Design and Modeling of A Crowdsourcing-Enabled System for Urban Parcel Relay and Delivery

Nabin Kafle, Bo Zou¹, Jane Lin
Department of Civil and Materials Engineering, University of Illinois at Chicago, Chicago, IL 60607, United States

Abstract: This paper proposes a crowdsource-enabled system for urban parcel relay and delivery. We consider cyclists and pedestrians as crowdsources who are close to customers and interested in relaying parcels with a truck carrier and undertaking jobs for the last-leg parcel delivery and the first-leg parcel pickup. The crowdsources express their interests in doing so by submitting bids to the truck carrier. The truck carrier then selects bids and coordinates crowdsources’ last-leg delivery (first-leg pickup) with its truck operations. The truck carrier’s problem is formulated as a mixed integer non-linear program which simultaneously i) selects crowdsources to complete the last-leg delivery (first-leg pickup) between customers and selected points for crowdsource-truck relay; and ii) determines the relay points and truck routes and schedule. To solve the truck carrier problem, we first decompose the problem into a winner determination problem and a simultaneous pickup and delivery problem with soft time windows, and propose a Tabu Search based algorithm to iteratively solve the two subproblems. Numerical results show that this solution approach is able to yield close-to-optimum solutions with much less time than using off-the-shelf solvers. By adopting this new system, truck vehicle miles traveled (VMT) and total cost can be reduced compared to pure-truck delivery. The advantage of the system over pure-truck delivery is sensitive to factors such as penalty for servicing outside customers’ desired time windows, truck unit operating cost, time value of crowdsources, and the crowdsource mode.

Keywords: Crowdsource-enabled delivery, bidding, relay, truck routes and schedule, simultaneous pickup and delivery problem, Tabu Search

1 Introduction

The need to rethink urban parcel delivery is never more urgent. E-commerce in the US today has reached the size of $300 billion annual sales, five times ten years ago (Braunstein, 2015). Pushed by the e-commerce explosion, urban truck traffic has increased precipitously. Looking at retail-related truck trips in New York City, it is estimated that each establishment generates an average of 1.89 truck trips per day (Holguin-Veras et al., 2012). Considering that Manhattan has 105,998 establishments (USCB, 2016), the total number of retail-related truck trips exceeds 200,000 per day. The rise of truck traffic has led to many negative consequences on the urban environment, such as congestion, air pollution, wear-and-tear of road infrastructure, and demand-supply imbalance in truck parking space. Take air pollution and parking space shortage as examples. On average, freight vehicles contribute 16-50% of total vehicle emissions in urban areas (Thompson, 2015). In New York City, truck carriers pay $500-1000 parking fines per truck-month (Holguin-Veras et al., 2007; 2008). The payment is even higher in Manhattan, where a truck accumulates $750 weekly parking fines (Rodrigue and Dablanc, 2013). These negative consequences will only become more severe in the future, with e-commerce expected to more than double its size by 2019 (Braunstein, 2015) and characterized by smaller parcels and more frequent and expedited deliveries than today.

¹ Corresponding author.
Email addresses: nkafle3@uic.edu (N. Kafle), bzou@uic.edu (B. Zou), janelin@uic.edu (J. Lin).
In contrast to the push from e-commerce explosion, urban parcel delivery is also pulled by the development of livable urban communities that require significant reduction of truck traffic. Many of the pulls are in the form of city ordinances such as restrictions on truck delivery time (Holguin-Veras et al., 2012), size and weight (Qureshi et al., 2012; Holguin-Veras et al., 2013), and introduction of low emission zones (Browne et al., 2005; Giuliano and Dablanc, 2013). In response, truck carriers have taken a variety of initiatives, including setting up urban consolidation centers (Quak and Tavasszy, 2011; Ville et al., 2013; Morana et al., 2014), deploying truck parking reservation systems (Taniguichi, 2012; PIARC, 2012), and implementing dynamic vehicle routing techniques (Kritzinger, 2012; Wolfe and Troup, 2013).

The present paper contributes to reconciling the above push and pull by designing, modeling, and evaluating a new system empowered by crowdsourcing parcel relay and delivery for the last-mile problem. The term “last mile” is commonly used in city logistics and refers to trips between a depot located at the border of a metro area and customers in the city. The system design is from the perspective of a truck carrier, with focus on operational planning (e.g., day-before). Conceptually, the system functions in five steps: (1) the truck carrier posts online pickup and delivery jobs to attract local crowdsources – in this paper cyclists and pedestrians who are close to customers – to undertake the last leg of the last-mile delivery.2 The last leg is connected to the rest of the last mile at points where parcel relay between trucks and crowdsources occurs. In addition to jobs, all possible relay points are posted online; (2) on seeing the posted jobs, a crowdsourcer generates and submits bids for undertaking a subset of the jobs. Each bid specifies what customers to serve, the bid price, and the point for parcel relay; (3) the truck carrier selects bids and determines truck routes and schedule for visiting the relay points in the selected bids as well as customers not covered by any selected bids; (4) the truck carrier informs crowdsources with winning bids about the time that a truck relays parcels at each relay point; (5) crowdsources and trucks execute delivery based on the bid selection results and the planned routes and schedule (Figure 1).

