow c(s)$
7 **while** *Not all nodes are visited* **do**
8 Find node q^- as the unvisited node with minimal $dist(q^-)$
9 Set $visit(q^-) \rightarrow$ **true**
10 **if** $dist(q^-) = \infty$ **then**
11 **return** shortest paths
12 **if** $q^- \in N_D$ **then**
13 **continue** to next node
14 **for** $q^+ \in A(q^-)$ **do**
15 **if** $visit(q^+) =$ ***false and*** $dist(q^-) + c(q^+) < dist(q^+)$ **then**
16 $dist(q^+) \rightarrow dist(q^-) + c(q^+)$
17 **return** shortest paths

4.4.1. Removing suboptimal nodes and arcs

For some nodes and arcs, we can immediately see that they will not be part of the shortest path because the cost on the node is too high or the joint costs of two nodes connected by an arc is too high. Propositions 1 and 2 identify several of these cases where nodes and arcs can be eliminated from the graph. This also leads to identifying parcels for which the pricing problem does not need to be solved because no column with positive reduced cost exists for that parcel. By eliminating nodes and arcs, the size of the graph can be reduced, which improves the speed of the shortest path algorithm.

Proposition 1 (*Disregarding Nodes*). *Let* $\underline{\rho} = \min_{p \in P} \rho_p$ *and* $\underline{w} = \min_{c \in C, s \in S_c} w_{cs}$. *A parcel* $p \in P$ *can be disregarded if* $\rho_p - v_p - 2\underline{w} \leq 0$. *The corresponding pickup and dropoff segments (nodes) can then also be disregarded. A segment (node)* $s \in S_c$ *of crowd-shipper* $c \in C$ *can be disregarded if* $\underline{\rho} - u_c - w_{cs} - \underline{w} \leq 0$ *or* $\underline{\rho} - u_c - w_{cs} - 2\underline{w} \leq 0$ *if* s *is a transfer segment (node).*

Proposition 2 (*Disregarding Arcs*). *Let* $\underline{\rho} = \min_{p \in P} \rho_p$ *and* $\underline{w} = \min_{c \in C, s \in S_c} w_{cs}$. *An arc between two nodes* $s_1 \in S_{c_1}$ *of crowd-shipper* $c_1 \in C$ *and* $s_2 \in S_{c_2}$ *of crowd-shipper* $c_2 \in C$ *can be disregarded if* $\underline{\rho} - u_{c_1} w_{c_1 s_1} - u_{c_2} w_{c_2 s_2} \leq 0$ *or* $\underline{\rho} - u_{c_1} w_{c_1 s_1} - u_{c_2} w_{c_2 s_2} - \underline{w} \leq 0$ *and either* s_1 *or* s_2 *is a transfer segment.*

As we consider multi-stage deliveries, a parcel path consists of at least two segments (i.e., a pickup and a dropoff segment). In case the considered segment is a transfer segment, there are at least two other segments involved. Using this property, the proof of these propositions is straightforward.

4.4.2. Constructing several smaller subgraphs

The nodes of the considered graph of the shortest path problem are partitioned into three categories: pickup nodes (N_P), dropoff nodes (N_D) and transfer nodes (N_T). Transfer nodes are independent of the specific parcels and only depend on crowd-shippers' itineraries. Pickup and dropoff nodes, for their part, depend on the specific parcel through the origin, destination and delivery time window. Constructing separate graphs, hereafter referred to as subgraphs, that only include a part of the pickup and dropoff nodes can solve memory issues, at the cost of a slight increase in computation time. Transfer nodes, finally, need to be included in every subgraph to guarantee the optimality of the solution.

We consider a fraction $\eta \in (0, 1]$ of the parcels that are included in a subgraph. This means that $1/\eta$ subgraphs are constructed for which the pricing problems are solved separately. Basically, the value of η forms a trade-off between time-savings and memory-savings, as well as the number of subgraphs and the size of those subgraphs. When η is small, subgraphs are small and therefore do not lead to memory issues, but many subgraphs need to be constructed at the cost of extra computation time. When η is large, subgraphs are larger, which may lead to memory issues, but fewer subgraphs need to be constructed which is generally faster.

4.4.3. Randomly removing highly similar nodes

Whereas the aforementioned enhancements improve the speed of the algorithm and reduce the memory consumption without jeopardizing optimality, we now turn to a method that can very successfully reduce the size of the graph but can no longer guarantee optimality. Due to the nature of our problem, many of the segments (and therefore nodes in the graph) are highly similar and therefore likely unnecessary. For example, many transfer segments between the same two transfer hubs may exist, but with different crowd-shippers at slightly different times. For this reason, many of those nodes can be removed without influencing optimality. However, as we do not know in advance whether such a node will be in a shortest path or not, optimality can no longer be guaranteed. We maintain a fraction $\zeta \in [0, 1)$ of the nodes in the graph and remove the other $1 - \zeta$ (and the arcs connected to those nodes). These nodes are selected randomly and with equal probability. As this is repeated at every iteration of the column generation algorithm, different nodes can be removed across iterations. This limits the influence on optimality, yet maintains the goal of reducing the size of the graph.

4.5. Locker and shipper capacity

So far, we have assumed that lockers have an infinite capacity and that crowd-shippers can only carry a single parcel. In this section, we relax those assumptions and extend the formulation accordingly. This will come at the cost of increased complexity in both the master problem and the pricing problem but will lead to more realistic solutions.

4.5.1. Shipper capacity

Instead of assuming that a crowd-shipper can only make a single delivery, we now relax this assumption and allow crowd-shippers to make multiple deliveries. We assume that crowd-shippers only perform multiple pickups and deliveries if they involve the exact same itinerary. That is, a crowd-shipper may carry multiple parcels at the same time, but only if they are picked up and delivered at the exact same stations. The reason for this is that significant effort is involved with every pickup and delivery (i.e., stopping at a locker, collecting or storing the parcel and continuing the journey). Whereas this can be largely consolidated if the pickup and drop-off locations are the same for the different items, this is not the case if these locations are different. We denote the capacity of a crowd-shipper $c \in C$ by Q_c.

To efficiently model the capacity, we duplicate every segment in S_c a total of Q_c times. We redefine the set S_c by introducing S_c^q with $1 \leq q \leq Q_c$ as the qth copy of the set of segments and $S_c = \cup_{q=1}^{Q_c} S_c^q$. For the sake of notation, let $s_1 \sim s_2$ denote the property that segments s_1 and s_2 are copies of each other and $s_1 \nsim s_2$ the fact that they are not copies. Then, we reformulate problem (2)–(5) by replacing Constraints (4) by the following set of constraints, which ensures the capacity of a crowd-shipper:

$$\sum_{k \in K} a_{ck} x_k \leq Q_c \qquad \forall c \in C \quad (u_c). \tag{15}$$

In addition to this, we add the following set of constraints to enforce that only segments that are duplicates of each other are performed by the same crowd-shipper. A pair of segments that are not duplicates of each other are deemed *incompatible* and columns that cannot be used together because of such an incompatibility are part of an incompatible set I. The full collection of incompatible sets is denoted as $\mathcal{I}^{\text{crowd}}$ with $I \in \mathcal{I}^{\text{crowd}}$. Let b_{Ik} be a binary parameter taking value 1 if parcel path k uses a segment that is part of incompatible set I, and 0 otherwise. Basically, the set I contains all paths that are incompatible because they contain one of two incompatible segments s_1 and s_2 for which it holds that $s_1 \in S_c$ and $s_2 \in S_c$ for some crowd-shipper $c \in C$ and $s_1 \nsim s_2$. Every set I, therefore, corresponds to a pair of incompatible segments (s_1, s_2). The following set of constraints is added to exclude incompatibilities, with dual variable δ_I for every constraint $I \in \mathcal{I}^{\text{crowd}}$:

$$\sum_{k \in K} b_{Ik} x_k \leq 1 \qquad \forall I \in \mathcal{I}^{\text{crowd}} \quad (\delta_I). \tag{16}$$

Instead of adding all constraints, which is computationally impossible due to the large number of segments, we only add those constraints that are violated in the current solution. We can still guarantee optimality as satisfied constraints do not influence the solution or the objective function. Given that they are inactive, their dual variable is by definition equal to 0 and therefore this also does not influence the pricing problem. The procedure to identify violated constraints is as follows. We define in advance all possible combinations of segments that would constitute a violation. That is, we identify all possible $I \in \mathcal{I}$. Then, every time the master problem is solved, for all the newly added columns we verify whether they contain a segment that is in any $I \in \mathcal{I}$. For every violation $I \in \mathcal{I}$ we maintain the set of columns that contain any segment in this set. We note that the violation $I \in \mathcal{I}$ is only added to the master problem if the corresponding set of columns contains more than one column.