The core of the proposed system is the substitution of trucks by local crowdsources for the last leg of the last-mile delivery, which is based on the following economic rationale. Traditional truck delivery is not efficient in urban neighborhoods due to constraints imposed by city ordinances and geographic conditions such as limited truck access time and presence of one-way roads. However, the constraints generally do not apply to crowdsourcing cyclists and pedestrians, who have high maneuverability in urban neighborhoods

2 Unless “pickup” is explicitly mentioned, in the rest of the paper the term “delivery” refers to both pickup and delivery. The “last leg of the last-mile delivery” then also covers the first leg of the first-mile pickup.
and low cost in delivery. On the other hand, cyclists and pedestrians can only travel shorter distances, making parcel relay necessary between crowdsources and trucks near customers. In fact, having a truck carry many parcels from the depot to a common destination, e.g., a relay point, keeps delivery cost low in this “line-haul” part due to the economies of load consolidation. Thus by leveraging the advantages of crowdsourcing and truck modes in the last-leg and the remaining part of the last-mile delivery respectively, the proposed system is expected to improve the overall delivery efficiency compared to the pure-truck delivery.

We note a growing popularity of crowdsourced delivery in the real world over the past few years. A number of online platforms (e.g., Postmates and Deliv in the US, Jing Dong in China, and PiggyBaggy in Europe) have been dedicated to facilitating or allows for crowdsourced delivery. Rouges and Montreuil (2014) give a comprehensive documentation of recently emerged business models for crowdsourced delivery. Among them, the dominant model is Business-to-Customer (B2C) delivery where a parcel is picked up directly from a fixed point such as a retail store or a restaurant by a crowdsource and delivered to the customer’s doorstep. Each delivery route is typically associated with a single parcel.

Our proposed system is different from the existing crowdsourced delivery practice in three aspects. First, rather than from fixed pickup points our model considers greater coverage of the logistics chain from the depot to customers, with trucks acting as intermediate moving pickup/delivery points. Second, while the existing practices outsource delivery entirely, our model is “partial” outsourcing because a truck carrier still performs part of the delivery by itself. Thus the truck carrier in our proposed system needs to simultaneously select crowdsources and determine its own truck routes and schedule with the objective of minimizing the overall cost. Third, we consider both facts that multiple crowdsources may compete for one delivery job and a crowdsource may deliver multiple parcels in a single tour, and consequently introduce a combinatorial bidding process to select crowdsources.

Methodologically, this paper makes two major contributions. The first contribution is system design that involves crowdsourcing and parcel relay between crowdsources and trucks. We formulate the design problem as a mixed integer non-linear program (MINLP). Compared to traditional vehicle routing problems, the new formulation integrates crowdsource bids selection and relay points locationing with truck routes design. The second contribution is on the solution approach for the MINLP. In addition to providing a linearized formulation of the MINLP which allows off-the-shelf solvers to solve small-size problems, an efficient algorithm is developed for large-size instances. The basic idea of the algorithm is to decompose the overall design problem into a winner determination problem (WDP) and a simultaneous pickup and delivery problem with soft time windows (SPDPSTW), and iteratively solve the two subproblems based on the Tabu Search concept to improve the overall solution. The effectiveness of the algorithm is validated by comparing solutions with those using off-the-shelf solvers for small-size problems, and further demonstrated by solving problems of larger, realistic sizes.

In addition to the above contributions, we perform extensive numerical experiments to assess the cost savings using the new design compared to pure-truck delivery. Results indicate that at median 9.25% cost saving and 24% truck vehicle miles traveled (VMT) reduction can be achieved. We further find that the advantage of the crowdsource-enabled system over pure-truck delivery is sensitive to factors such as penalty for servicing outside customers’ desired time windows, truck unit operating cost, time value of crowdsources, and the crowdsource mode.

The paper continues with a review of the literature in Section 2 on last-mile delivery that bears relevance to the crowdsource-enabled system. Section 3 provides a conceptual description of the proposed system design, followed by its mathematical formulations in Section 4. The solution approach is presented in Section 5. Section 6 offers numerical experiments and discussions of the results. Summary of major findings and directions for the future research are given in Section 7.
2 Literature review

Urban delivery system design has close relevance to the general topic of freight network design and facility location (Bektas et al., 2015), which has seen a large body of literature (Daskin, 1995; Drezner, 1995; Drezner and Hamacher, 2002; to list only a few books on this topic). Yet the literature dedicated to urban delivery is sparse. Two forms of urban delivery that are mostly studied are single- and two-echelon systems. The single-echelon system, suitable for small-size urban areas, considers that all city distribution centers (CDCs) are at the same level (Crainic et al., 2009; Guyon et al., 2012; Gianessi et al., 2015). In two-echelon systems, satellite facilities are added between CDCs and customers, and the satellite facilities are where parcel relay occurs. Two separate and dedicated fleets of vehicles move parcels between CDCs and satellites, and between satellites and customers respectively. The two-echelon systems bear some similarity to our proposed system in terms of parcel relay. Because of this, the rest of the literature review focuses on routing problems in two-echelon systems.

According to Cuda et al. (2015), research on two-echelon system routing can be classified into three categories: (i) two-echelon location routing problem (2E-LRP); (ii) two-echelon vehicle routing problem (2E-VRP); and (iii) truck and trailer routing problem (TTRP). 2E-LRP presents the basic form of the three categories. 2E-VRP is a simpler version of 2E-LRP, where the locations of satellite facilities are given. TTRP is a special case of 2E-VRP in that trucks in the second-echelon operation are attached to trailers in the first-echelon operation (in the first echelon, each vehicle is a coupled truck and trailer). In our proposed system, truck-crowdsourcing relay points resemble satellites in 2E-LRP; truck routes are similar to first-echelon routes; and crowdsourcing routes are analogous to second-echelon routes. However, a distinctive feature of our proposed system is that it involves a distributed bidding process. The truck carrier needs to simultaneously select crowdsourcing bids and design truck routes and schedule.

2E-LRP is well recognized as an NP-hard problem (Crainic et al., 2009; Nguyen et al., 2012a), which prevents the problem from being solved to optimality using off-the-shelf solvers. We are only aware of two studies reporting optimal solutions (Crainic et al., 2011; Contardo et al., 2012). The first one introduces three mixed integer linear program formulations for 2E-LRP using one-index, two-index and three-index variables. The two-index and three-index formulations for up to 25 customers and 10 satellites are solved with a commercially available solver. The second study proposes a branch-and-cut algorithm to solve a small-size problem to optimality. All other studies focus on developing heuristic approaches.

Jacobsen and Madsen (1980) and Madsen (1983) are among the first researchers to heuristically solve 2E-LRP. In their research, three heuristics based on the Minimum Spanning Tree method (Jacobsen and Madsen, 1978), the Alternate Location Allocation method in combination with Savings method (Rapp, 1962; Clarke and Wright, 1964), and the Savings method in combination with Drop method (Clark and Wright, 1964; Feldman et al., 1966) are introduced. Boccia et al. (2010) develop an iterative nested approach which is built on the nested approach of Nagy and Salhi (1996) and the two-phase iterative approach of Tuzun and Burke (1999). The problem is decomposed into two components, one for each echelon. A bottom-up approach is developed to combine the two components, i.e., the first echelon solution is built and optimized based upon the second echelon solution. An initial solution for each echelon is constructed using some fast and greedy heuristic and subsequently improved with neighborhood searches including add, drop, and swap moves.

Recently, Nguyen et al. (2012a) present four constructive heuristics and a hybrid metaheuristics GRASP (greedy randomized adaptive search procedure), enforced by a learning process and path relinking to solve 2E-LRP. The constructive heuristics work in two phases: first constructing the second-level routes from satellites to customers, and then the first-level routes from the depot to satellites. The constructed routes are improved using relocation, swap, 2-opt, 3-opt and open/close moves. The path relinking, originally proposed by Glover and Laguna (1993), adds a memory mechanism to improve the performance of the GRASP metaheuristic. The same authors also develop a multi-start iterated local search coupled with Tabu
search and path relinking (Nguyen et al., 2012b). For solving large-size problems, an Adaptive Large Neighborhood Search (ALNS) heuristic is introduced and tested by Contardo et al. (2012).

Review of the literature shows that finding exact solutions for 2E-LRP has so far been successful for only small-size problems. Heuristic approaches have received more attention to obtain quality results for large-size problems within a reasonable amount of computation time. Given the similar NP-hard nature of our problem, in this paper we also seek to develop a heuristic approach to solve the crowdsourcing-enabled parcel relay and delivery system design.

3 Conceptual description of the system

Before delving into the mathematical formulation and solution approach for the system design, it is useful to have a conceptual picture of the system. Recall in Section 1 that the functioning of the system consists of five steps, in which decision-making occurs in steps 2 and 3, by crowdsources and the truck carrier respectively (step 1 is simply information post, step 4 communication of the decisions made, and step 5 execution of the planned delivery). In step 2, each crowdsource determines how to generate and submit bids, with the objective to maximize one’s utility. On receiving the bids, in step 3 the truck carrier simultaneously selects bids and designs truck routes and schedule to minimize its total cost. Below we describe how decisions are made in steps 2 and 3.

For crowdsource decision making, we consider that each bid includes five pieces of information: (1) the customer(s) that the crowdsource plans to serve; (2) the relay point where parcels are transferred between the crowdsource and a truck (we term the combination of the customer(s) and the relay point in a bid a bundle); (3) the crowdsource routing which starts and ends at the crowdsource’s origin and visits all the customers and the relay point in the bundle; (4) the bid price for serving the customer(s); and (5) the crowdsource mode (cycling or walking). The last piece is used to infer the crowdsource’s travel speed.