The new reduced cost then looks as follows, where we identify if parcel path k contains a segment that makes it part of any of the incompatible sets $I \in \mathcal{I}$:

$$r_k = \pi_k - v_p - \sum_{c \in C} u_c a_{ck} - \sum_{I \in \mathcal{I}^{\text{crowd}}} b_{Ik} \delta_I. \tag{17}$$

The pricing problem remains the same apart from an extra cost δ_I that is subtracted whenever the new column is part of an incompatible set. We emphasize that the computational complexity of the pricing problem remains unchanged after the addition of the capacity constraint. Even though we duplicate the number of segments by the capacity, only the duplicate segment s with the lowest value of $\sum_{I \in \mathcal{I}^{\text{crowd}}} b_{Ik} \delta_I$ is considered in the pricing problem. The reason for this is that the duplicate segments are identical. Therefore, a segment s' with a higher sum of dual variables can never be in the shortest path, as replacing it with segment s would always reduce the cost of the path.

4.5.2. Locker capacity

In our framework, we allow parcels to be stored in parcel lockers at the transfer point. So far, we assumed that parcel lockers had infinite capacity. Here, we limit the number of parcels that can be stored at a transfer point $h \in H$ to be $\bar{Q}_h$. Similar to the capacity of the crowd-shippers, we identify sets of columns that are incompatible because the capacity of a locker is exceeded at some point in time. The full collection of incompatible sets is denoted as $\mathcal{I}^{\text{locker}}$. We adapt the master problem by adding the same set of constraints as in (16), but for the new collection:

$$\sum_{k \in K} b_{Ik} x_k \leq 1 \qquad \forall I \in \mathcal{I}^{\text{locker}} \quad (\delta_I). \tag{18}$$

Again, we do not add all constraints at once but identify those that are violated. The capacity of the transfer point needs to be considered at every time interval. We only consider transfer points with transfer lockers, as direct time-synchronized transfers do not need to be stored and therefore do not influence the capacity. To identify the violated constraints, we use the following procedure. For every transfer point,

we identify the parcel paths that store a parcel at this point. We sort the parcel paths twice: once in ascending order of their arrival time at the transfer point and once in ascending order of their departure time from the transfer point. We start with an empty set of paths V. We then go over those events one by one in chronological order. Every time an arrival is recorded, the parcel path is added to V. Every time a departure is recorded, the parcel path is removed from V. Whenever a parcel arrives that causes the cardinality of V to exceed $\bar{Q}_h$, we add violation I with $b_{Ik} = 1$ for all $k \in V$ and we store the time t_{Ih} at which the violation occurs, which will later aid the pricing problem. For every transfer point, we only add a single constraint and then re-solve the master problem. This is repeated until no violated constraints are encountered.

The reduced cost can then be computed as follows, where we identify if a parcel path k stores a parcel at transfer point h at time t_{Ih} for any of the incompatibilities $I \in \mathcal{I}^{\text{locker}}$:

$$r_k = \pi_k - v_p - \sum_{c \in C} u_c a_{ck} - \sum_{I \in \mathcal{I}^{\text{crowd}}} b_{Ik}\delta_I - \sum_{I \in \mathcal{I}^{\text{locker}}} b_{Ik}\delta_I. \tag{19}$$

If this is the case, $b_{Ik} = 1$ for the new parcel path. The pricing problem can then be extended by exploiting the start and end times of every segment. We recall that every node in the network discussed in Section 4.2 corresponds to a segment. So far, a node corresponding to a segment $s \in S_c$ of crowd-shipper $c \in C$ was attributed a cost $u_c + w_{cs}$, and no costs were attributed to arcs. Now, for an arc between two nodes corresponding to segment s_1 with end time $\underline{t}$ and s_2 with start time $\bar{t}$ where $d_{s_1} = o_{s_2} = h$ a cost of δ_I is added for every $I \in \mathcal{I}^{\text{locker}}$ for which it holds that $\underline{t} \leq t_{Ih} \leq \bar{t}$. We note that if an arc violates multiple constraints, multiple dual variables can be added to the same arc.

5. Results

We describe the details of the case study and the parameter settings in Section 5.1. In Section 5.2 we evaluate the performance of the algorithm in terms of optimality gap and computation time. We evaluate the effect of crowd-shipper capacity and locker capacity in Sections 5.4 and 5.5, respectively. Finally, we perform a sensitivity analysis on the cost parameters in Section 5.6.

5.1. Case study

The city of Washington DC is used as a case study. We use data on the spatial distribution of the population [45] and the movement of individuals throughout the city based on bike-sharing users [46]. The bike-sharing system of Washington DC has over 500 stations and 4500 bikes, making it one of the largest in the USA. A selection of 240 stations are used in our case study. For this, all stations in the city center have been selected combined with those in the closest suburbs (mainly consisting of Georgetown and Columbia Heights). Bike-sharing stations are considered as demand locations. This can either be through parcel lockers or home delivery to an individual living arbitrarily close to a station. Thereby, historical data on the movement of bike-sharing users throughout the city is used to approximate the movement of potential crowd-shippers.

The case study and the construction of the dataset are highly similar to that of Stokkink and Geroliminis [47]. The main difference is that the size of the network we consider in this work is more than three times as large. Thereby, we consider time-dependent arrival rates of crowd-shippers. For a detailed description of how the case study is constructed, the reader is referred to their work. Fig. 3 displays a bubble chart of the considered network, where the size of the bubble is determined relative to the population around the corresponding station. Whereas most commuters travel around Union Station, the Mall, and the center of Washington DC, most people live in the suburbs and this is therefore where demand is the highest. We note the large asymmetry in supply and demand for a crowd-shipping system in an urban network,

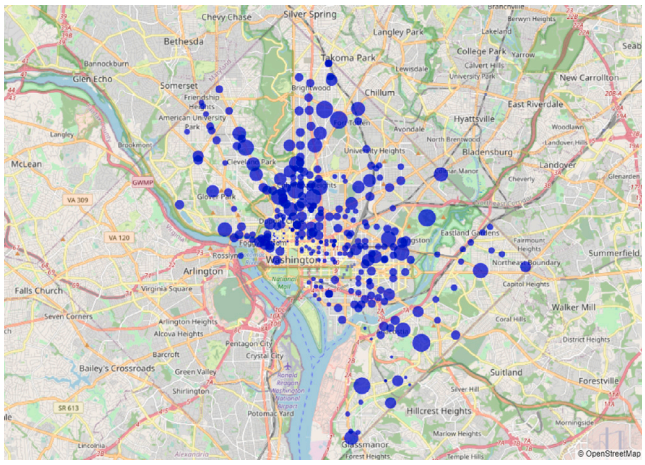


Fig. 3. Bubble chart of bike-sharing stations, where the size of the bubble is determined by the population in the area.

making our case study highly realistic. We emphasize that, although our methodology is able to capture direct transfers from one crowd-shipper to another, in this case study we only consider transfers where the parcel is temporarily stored in a parcel locker. The reason for this is that direct transfers can be subject to more practical issues regarding privacy, security, and timeliness, which are outside the scope of this work.

The baseline parameters used for the model and the column generation algorithm are given in Table 1. These parameters are used in all numerical experiments, except for sensitivity analyses on these parameters. According to an analysis from American survey data in [48], on average, crowd-shippers expect a compensation of 12$ per hour. Considering 10 min to perform both the pickup and delivery, we set $\alpha^1 = \$2$. Using an average bikers speed of 12 km/h, we set $\alpha^3 = \$1$/km, which is similar to the value chosen by Le et al. [49]. Intuitively, $\alpha^2 > \alpha^3$ and therefore we set $\alpha^2 = \$2$/km. This is in line with the findings of Rougès and Montreuil [50], who studied 26 crowd-shipping businesses, that found the prices of intra-urban deliveries to start between $4 and $10 plus additional charges for inconveniences such as heavy loads and long distances. According to Le and Ukkusuri [48], a traditional carrier charges $15 per parcel. To accommodate distance aspects, we set the cost per parcel to a base cost of $10, which can increase up to $15 with $2.00 per kilometer between the origin and destination of the parcel. The maximum runtime of the algorithm is set to 1800 s. The maximum runtime is checked before every call to the pricing problem and may therefore be slightly exceeded.

The base case we consider has two origin locations and we construct a subgraph for every origin in the pricing problem. This means $\eta = 1/2$. Parcels are stored at a random origin in the morning and not necessarily at the closest origin to the destination. The relative rate of parcels and crowd-shippers ($|C|/|P|$) is fixed. For computational reasons, we reduce the set C by removing crowd-shippers that cannot contribute to any delivery (complete or partial). This yields the reduced set C'. The number of crowd-shippers in $|C'|$ depends on other parameters such as the transfer locations H and the maximum detour τ. Therefore, in the experiments that follow, the reported ratio $|C'|/|P|$ is not necessarily constant.

CPLEX version 12.6.3.0 is used in Java to solve all ILPs and LPs. The LPs during the iterations of the column generation algorithm are solved to optimality and the IP after the final iteration of the column generation algorithm is solved up to a 0.5% optimality gap.

5.2. Algorithm evaluation

In this section, we evaluate the performance of our column-generation algorithm in terms of objective value and computation

Table 1
Parameter settings.

Model parameters	
α^1	\$1.00/parcel
α^2	\$2.00/parcel/km
α^3	\$1.00/parcel/km
$\Delta^{\min}$	1 min
$\Delta^{\max}$	10 h
ρ	min{\$15.00, \$10 + \$2.00/km}/parcel
Algorithm parameters	
η	0.5
ζ	0.3
CPU time limit	1800 s

time. We evaluate the performance for various model parameters and problem sizes. Thereby, we compare the performance of the algorithm for multiple levels of ζ. The results are displayed in Table 2. Clearly, the computation time of the algorithm increases as the size of the problem increases. The most important determinant of the complexity of the algorithm is the number of segments that are used to construct the graph in the pricing problem. Therefore, the computation time increases drastically with $|P|, |C|$, and τ. This also explains why using only a random portion of the segments to construct the graph in every iteration leads to a significant reduction in computation time. By using a portion ζ, the computation time is reduced almost by a factor 10. Hence, larger instances can be solved without decomposing the pricing problem over more subgraphs.

The algorithm finds optimal or near-optimal solutions. When ζ is 1 and the algorithm converges before the time limit, we can use the LP solution as an upper bound to the objective value and therefore compute an optimality gap. For $\zeta < 1$, the LP solution is not necessarily an upper bound. Hence, we only compute the optimality gap if the optimal LP solution is found for $\zeta = 1$. The optimality gap is at most 0.5% for all tested instances for which the optimality gap was computable. Even when $\zeta = 0.3$, the optimality gap is almost negligible. Furthermore, using transfers leads to an improvement between 15% and 50% both in the objective value (i.e., revenue - costs) and the service level (i.e., number of served parcels).

To further evaluate the effect of ζ on computation time and optimality gap, we evaluate the case where $|P| = 681$, $|C'| = 1202$, $\tau = 500$, and $|H| = 11$ for 6 different values of ζ. In Fig. 4, the optimality gap is displayed relative to the number of iterations (left) and the computation time in seconds (right). Clearly, the computation time per iteration decreases drastically by decreasing the random portion of segments that are used at every iteration. However, because the subgraphs are not complete, they may lead to not all columns with positive reduced costs being identified in an iteration. Therefore, the algorithm may require more iterations and can lead to suboptimal solutions. The best value of ζ is thus a trade-off between the number of iterations and the computation time per iteration. The best value is also dependent on the size of the problem. In general, for larger problems, smaller values of ζ can be chosen at the cost of limited losses.

In general, the number of segments is the component that has the biggest influence on computation time. Increasing the size of the network does not have a direct effect on the computation time of our algorithm. However, as a larger network generally corresponds to a higher number of parcels, crowd-shippers, and transfer points, this will in turn lead to a higher number of segments and therefore increase the computation time of the algorithm. Following from this, for larger networks the computation time can be reduced by reducing the value of ζ, without leading to a substantial deterioration of the optimality gap.

5.3. Performance compared to locally optimized benchmark

In this section, we compare the performance of our optimized assignment procedure to a myopic first-best assignment policy. Such a policy is commonly applied for settings with incomplete information and without coordination [51–53].

For this benchmark, crowd-shippers are sorted based on their departure time. For every crowd-shipper the parcel that is locally optimal is assigned, without considering future crowd-shippers. For a direct delivery, the profit can be computed exactly as the full trip is known. For an indirect delivery, the costs of the current delivery stage are known. The costs of previous delivery stages have already been incurred and can be considered sunk costs, which are therefore omitted from the optimization. It is assumed that after the current stage, the parcel is directly picked up from the transfer point and taken to the final destination of the parcel. Due to coordination issues in the dynamic arrival of crowd-shippers, we only allow for two-stage deliveries and strictly prefer delivering a parcel to the final destination over delivery to a transfer point. With these cost components and the revenue obtained from delivering the parcel, the expected profit can be computed. Since the strategy is myopic, no information is used on potential future crowd-shippers.

We note that for the myopic first-best assignment policy, parcels may remain at transfer points whereas for the optimized assignment this is not possible. In case a parcel remains at the transfer point, the revenue is not obtained although a part of the costs is already incurred. For the sake of comparison, we compare the optimized assignment to two dynamic benchmarks. One where the costs for uncompleted deliveries are excluded (B1) and one for which the costs for uncompleted deliveries are included (B2).

The results are displayed in Table 3 where the first set of rows denotes the results for $|H| = 0$, implying that only direct deliveries are allowed and the second set of rows denotes the results for $|H| = 11$ where transfers are allowed. Global optimization allows for the coordination of transfers. As a consequence, global optimization outperforms local optimization by 25% in terms of service level and objective value when transfers are allowed. Without transfers, the effect is only 5%.