Each crowdsource can have a combinatorial number of possible bundles. With the aid of personal computing devices (e.g., mobile Apps), we assume that each crowdsource can and will enumerate all feasible bundles. The feasibility of a bundle is constrained by the maximum distance a crowdsource can travel, his/her carrying capacity, and the bidding rule. Subsection 4.1 will provide further detail. Provided that bundles are feasible, a crowdsource considers two aspects when submitting bids. First, the bid price should be competitive against bids from other crowdsources for a given bundle. This requires the route covering the chosen customers and the relay point to be the cost-minimum route for the crowdsource. Second, if feasible bundles are all competitively priced and each crowdsource is allowed to submit a limited number of bids, then the bids submitted by a crowdsource must be those with the highest bid prices, in order to maximize the possible revenue (thus utility) gained. We restrict the number of bids a crowdsource can submit, because otherwise an exponential number of bids would be received by the truck carrier due to the combinatorial nature of bidding (Song and Regan, 2005). This would make the subsequent bid selection computationally difficult.

For decision making by the truck carrier, selecting bids and designing truck routes and schedule are intertwined: selected bids determine not only customers served by crowdsources, but also relay points and customers not covered by the bids and to be visited by truck. The total cost to be minimized is the sum of payment to selected bids, truck operating cost, and time penalty cost for servicing outside customers’ desired time windows. The first component depends on bid selection; the second component on truck routes; whereas the third component is collectively determined by selected bids and truck schedule due to parcel relay.

We now proceed to the mathematical formulations of the decision-making problems facing each crowdsource and the truck carrier.
4 Mathematical formulation

4.1 Crowdsource decision-making: bid generation and submission

As mentioned in Section 3, on seeing the posted pickup and delivery jobs each crowdsource enumerates bundles which are subject to the following feasibility constraints. First is the travel distance limit. We restrict all customers and the relay point in a crowdsource’s bid to be within an $R$-mile radius from the crowdsource’s origin. Second, a bid can only include a maximum number of customers, and there is a limit on the total parcel weight a crowdsource can carry in a bid. Third, as a bidding rule we consider that a bid is for pickup only or delivery only, but not a mix. The rationale for the last constraint is that, at the time of bid generation when truck routes and schedule have not been determined, crowdsources would be facing much more uncertainty and complexity in a bid while coordinating pickup and delivery jobs than if dealing with only pickups or only deliveries. The three constraints add realism to the bid generation process and help reduce the computational burden facing each crowdsource.

Note that the generated bids do not involve the arrival time of the crowdsource at a customer, but only the routes. Again, this is because at the bid generation step crowdsources have no information about the truck arrival time at the relay point in a bid. Consequently, the cost-minimum routes are determined without considering whether pickup/delivery is within the desired time window (in fact, only the truck carrier truly cares about this). If a bid is selected, the associated crowdsource should serve the customers according to the time determined by the truck carrier (in determining the time the truck carrier will respect the travel speed of the crowdsource). In this sense, the system will be mostly attractive to crowdsources who have flexible schedule.

For a bundle that meets the above feasibility constraints, finding the cost-minimum route for the bundle is an Undirected Travel Salesman’s Problem (U-TSP), which is well studied in the vehicle routing literature (Miller et al., 1960; Laporte, 1992). The cost to a crowdsource for servicing a route is calculated as the sum of the time for traversing the route and the time for parcel transfer at the relay point, multiplied by the time value of the crowdsource. For brevity, the explicit U-TSP formulation is not presented here. The only added constraints in our problem is that the relay point must be connected to the origin of the crowdsource, because the crowdsource must first head to the relay point (for delivery), or returns from the relay point (for pickup). Due to the constraints on bundle construction, the sizes of the U-TSP problems will be relatively small. As a result, optimal solutions can be quickly found using standard integer programming solvers. Each time a U-TSP is solved, the consequent routing cost will be the bid price for the bundle.

Once the bid price for each feasible bundle is obtained, following the discussions in Section 3 a crowdsource will submit $B$ highest priced bids to the truck carrier, where $B$ is the maximum number of bids a crowdsource is allowed to submit.

4.2 Truck carrier decision-making: bid selection and truck routes and schedule design

In selecting bids and designing truck routes and schedule, the objective of the truck carrier is minimizing its total cost. The cost minimization problem is formulated as a mixed integer non-linear program, termed MINLP-(BS+TRS) (Mixed Integer Non-Linear Program for Bid Selection and Truck Routing and Scheduling). The formulation entails the following assumptions:

Assumption 1: Each customer has a unique location\(^3\) and corresponds to a single pickup or delivery demand;

Assumption 2: Each pickup or delivery demand must be met via a single visit, either by a truck or by a crowdsource;

\(^3\) The terms location, node and point are used interchangeably throughout the text.
Assumption 3: A relay point may not be visited, visited once, or more than once depending on the bids selected;

Assumption 4: A truck is allowed to wait at a location before serving a customer, if the truck arrives earlier than the start of the customer’s desired time window.

Assumption 5: Crowdsources are not expected to stop en route (from the planning perspective), which can be justified by the fact that in bid generation each crowdsourcer determines bundle bid price assuming no en-route stops. As utility maximizers, crowdsources have no incentive to stop en route, as doing so would increase one’s time in undertaking the delivery job without generating additional payment.