5.4. Crowd-shipper capacity

In this section, we evaluate the influence of crowd-shipper capacity on the profit and service level. We consider that a part of the potential crowd-shippers can carry multiple parcels at the same time. Whereas some crowd-shippers can only carry small parcels in a backpack, others may have a small basket in the back or front which allows them to carry several parcels at the same time. When generating the instance, every crowd-shipper has an equal probability for each capacity level, such that we obtain an evenly distributed population. The results are obtained for $\zeta = 0.05$, to further reduce the CPU time. Here, we also apply the described row generation procedure to identify violated constraints that we iteratively add to the formulation. As a result of using $\zeta = 0.05$, the obtained LP solution is not necessarily optimal. Therefore, the optimality gap is an approximation.

The results are displayed in Table 4. By considering higher capacities, computation times increase drastically. For this reason, a time limit of 3600 s (1 h) is used instead. The reason for the increased computation times is two-fold. First, we consider duplicate segments, such that the number of considered segments and therefore the computation time of the pricing problem increases. Second, the violated constraints need to be identified and added to the master problem, which makes solving the master problem more computationally demanding. Furthermore, we observe that the optimality gap increases with capacity. However, the highest observed optimality gap is 9%, which is deemed reasonable.

We note that to obtain an optimal solution the column generation framework would have to be integrated into a branch-and-price framework. However, given the relatively small optimality gap, the

Table 2
Algorithm evaluation.

$\|P\|$	$\|C'\|$	$\|H\|$	τ	ζ	CPU time (s)	Opt. gap (%)	Obj. ($)	Gain (%)	SL	Gain (%)
310	339	6	250	1	2.4	0.0	1094.33	30.7	32.7	35.4
310	491	6	500	1	8.7	0.0	1475.85	24.6	31.0	28.8
681	777	6	250	1	25.0	0.0	2715.32	18.0	34.6	20.1
681	1028	6	500	1	387.3	0.4	3762.00	13.2	37.5	16.3
1043	1165	6	250	1	79.8	0.1	3988.24	14.7	34.3	17.0
1043	1531	6	500	1	[a]1800.0	–	5461.77	12.5	36.7	15.9
310	442	11	250	1	4.5	0.1	1178.63	40.7	27.1	46.3
310	577	11	500	1	45.5	0.2	1636.59	38.1	29.8	45.8
681	976	11	250	1	53.7	0.0	2823.08	22.6	28.8	25.4
681	1202	11	500	1	1456.3	0.1	4033.96	21.4	34.9	26.9
1043	1447	11	250	1	520.4	0.2	4195.71	20.6	29.2	23.4
1043	1774	11	500	1	[a]1800.0	–	5860.22	20.7	34.5	26.2
310	339	6	250	0.3	0.9	0.0	1094.26	30.7	32.7	35.4
310	491	6	500	0.3	1.9	0.0	1475.81	24.6	31.0	28.8
681	777	6	250	0.3	4.1	0.0	2716.18	18.0	34.6	20.1
681	1028	6	500	0.3	89.7	0.4	3763.67	13.3	37.5	16.3
1043	1165	6	250	0.3	16.0	0.0	3989.80	14.7	34.3	17.0
1043	1531	6	500	0.3	200.2	–	5453.03	12.3	36.6	15.7
310	442	11	250	0.3	2.2	0.0	1179.20	40.8	27.1	46.3
310	577	11	500	0.3	10.1	0.4	1633.30	37.9	29.6	44.9
681	976	11	250	0.3	10.1	0.0	2823.58	22.7	28.8	25.4
681	1202	11	500	0.3	230.7	0.5	4018.04	20.9	34.7	26.0
1043	1447	11	250	0.3	36.5	0.1	4200.88	20.8	29.2	23.7
1043	1774	11	500	0.3	991.9	–	5869.61	20.9	34.6	26.6

Note: $|P|$ = number of parcels, $|C'|$ = number of potential crowd-shippers, τ = maximum detour of crowd-shippers, $|H|$ = number of transfer hubs, ζ is the portion of random segments used in the construction of the subgraph for the pricing problem. The optimality gap is the percentage difference between the IP solution and the LP solution for $\zeta = 1$. SL = service level. The gain columns display the improvement that is obtained by using transfers over not using transfers. The two largest instances for $\zeta = 1$ cannot be solved due to memory issues.
[a] Denotes scenarios for which the CPU time limit is reached and therefore no optimality gap can be obtained.

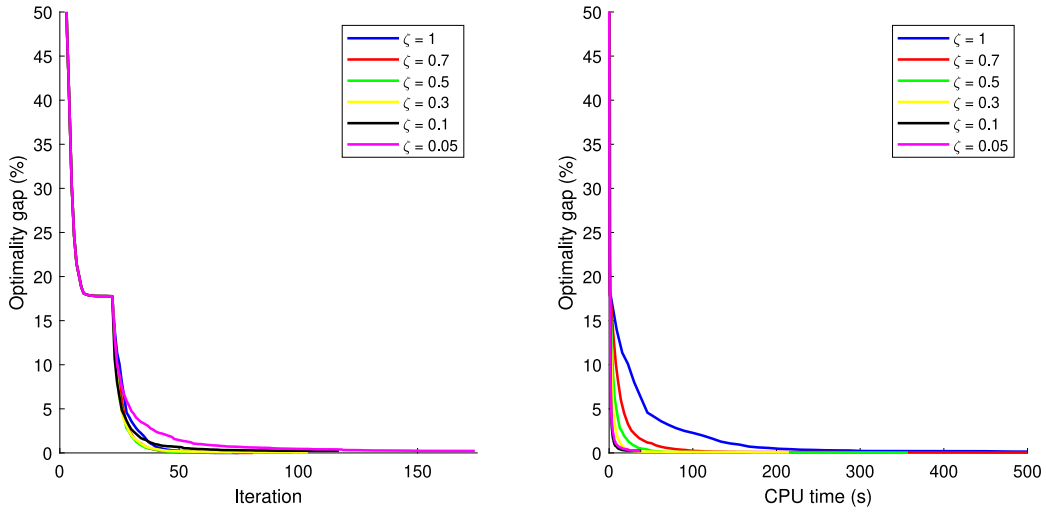


Fig. 4. Iterative optimality gap for different values of ζ. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

already substantial computation time of the column generation algorithm, and the fact that we are dealing with an operational problem rather than a strategic one, developing a branch-and-price framework looks unappealing for our specific problem.

The increase in capacity leads to a substantial improvement in the objective (profit) and the service level. Depending on the problem setting, using a capacity of 2 for half of the population improves the objective and service level by 20% to 50%. For higher capacities, the observed increase is even higher, even though the algorithm has reached the time limit before the optimal solution has been found.

Fig. 5 displays a Gantt chart of the movement of parcels. Each colored bar represents the time spent with a crowd-shipper. Identical bars in identical locations signal that a crowd-shipper is carrying multiple parcels at the same time. A deeper investigation reveals that the additional flexibility leads to parcels being carried collectively on one leg and separately on the other, which is clear from the zoomed figure. We also observe the influence of travel patterns on crowd-shipping activity. We observe a clear morning and evening peak, by the frequency of the activities. The evening peak contains significantly more activities, despite the number of potential crowd-shippers not being significantly different from the morning commute. The reason for this is that most destinations for parcels are in the suburbs. Hence, the evening commute from the center to the suburbs is more useful for reaching these destinations.

5.5. Parcel locker capacity

As parcel locker capacity does not seem to be a restrictive parameter for reasonable values of $\bar{Q}_h$, these constraints are not considered in the obtained results. In this section, we discuss the evolution of locker

Table 3
Benchmark comparison.