Assumption 6: A truck route always starts from the departing depot and ends at the returning depot. The departing and returning depots can be the same or different. Our model formulation is adaptable to both cases.

The MINLP-(BS+TRS) is specified on an undirected, complete graph \( G = (V, E) \) where node set \( V \) is the combination of: (i) the customer set \( N = N_p \cup N_d \), where \( N_p \) is the set of pickup customers and \( N_d \) the set of delivery customers; (ii) truck’s departing depot \( \{d\} \) and returning depot \( \{r\} \); (iii) the set of relay points \( A \). Set \( E \) corresponds to links connecting the nodes in \( V \).

Assumptions 2 and 3 require the MINLP-(BS+TRS) formulation to distinguish between customers and relay points. A relay point will not be visited by truck if not associated with a selected bid, must be visited once if associated with one selected bid, and may be visited more than once if associated with multiple selected bids. To differentiate possibly multiple visits to a relay point, copies of relay points are created. In this study, we set the number of copies of a relay point (including itself) equal to the number of times the relay point is referred to in submitted bids. Each copy has a unique label. These copies augment the original set of relay points \( A \) to an expanded set \( A' \).\(^4\)

4.2.1 MINLP-(BS+TRS) formulation

We now present the formulation of the MINLP-(BS+TRS). The sets, incidence relation, parameters, and decision variables in MINLP-(BS+TRS) are first listed:

**Sets**

- \( S \): Set of crowdsources
- \( L \): Set of available trucks

**Incidence relation**

- \( \delta_{i}^{bs} \): Incidence relation taking value 1 if customer \( i \) is included in bid \( b \) submitted by crowdsourcer \( s \) and 0 otherwise

---

\(^4\) Replication of relay points could have negative effects, such as the symmetry generated. Symmetry occurs when the variables of a problem can be permuted without changing the structure of the problem (Margot, 2010). This means different permutations of a solution can give the same objective value, and viewed as different solutions. For example, if there are two copies of a relay point (say 1 and 2) then for a given \( s \) and \( b \), keeping everything else constant, a solution with \( y_{bs}^{s1} = 1 \) and \( y_{bs}^{s2} = 0 \) is symmetric to a solution with \( y_{bs}^{s1} = 0 \) and \( y_{bs}^{s2} = 1 \) (see section 4.2.1 for description of variable \( y_{bs}^{s} \)). The symmetry issue becomes more important with the number of replications made for each relay point. A large number of such symmetries can lead to solving many unnecessary sub-problems in the branch-and-bound algorithm and even make relatively easy problems impossible to solve with branch-and-bound (Ostrowski et al., 2010; Margot, 2010). Further details about treating the symmetry issue in integer programming can be found in Ostrowski et al. (2010) and Margot (2010). We thank one of the reviewers for bringing up this issue.
Parameters

- $B$: Maximum number of bids a crowdsource is allowed to submit
- $t_{uv}$: Truck travel time between nodes $u$ and $v$
- $t_{vi}^{bs}$: Time required for crowdsource $s$ to travel from relay point $v$ to customer $i$ included in the crowdsource’s bid $b$
- $\xi_v$: Time for parcel transfer at relay point $v$
- $q_v$: Weight of parcel(s) associated with node $v$; pickup demands are considered positive while delivery demands are considered negative. Node $v$ has a single parcel if it is a customer point and may have multiple parcels if it is a relay point.
- $q_v^D$: Weight of delivery parcel(s) associated with node $v$. $q_v^D$ is positive for locations with delivery demand(s). However, $q_v^D$ will be 0 for nodes with pickup demand(s), which is different from $q_v$
- $K$: Loading capacity of a truck
- $c_{uv}$: Trucking operating cost traveling from node $u$ to node $v$
- $p_{bu}$: Bid price of crowdsource $s$ for serving customer(s) in bid $b$ which also includes relay point $u$
- $e_u, l_u$: Desired earliest and latest service time for customer $u$
- $\pi(T_u, e_u, l_u)$: Penalty for early/late service of customer $u$, which is a function of the truck (or crowdsource) arrival time $T_u$, $e_u$ and $l_u$ (for details see Subsection 4.2.2)

Decision variables

- $T_u$: Service time at node $u$ (by either a truck or a crowdsource)
- $T_d^l$: Departure time of truck $l$ from the departing depot $d$
- $T_r^l$: Arrival time of truck $l$ at the returning depot $r$
- $Q_u$: Truck load after serving node $u$
- $Q_d^l$: Load of Truck $l$ when leaving the departing depot (node) $d$
- $x_{uv}^l$: Binary variable taking value 1 if truck $l$ travels consecutively from node $u$ to $v$ and 0 otherwise
- $y_{bv}^s$: Binary variable taking value 1 if the bid $b$ submitted by crowdsource $s$ with relay point $v$ is selected and 0 otherwise