$\|P\|$	$\|C'\|$	τ	$\|H\|$	Global optimization		Local optimization			Effect of global optimization		
				SL	Obj	SL	B1	B2	SL	B1	B2
310	442	250	0	26.5	837.40	25.2	797.75	797.75	−4.9	−4.7	−4.7
310	577	500	0	38.1	1184.80	37.1	1144.06	1144.06	−2.5	−3.4	−3.4
681	976	250	0	32.9	2301.94	32.3	2251.91	2251.91	−1.8	−2.2	−2.2
681	1202	500	0	48.6	3322.31	45.5	3107.11	3107.11	−6.3	−6.5	−6.5
1043	1447	250	0	32.8	3477.75	31.4	3328.68	3328.68	−4.1	−4.3	−4.3
1043	1774	500	0	46.5	4856.33	44.9	4650.70	4650.70	−3.5	−4.2	−4.2
310	442	250	11	38.7	1179.20	29.4	905.14	864.79	−24.2	−23.2	−26.7
310	577	500	11	55.2	1633.30	40.6	1208.52	1098.46	−26.3	−26.0	−32.7
681	976	250	11	41.3	2823.58	34.5	2367.28	2298.17	−16.4	−16.2	−18.6
681	1202	500	11	61.2	4018.04	48.3	3198.28	3044.50	−21.1	−20.4	−24.2
1043	1447	250	11	40.6	4200.88	33.3	3471.00	3379.15	−18.0	−17.4	−19.6
1043	1774	500	11	58.9	5869.61	47.2	4768.41	4522.80	−19.9	−18.8	−22.9

Note: $|P|$ = number of parcels, $|C'|$ = number of potential crowd-shippers, τ = maximum detour of crowd-shippers, $|H|$ = number of transfer hubs, SL = service level given as a percentage, obj = objective value given in dollars, B1 and B2 are the objective values of two local optimization benchmarks given in dollars. The last three columns denote the percentual difference between the local and the global optimization strategies.

Table 4
Influence of crowd-shipper capacity.

$\|P\|$	$\|C'\|$	q_c	τ (m)	CPU time (s)	Obj. ($)	SL (%)	$\widetilde{\text{Opt. gap}}$ (%)	Gain obj. (%)	Gain SL (%)
310	442	{1}	250	2.2	1179.20	38.7	0.0	–	–
310	442	{1,2}	250	132.4	1559.52	52.3	2.5	32.3	35.0
310	442	{1,2,3}	250	176.8	1661.95	55.2	8.0	40.9	42.5
310	577	{1}	500	8.6	1633.30	55.2	0.4	–	–
310	577	{1,2}	500	664.0	2008.52	68.4	2.8	23.0	24.0
310	577	{1,2,3}	500	1642.1	2158.09	73.2	5.3	32.1	32.7
681	976	{1}	250	10.0	2823.58	41.3	0.0	–	–
681	976	{1,2}	250	3433.3	4025.81	60.2		42.6	45.9
681	976	{1,2,3}	250	[a]3600.0	4431.95	66.1		57.0	60.1
681	1202	{1}	500	188.2	4018.04	61.2	0.5	–	–
681	1202	{1,2}	500	[a]3600.0	4079.37	59.2		1.5	−3.4
681	1202	{1,2,3}	500	[a]3600.0	–	–	–	–	–

Note: $|P|$ = number of parcels, $|C'|$ = number of potential crowd-shippers, q_c is the considered crowd-shipper capacity, each with equal probability, τ = maximum detour of crowd-shippers, The optimality gap is the percentage difference between the IP solution and the LP solution. Since we use $\zeta = 0.3$, the optimality gap is not exact but an approximation. SL = service level. The gain columns display the improvement that is obtained by increasing the capacity. The largest instance cannot be solved due to memory issues.

[a] Denotes scenarios for which the CPU time limit is reached and therefore no optimality gap can be obtained.

capacity over time. Out of the 11 transfer hubs, we specifically focus on three locations that have distinct patterns. The locations are identified in Fig. 6(b), where origins are marked in red, transfer points are marked in yellow, destinations of parcels are marked in green and the flow of parcels that make at least one transfer is marked by blue lines. Here, the size of the line denotes the number of parcels. The total number of parcels stored in the transfer points is displayed in Fig. 6(a).

The three chosen transfer points are the most used among a total of 11. It is clear that a capacity of 10, therefore, suffices for all transfer points. The first two points are in the city center. This is clear because they fill up quickly during the morning commute after which they are emptying out slowly during the evening commute. During the morning commute, potential crowd-shippers travel from the suburbs to the city center, passing by these transfer points. The opposite is observed for the third transfer point, which is at the main train station of Washington DC, parcels gradually accumulate throughout the day before being emptied out rapidly during the evening commute when people are traveling back home (i.e., to the suburbs) from the train station. Clearly, the results in Fig. 6 align with the results in Fig. 5.

We emphasize that even though in this case study the parcel locker capacity is not restrictive, it may be different for other case studies depending on the distribution of the destinations of parcels and the distribution of origin–destination pairs of crowd-shippers. In some cases, parcel lockers may need higher or lower capacities depending on the frequency of pickups and dropoffs over time.

5.6. Sensitivity for cost parameters

In this section, we evaluate the effect of the cost parameters on the observed performance and the number of transfers per parcel path. In this way, we can consider scenarios where more emphasis is placed on service level rather than costs, as well as scenarios where potential crowd-shippers are more or less sensitive to detour and distance traveled with a parcel. We consider similar settings as in the previous experiment, but with a constant $|H| = 11$ and $\zeta = 0.3$. We consider non-linear cost components for crowd-shipper compensation where we replace the profit in Eq. (1) with the following function:

$$\pi_k = \rho_p - \left[\sum_{c\in C} a_{ck}\alpha^1 + \sum_{c\in C}\sum_{s\in S_c} b_{csk}\left(\alpha^2(det_{cs})^{\beta_2} + \alpha^3(len_s)^{\beta_3}\right)\right]. \tag{20}$$

The results are displayed in Table 5. We observe that the effect of transfers on service level is relatively constant for different cost parameter combinations. The effect on the objective improvement is more substantial. For higher values of τ, the relative improvement of the objective function compared to the case $|H| = 0$ is lower than for lower values of τ. Thereby, if the penalty for distance traveled with a parcel is non-linear, the improvement of the objective decreases by approximately 10%. The value of α_1 has a significant influence on the number of transfers on a path. When the fixed compensation is negligible, transfers become more beneficial and we observe significantly more paths with two or more transfers.

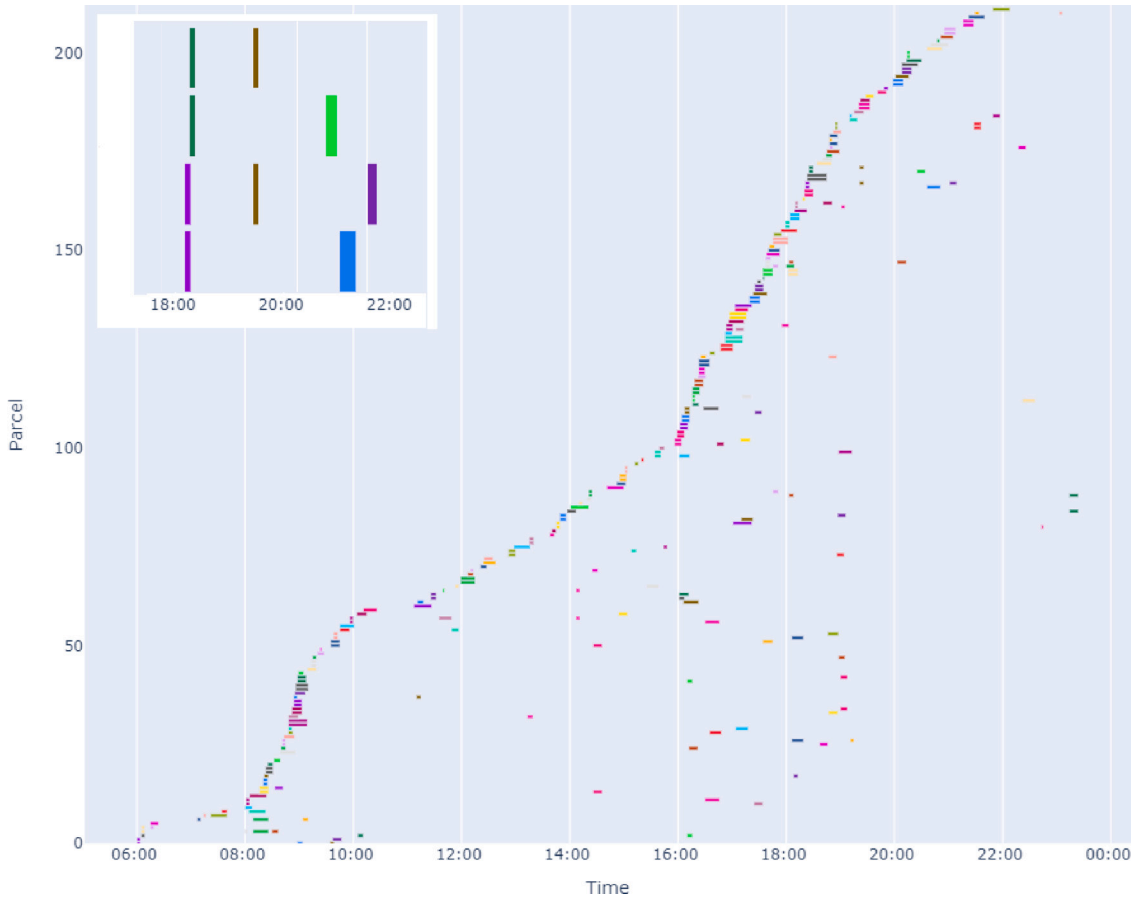


Fig. 5. Gantt chart of the movement of parcels with time on the x-axis and the parcel index on the y-axis. Parcels are sorted by the start time of the first segment. Each colored bar represents the time spent with a crowd-shipper. In the upper left corner, we zoom on four specific parcels. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

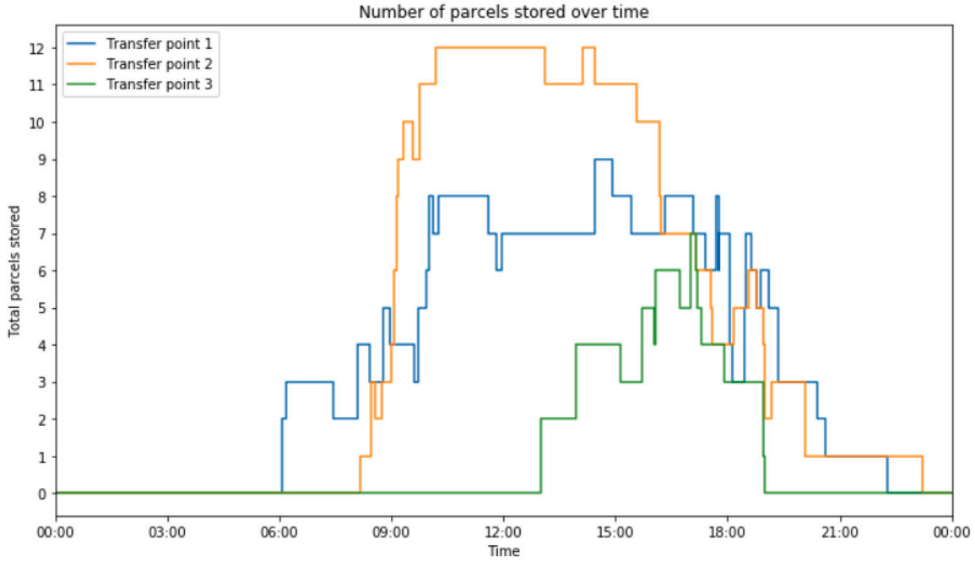
The economic sustainability of the designed crowd-shipping system, in comparison to other last-mile delivery systems, is also highly dependent on these cost parameters. A detailed analysis of this is outside the scope of this work, as it depends on many parameters of the crowd-shipping system and the traditional delivery system. For example, slightly changing the size of the parcels or the capacity of the truck can drastically influence the number of trucks that is needed and with that the full solution. However, a couple of useful managerial insights can be drawn from the results in this paper. First, the results in Section 5.2 show the presence of economies of scale with respect to supply and demand. This suggests that crowd-shipping is more likely to be economically sustainable if a critical mass of demand and, most importantly, potential crowd-shippers is reached. Clearly, this is closely related to the number of parcels that a crowd-shipper is willing to carry, as illustrated in Section 5.4. Second, due to the reliance of the performance on supply and demand, crowd-shipping is likely to be profitable in some parts of the network where supply is high, but not in other parts where supply is low, as also previously indicated by Stokkink and Geroliminis [47]. To properly evaluate this, traditional delivery and crowd-shipping need to be solved in an integrated framework where for every parcel the decision is made to either deliver it through a crowd-shipping system or through traditional delivery. This can also be anticipated through a dynamic pricing scheme, which is an important direction of future research.

6. Conclusion

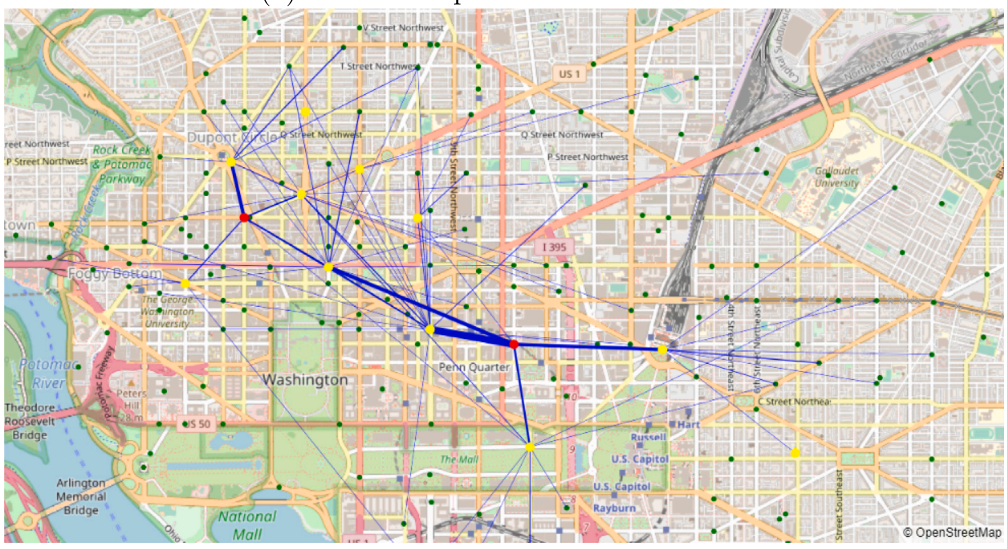
In this paper, we developed a crowd-shipping model with intermediate transfers. In contrast with the majority of the existing literature, our model allows for high levels of heterogeneity of crowd-shippers, parcels, and transfer points. We consider a detailed individual-specific cost structure for crowd-shipper compensation and allow for different weights to be assigned to different parcels, for example, to differentiate between locations in the network. Thereby, we allow for direct time-synchronized transfers, where a parcel is directly handed from one crowd-shipper to another, as well as transfers with intermediate storage at strategically located parcel lockers. We designed a column generation algorithm to solve large-scale realistic scenarios to optimality within a reasonable amount of time.