With these notations, the MINLP-(BS+TRS) is written as the following program (1.1)-(1.23):

$$
\begin{align*}
\text{min} & \sum_{u \in \text{N} \cup \text{A}'} \sum_{d \in \text{D}} \sum_{l \in \text{L}} c_{uv} x_{uv}^l + \sum_{u \in \text{A'}} \sum_{s \in \text{S}} \sum_{b=1}^B p_{bu} y_{bv}^s + \sum_{u \in \text{N}} \pi(T_u, e_u, l_u) \\
\text{s.t. } & \text{Truck routes constraints} \\
\sum_{u \in \text{N} \cup \text{A}'} \sum_{l \in \text{L}} x_{uv}^l \leq 1 & \forall v \in \text{N} \cup \text{A}' \cup \{r\} \\
\sum_{u \in \text{N} \cup \text{A}'} \sum_{l \in \text{L}} x_{dv}^l \leq |\text{L}| & \forall v \in \text{N} \cup \text{A}' \cup \{d\} \\
\sum_{u \in \text{N} \cup \text{A}'} \sum_{l \in \text{L}} x_{uv}^l = \sum_{w \in \text{N} \cup \text{A}'} x_{bw}^l & \forall v \in \text{N} \cup \text{A}', l \in \text{L}
\end{align*}
$$
The objective (1.1) minimizes the sum of truck operating cost (1st term), payment to selected crowdsources (2nd term), and time penalty cost for servicing outside customers’ desired time windows (3rd term). We follow van Duin et al. (2007) and consider the time penalty cost to increase linearly with the deviation of the service time from a customer’s desired time window. The specific functional form of the time penalty cost is presented later in Subsection 4.2.2.

The cost minimization is subject to five groups of constraints. The first group relates to truck routes: every location is visited by a truck at most once (constraints (1.2)-(1.3)). The number of vehicles departing from the departing depot (d) should be no greater than the total number of available trucks |L| (constraint (1.4)). Constraint (1.5) denotes truck flow conservation at each customer location or relay point.

The second group of constraints relates to truck schedule. Constraint (1.6) calculates the time change as a truck travels through a link not involving the departing and returning depots. It is a non-linear constraint and valid only when there exists a direct truck link between u and v. The inequality of this constraint suggests that truck waiting at a location is allowed. Constraint (1.7) calculates the time change as a truck leaves the departing depot for the immediate downstream stop. Note that constraint (1.7) does not include \( \xi_u \) as no parcel transfer time is required at the departing depot. Similarly, constraint (1.8) calculates the time change as a truck leaves the immediate upstream stop for the returning depot. Constraints (1.7)-(1.8) are different from constraint (1.6) in that the departing and returning depots, unlike customer locations or relay points, involve multiple trucks (thus the truck superscript l is necessary for \( T_d^l \) and \( T_r^l \)). Together, constraints (1.6)-(1.8) eliminate the formation of sub-tours (Ropke and Pisinger, 2006). Constraints (1.9)-(1.10) relate the pickup or delivery time of a customer by crowdsourceto the truck service time at the
corresponding relay point. The association is made possible by the mode and routing information provided in the submitted bids (recall that the mode information gives the travel speed of crowdsource). The equality sign reflects that crowdsources do not stop en route. Constraint (1.10) does not involve $\xi_v$ because parcel transfer is made after a parcel-pickup crowdsource arrives at the relay point.

The third group of constraints relates to truck load and capacity, which follows Desrochers et al. (1987). Constraint (1.11) calculates the total delivery demands loaded on truck $l$ at the departing depot. The total weight carried on a truck at the departing depot is the sum of deliveries to be made at downstream nodes. Constraint (1.12) states that the load of a truck at the departing depot should not exceed capacity. Constraint (1.13) calculates truck loads at the immediate downstream to the departing depot. Constraint (1.14) gives truck loads at each node visited by a truck, except the departing and returning depots. Constraint (1.15) states that the truck load should not exceed capacity at any customer or relay points. Note that a capacity constraint at the returning depot is unnecessary because trucks only offload parcels at the returning depot. As long as truck capacity is not exceeded at the immediate upstream node, the capacity constraint will also be met at the returning depot.

The fourth group of constraints relates to bid selection. Constraint (1.16) specifies that every customer must be visited, either by truck or crowdsource. Constraint (1.17) denotes that if a bid is selected, then the associated relay point must be visited by truck. Constraint (1.18) restrains that a crowdsource wins at most one bid. This is to avoid the circumstance that when a crowdsource wins multiple bids, it may require pickup/delivery jobs with overlapping (thus conflicting) times.

The last group of constraints specifies the binary conditions of $x_{uv}^l$’s and $y_{bv}^l$’s and the non-negativity of all decision variables.

With this formulation, the following observation is made.

**Observation 1:** By implementing the crowdsource-enabled system, the truck carrier will be no worse off than pure-truck delivery.

Observation 1 is easy to verify. The optimal solution to the pure-truck delivery problem always presents a feasible solution to MINLP-(BS+TRS) when no bid is selected. This feasible solution provides an upper bound for MINLP-(BS+TRS). Therefore, the minimum cost under MINLP-(BS+TRS) is no greater than the minimum cost with pure-truck delivery, i.e., the truck carrier will not be worse off by implementing the crowdsource-enabled delivery system.