To improve the performance of the system, we allow crowd-shippers to carry more than one parcel at the same time. This further complexifies the problem, as additional constraints are required to regulate crowd-shipper capacity and compatibility of parcels. To solve this problem, we extend our column generation algorithm to simultaneous column and row generation. This algorithm identifies violated compatibility constraints and adds these to the master problem after every column generation iteration. A similar approach can be taken to enforce constraints on locker capacity, but the results show that in the considered scenarios locker capacity is not restrictive.

We evaluated the performance of our model and algorithm on a realistic case study in the city of Washington DC. Demand for parcels is approximated through the number of inhabitants of a region and the flow of potential crowd-shippers through the network is based on the flow of bike-sharing users. For a large network with 250 regions, 11 transfer points, approximately 500 parcels, and 500 crowd-shippers, our algorithm finds the optimal matching within one minute. For larger models of approximately 1000 parcels and 1000 crowd-shippers, we find solutions that are optimal or near-optimal within 10 to 30 min.



(a) Number of parcels stored over time



(b) Flow of parcels that make at least one transfer

Fig. 6. Step and flow charts that indicate the number of parcels stored at transfer points over time and the flow of parcels in the network. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Computation times increase with the total number of segments. This means that as the number of crowd-shippers, transfer points, and maximum detour crowd-shippers are willing to make increases, the computation time also increases. However, computation time can be significantly improved by randomly reducing the set of segments in every iteration. Although this removes the optimality guarantee, an optimality gap smaller than 0.5% is observed across the tested instances. Due to the complexity of coordination between crowd-shippers in a system with transfers, our optimal approach outperforms a myopic first-best (locally optimized) approach by 25%.

Our results indicate that the use of parcel lockers for intermediate transfers allows for increasing the total revenue and service level by around 30%, depending on the system configurations. A further increase of 30 to 50% can be obtained by allowing some crowd-shippers to carry two or three parcels at the same time (with an average capacity of two across the population). Due to the asymmetric movement of crowd-shippers during the day (i.e., suburbs to the city center in the morning and city center to suburbs in the afternoon), some parcels can be stored for an entire work day in a parcel locker before being transported to their final destination. Due to the increased computation time for the case where crowd-shippers can carry more than one parcel, developing a heuristic solution approach is an important direction of future research.

A comparison between transfer points shows that the location of the transfer point strongly impacts the quality of the obtained solution. The strategic decision of choosing the optimal location of depots and transfer points remains an important direction of future research. In this work, we consider that crowd-shippers that carry two parcels at the same time receive double the compensation. An adaptive pricing strategy and behavioral analysis of the crowd-shippers' response to prices is an important direction of future research. Such pricing policies typically improve the efficiency of the system. However, it also comes at a cost of reduced transparency and lower utilization rates for some crowd-shippers [54].

Finally, we considered a fully static setting for the crowd-shippers and parcel requests, where all crowd-shippers and parcel requests are known before the routing and matching decisions are made (for example, a day in advance). An extension of this work to a dynamic

Table 5
Sensitivity to cost parameters.

$\|P\|$	$\|C'\|$	τ	α_1	α_2	α_3	β_2	β_3	Improvement over $\|H\| = 0$		Transfers per path (%)		
								Obj. (%)	SL (%)	0	1	2+
310	442	250	1	2	1	1	1	40.8	46.3	66.7	33.3	0.0
310	577	500	1	2	1	1	1	38.4	45.8	66.3	32.0	1.7
310	442	250	1	1.6	1	1.2	1	40.3	48.1	65.0	34.2	0.8
310	577	500	1	1.6	1	1.2	1	34.4	46.2	64.9	33.3	1.8
310	442	250	1	2	0.8	1	1.2	29.7	46.3	64.2	34.2	1.7
310	577	500	1	2	0.8	1	1.2	23.1	43.2	68.0	30.8	1.2
310	442	250	1	1.6	0.8	1.2	1.2	28.3	46.9	66.4	32.8	0.8
310	577	500	1	1.6	0.8	1.2	1.2	18.6	32.5	70.3	29.0	0.6
310	442	250	0.01	2	1	1	1	46.2	46.3	56.7	38.3	5.0
310	577	500	0.01	2	1	1	1	43.8	45.8	60.5	33.7	5.8
310	442	250	0.01	1.6	1	1.2	1	46.4	48.1	56.7	38.3	5.0
310	577	500	0.01	1.6	1	1.2	1	41.3	47.0	58.7	34.3	7.0
310	442	250	0.01	2	0.8	1	1.2	39.3	46.3	49.2	42.5	8.3
310	577	500	0.01	2	0.8	1	1.2	33.9	44.1	51.8	38.8	9.4
310	442	250	0.01	1.6	0.8	1.2	1.2	38.0	46.9	53.8	42.0	4.2
310	577	500	0.01	1.6	0.8	1.2	1.2	29.0	41.0	53.9	38.8	7.3

Note: $|P|$ = number of parcels, $|C'|$ = number of potential crowd-shippers, τ = maximum detour of crowd-shippers, α_1 = fixed crowd-shipper compensation in dollars, α_2 = variable crowd-shipper compensation per km detour, α_3 = variable crowd-shipper compensation per km traveled with parcel, β_2 = power of detour component, β_3 = power of distance component. SL = service level. α_1, α_2, and α_3 are given in dollars (per kilometer).

framework where crowd-shippers gradually arrive throughout the day marks an interesting direction of future research. In this case, the assignment problem of parcels to crowd-shippers can be (re-)optimized in a rolling horizon framework combined with a dynamic programming approach. Here, we also note the added value of the integration of a machine-learning approach to predict the existence of future crowd-shippers.

CRediT authorship contribution statement

Patrick Stokkink: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Jean-François Cordeau:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Investigation, Formal analysis. **Nikolas Geroliminis:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Investigation, Formal analysis, Conceptualization.

Declaration of competing interest

None

Data availability

The used data is freely accessible online.

Acknowledgments

The authors would like to thank the Editor and three anonymous referees for their constructive comments on an earlier version of this paper.

References

[1] Arnott R, Rave T, Schöb R, et al. Alleviating urban traffic congestion, Vol. 1, MIT Press Books; 2005.
[2] Shoup DC. Cruising for parking. Transp Policy 2006;13(6):479–86.
[3] Ermagun A, Shamshiripour A, Stathopoulos A. Performance analysis of crowd-shipping in urban and suburban areas. Transportation 2020;47(4):1955–85.
[4] Pourrahmani E, Jaller M. Crowdshipping in last mile deliveries: Operational challenges and research opportunities. Soc-Econ Plan Sci 2021;78:101063.
[5] Le TV, Stathopoulos A, Van Woensel T, Ukkusuri SV. Supply, demand, operations, and management of crowd-shipping services: A review and empirical evidence. Transp Res C 2019;103:83–103.
[6] Alnaggar A, Gzara F, Bookbinder JH. Crowdsourced delivery: A review of platforms and academic literature. Omega 2021;98:102139.
[7] Archetti C, Savelsbergh M, Speranza MG. The vehicle routing problem with occasional drivers. European J Oper Res 2016;254(2):472–80.
[8] Macrina G, Di Puglia Pugliese L, Guerriero F, Laganà D. The vehicle routing problem with occasional drivers and time windows. In: International conference on optimization and decision science. Springer; 2017, p. 577–87.
[9] Chen C, Pan S, Wang Z, Zhong RY. Using taxis to collect citywide E-commerce reverse flows: a crowdsourcing solution. Int J Prod Res 2017;55(7):1833–44.
[10] Macrina G, Pugliese LD, Guerriero F, Laporte G. Crowd-shipping with time windows and transshipment nodes. Comput Oper Res 2020;113:104806.
[11] Laporte G, Nobert Y. A vehicle flow model for the optimal design of a two-echelon distribution system. In: Advances in optimization and control. Springer; 1988, p. 158–73.
[12] Kafle N, Zou B, Lin J. Design and modeling of a crowdsource-enabled system for urban parcel relay and delivery. Transp Res B 2017;99:62–82.
[13] Lan Y-L, Liu F, Ng WW, Gui M, Lai C. Multi-objective two-echelon city dispatching problem with mobile satellites and crowd-shipping. IEEE Trans Intell Transp Syst 2022.
[14] Enthoven DL, Jargalsaikhan B, Roodbergen KJ, Uit het Broek MA, Schroten-boer AH. The two-echelon vehicle routing problem with covering options: City logistics with cargo bikes and parcel lockers. Comput Oper Res 2020;118:104919.
[15] dos Santos AG, Viana A, Pedroso JP. 2-echelon lastmile delivery with lockers and occasional couriers. Transp Res E 2022;162:102714.
[16] Vincent FY, Jodiawan P, Redi AP. Crowd-shipping problem with time windows, transshipment nodes, and delivery options. Transp Res E 2022;157:102545.
[17] Mousavi K, Bodur M, Roorda MJ. Stochastic last-mile delivery with crowd-shipping and mobile depots. Transp Sci 2022;56(3):612–30.
[18] Kızıl KU, Yıldız B. Public transport-based crowd-shipping with backup transfers. Transp Sci 2023;57(1):174–96.
[19] Raviv T, Tenzer EZ. Crowd-shipping of small parcels in a physical internet. Workingpaper, Tel Aviv University; 2018.
[20] Chen W, Mes M, Schutten M. Multi-hop driver-parcel matching problem with time windows. Flex Serv Manuf J 2018;30(3):517–53.
[21] Agatz N, Erera A, Savelsbergh M, Wang X. Optimization for dynamic ride-sharing: A review. European J Oper Res 2012;223(2):295–303.
[22] Drews F, Luxen D. Multi-hop ride sharing. In: International symposium on combinatorial search. Vol. 4, 2013.
[23] Herbawi W, Weber M. Evolutionary multiobjective route planning in dynamic multi-hop ridesharing. In: European conference on evolutionary computation in combinatorial optimization. Springer; 2011, p. 84–95.
[24] Masoud N, Jayakrishnan R. A decomposition algorithm to solve the multi-hop peer-to-peer ride-matching problem. Transp Res B 2017;99:1–29.
[25] Chen Y, Guo D, Xu M, Tang G, Zhou T, Ren B. PPtaxi: Non-stop package delivery via multi-hop ridesharing. IEEE Trans Mob Comput 2019;19(11):2684–98.
[26] Lu W, Liu L, Wang F, Zhou X, Hu G. Two-phase optimization model for ride-sharing with transfers in short-notice evacuations. Transp Res C 2020;114:272–96.
[27] Spiess H, Florian M. Optimal strategies: a new assignment model for transit networks. Transp Res B 1989;23(2):83–102.
[28] Sampaio A, Savelsbergh M, Veelenturf LP, Van Woensel T. Delivery systems with crowd-sourced drivers: A pickup and delivery problem with transfers. Networks 2020;76(2):232–55.
[29] Mitrović-Minić S, Laporte G. The pickup and delivery problem with time windows and transshipment. INFOR Inf Syst Oper Res 2006;44(3):217–27.

[30] Rais A, Alvelos F, Carvalho MS. New mixed integer-programming model for the pickup-and-delivery problem with transshipment. European J Oper Res 2014;235(3):530–9.
[31] Voigt S, Kuhn H. Crowdsourced logistics: The pickup and delivery problem with transshipments and occasional drivers. Networks 2022;79(3):403–26.
[32] Yıldız B. Express package routing problem with occasional couriers. Transp Res C 2021;123:102994.
[33] Yıldız B. Package routing problem with registered couriers and stochastic demand. Transp Res E 2021;147:102248.
[34] Nieto-Isaza S, Fontaine P, Minner S. The value of stochastic crowd resources and strategic location of mini-depots for last-mile delivery: A benders decomposition approach. Transp Res B 2022;157:62–79.
[35] Pugliese LD, Guerriero F, Macrina G, Scalzo E. Crowd-shipping and occasional depots in the last mile delivery. In: Optimization and decision science: ODS, virtual conference, November 19, 2020. Vol. 7, Springer; 2022, p. 213.
[36] Desrosiers J, Lübbecke ME. A primer in column generation. In: Column generation. Springer; 2005, p. 1–32.
[37] Lübbecke ME, Desrosiers J. Selected topics in column generation. Oper Res 2005;53(6):1007–23.
[38] Ropke S, Cordeau J-F. Branch and cut and price for the pickup and delivery problem with time windows. Transp Sci 2009;43(3):267–86.
[39] Ghilas V, Cordeau J-F, Demir E, Woensel TV. Branch-and-price for the pickup and delivery problem with time windows and scheduled lines. Transp Sci 2018;52(5):1191–210.
[40] Torres F, Gendreau M, Rei W. Vehicle routing with stochastic supply of crowd vehicles and time windows. Transp Sci 2022;56(3):631–53.
[41] Torres F, Gendreau M, Rei W. Crowdshipping: An open VRP variant with stochastic destinations. Transp Res C 2022;140:103677.
[42] Archetti C, Bianchessi N, Speranza MG. A column generation approach for the split delivery vehicle routing problem. Networks 2011;58(4):241–54.
[43] Faiz TI, Vogiatzis C, Noor-E-Alam M. A column generation algorithm for vehicle scheduling and routing problems. Comput Ind Eng 2019;130:222–36.
[44] Borndörfer R, Grötschel M, Pfetsch ME. A column-generation approach to line planning in public transport. Transp Sci 2007;41(1):123–32.
[45] Census Reporter. Census reporter. 2021, https://censusreporter.org/. [Accessed: 10 March 2021].
[46] Capital Bikeshare. Capital bikeshare. 2020, https://www.capitalbikeshare.com/system-data.
[47] Stokkink P, Geroliminis N. A continuum approximation approach to the depot location problem in a crowd-shipping system. Transp Res E 2023;176:103207.
[48] Le TV, Ukkusuri SV. Crowd-shipping services for last mile delivery: Analysis from American survey data. Transp Res Interdiscip Perspect 2019;1:100008.
[49] Le TV, Ukkusuri SV, Xue J, Van Woensel T. Designing pricing and compensation schemes by integrating matching and routing models for crowd-shipping systems. Transp Res E 2021;149:102209.
[50] Rougès J-F, Montreuil B. Crowdsourcing delivery: New interconnected business models to reinvent delivery. In: 1st international physical internet conference. 1, 2014, p. 1–19.
[51] Bent R, Van Hentenryck P. Regrets only! online stochastic optimization under time constraints. In: AAAI. Vol. 4, 2004, p. 501–6.
[52] Bent R, Van Hentenryck P. Waiting and relocation strategies in online stochastic vehicle routing.. In: IJCAI. Vol. 7, Citeseer; 2007, p. 1816–21.
[53] Mousavi K, Bodur M, Cevik M, Roorda MJ. Approximate dynamic programming for crowd-shipping with in-store customers. 2021.
[54] Alnaggar A, Gzara F, Bookbinder JH. Compensation guarantees in crowdsourced delivery: Impact on platform and driver welfare. Omega 2024;122:102965.