### 4.2.2 Linearization of the objective function and constraints

The non-linear objective and constraints above make it non-trivial to solve MINLP-(BS+TRS) using off-the-shelf solvers even for small-size problems, which is desired to assess the effectiveness of the heuristic approach we later develop to solve large-size problems. One way to overcome this is to linearize the formulation (1.1)-(1.21). For the objective (1.1), the non-linear term is $\sum_{u \in N} \pi(T_u, e_u, l_u)$. Each $\pi(T_u, e_u, l_u)$ increases linearly with the deviation of service time from the desired time window. Assuming a constant penalty rate $P$ ($$/\text{unit time})$, $\pi(T_u, e_u, l_u)$ can be expressed by the following piece-wise linear function ($\forall u \in N$):

$$
\pi(T_u, e_u, l_u) = \begin{cases} 
P(T_u - l_u) & \text{if } T_u > l_u \\
0 & \text{if } e_u < T_u < l_u \\
P(e_u - T_u) & \text{if } T_u < e_u
\end{cases}
$$

(1.22)

We linearize (1.22) by introducing two additional decision variables, $e_u$ and $\tau_u$, replacing $\pi(T_u, e_u, l_u)$ in (1.1) by (1.23), and adding three new constraints (1.24)-(1.26):

$$
\sum_{u \in N} P(e_u + \tau_u) 
$$

(1.23)

$$
e_u \geq e_u - T_u \ \forall u \in N
$$

(1.24)
\[ \tau_u \geq T_u - l_u \quad \forall u \in N \quad (1.25) \]

\[ \varepsilon_u, \tau_u \geq 0 \quad \forall u \in N \quad (1.26) \]

where \( \varepsilon_u \) is the amount of time customer \( u \) is served before \( e_u \), and \( \tau_u \) is the amount of time customer \( u \) is served after \( l_u \). Proof of the equivalence between \( \pi(T_u, e_u, l_u) \) and (1.23)-(1.26) is provided in Appendix A.

Non-linear constraints (1.6)-(1.8) and (1.13)-(1.14), which have inequality signs, can be linearized using big-M following Desrochers et al. (1987). Non-linear constraints (1.9)-(1.10), which have equality signs, will each require an additional constraint to be linearized:

\[ (T_v + \xi_v + t_{vi}^{bs} - T_i) \leq (1 - \delta_i^b y_{bv}^s)M \quad \forall s \in S, b \in B_D, v \in A', i \in N_D \quad (1.9a) \]

\[ (T_v + \xi_v + t_{vi}^{bs} - T_i) \geq (\delta_i^b y_{bv}^s - 1)M \quad \forall s \in S, b \in B_D, v \in A', i \in N_D \quad (1.9b) \]

\[ (T_v - t_{vi}^{bs} - T_i) \leq (1 - \delta_i^b y_{bv}^s)M \quad \forall s \in S, b \in B_P, v \in A', i \in N_P \quad (1.10a) \]

\[ (T_v - t_{vi}^{bs} - T_i) \geq (\delta_i^b y_{bv}^s - 1)M \quad \forall s \in S, b \in B_P, v \in A', i \in N_P \quad (1.10b) \]

The above transformations result in a linearized version, MILP-(BS+TRS) (Mixed Integer Linear Program for Bid Selection and Truck Routing and Scheduling), of the original formulation.

Since MILP-(BS+TRS) contains VRPTW as a special case when no crowdsources are present, MILP-(BS+TRS) is an NP-complete problem like VRPTW. Given the NP-completeness, no polynomial-time running algorithms exist for solving MILP-(BS+TRS) unless \( P = NP \). Off-the-shelf solvers such as CPLEX are able to solve only small-size problems. In the next section, we propose a heuristic approach to solve problems of larger sizes.

5 Solution approach

The basic idea for solving MILP-(BS+TRS) is to decompose it into two sub-problems: i) selecting bids from the submissions; ii) routing and scheduling of trucks. The first sub-problem pertains to solving a Winner Determination Problem (WDP); the second one a Simultaneous Pickup and Delivery Problem with Soft Time Windows (SPDPSTW). We iteratively solve the two subproblems using a Tabu Search based algorithm to improve the overall solution to MILP-(BS+TRS). Below is the outline of the algorithm.
Algorithm 1: Solving MILP-(BS+TRS)

Step 0: Set up an empty Tabu list of relay points.

Step 1: Solve WDP to select crowdsources to serve customers. The solution indicates the relay points to be used by the selected crowdsources. Based on the solution, calculate the payment to each selected crowdsource.

Step 2: Solve SPDPSTW in which trucks visit the selected relay points as well as customers not covered by the selected crowdsources. Add the resulting truck operating cost and time penalty cost to the payment to crowdsources in Step 1. Label it as the current best total cost.

Step 3: Among the selected relay points that are not Tabu, select the one with the minimum associated cost and temporarily remove it. Re-solve WDP and SPDPSTW. Two cases may occur:

3.1: If the total cost is higher than the current best cost, label the temporarily removed relay point as Tabu and add it back to the set of selected relay points. Go back to the beginning of Step 3;

3.2: If the total cost is lower than the current best cost, permanently remove the relay point; update the current best cost and the set of selected relay points. Go back to the beginning of Step 3;

Stop when all selected relay points are labeled as Tabu, or when the current best solution does not involve relay points (i.e., all customers directly served by truck). Store the Tabu list. Go back to Step 0 to start the next iteration.

Step 4: Terminate when the stored Tabu lists in two consecutive iterations are identical.

While most parts of the algorithm are self-explanatory, two points are worth highlighting. First, after we temporarily remove the relay point with the minimum associated cost in step 3, the total cost resulting from WDP will increase or remain the same because the number of available bids is now fewer (the bids associated with the removed relay point will become infeasible). However, conditional on the bids and relay points selected the removal may reduce the SPDPSTW cost. Thus it is still possible that the total cost is reduced. Second, each time we start a new iteration in step 0, fewer relay points will remain as some are permanently removed in the previous iteration in step 3.2 (If the number of relay points does not change over two iterations, then the algorithm will terminate (step 4)).

Below we describe in detail how the two sub-problems, WDP and SPDPSTW, are solved.

5.1 Solving WDP

We formulate WDP as a binary integer program and solve it to optimality using the branch-and-bound algorithm. The problem is slightly modified compared to the classic formulation (for example, Blumrosen and Nisan (2007)), to allow trucks to serve customers not covered by selected bids. The modified WDP is formulated as below:

\[
\min \sum_{u \in A} \sum_{s \in S} \sum_{b \in B} P_{bu}^s y_{bu}^s + M \sum_{i \in N} z_i
\]

\[
z_i + \sum_{u \in A} \sum_{s \in S} \sum_{b \in B} \delta_i^b y_{bu}^s = 1 \quad \forall i \in N
\]

\[
\sum_{v \in A} \sum_{b \in B} y_{bv}^s \leq 1 \quad \forall s \in S
\]
\[ y_{iv}^s \in \{0,1\} \quad \forall v \in V_i, s \in S, v \in A \tag{2.4} \]
\[ z_i \in \{0,1\} \quad \forall i \in N \tag{2.5} \]

In the objective (2.1), the first summation is payment to selected crowdsources. In addition, a big-\(M\) term is added to penalize not having crowdsources serve customers. The underlying assumption is that the truck carrier wants crowdsources to cover as many customers as possible, because crowdsourcing is expected to be cheaper than using trucks. \(z_i\) is a binary variable taking value 1 if customer \(i\) is not served by selected crowdsources and 0 otherwise. Constraint (2.2) states that all customers must be served exactly once, either by crowdsource or by truck (in the latter case, \(z_i = 1\)). This constraint implicitly assumes that the bidding crowdsources are interested only in obtaining an entire bundle, not any subset of it. Constraint (2.3) stipulates that each crowdsource wins at most one bid. Constraints (2.4)-(2.5) state that the decision variables are binary. Note that consistent with our discussions in Subsection 4.1, the WDP does not consider time windows.

Solution to the WDP yields selected bids and relay points. In addition, the truck carrier will know the sequence of customer visits in each winning bid. Then the truck carrier needs to solve SPDPSTW to determine truck routes and schedule, so that all relay points in the selected bids as well as customers uncovered by the selected bids are visited by truck.

5.2 Solving SPDPSTW

SPDPSTW is a variant of vehicle routing problems with time windows (VRPTW), with the difference that customers can have both pickup and delivery requests. In addition, in SPDPSTW all deliveries originate from the departing depot and all pickups are destined to the returning depot. Thus SPDPSTW is different from traditional Pickup and Delivery Problems (PDP) where the origin of pickups and destination of deliveries can be customers. Soft time windows stipulate that pickups/deliveries are acceptable outside customers’ desired time windows, albeit with penalty. We consider that if a truck arrives at a customer early, it should wait till the beginning of the customer’s desired time window to serve the customer. Similar treatment is made when a truck visits a relay point. Since a relay point itself does not have a desired time window, we construct the desired time window of a relay point based on the desired time window of the first or last customer visited in the associated bid, depending on whether it is a delivery or pickup bid. If the associated bid is for delivery, \(e_u\) for the relay point is set to \(e_u - t_{u,u'}\), where \(u'\) is the first customer to visit in the bid and \(t_{u,u'}\) is the crowdsource travel time from \(u\) to \(u'\). \(l_u\) is \(e_u\) plus a pre-specified time window length. If the associated bid is for pickup, \(l_u\) for the relay point is set to \(l_u + t_{u,u'}\), where \(u'\) is the last customer to visit in the bid and \(t_{u,u'}\) is the crowdsource travel time from \(u'\) to \(u\). \(e_u\) is \(l_u\) minus a pre-specified time window length.

Our SPDPSTW bears some similarity with the simultaneous pickup and delivery problems with hard time windows, which has been investigated by several researchers. Angelelli and Mansini (2003) consider different branch-and-price strategies to solve a problem of this type to optimality. As this type of problems is also NP-hard, several heuristics have been developed, including an Improved Differential Evolution (IDE) algorithm (Mingyong and Erbao, 2010), a multi-ant colonies algorithm (Boubahri et al., 2011), a coevolution genetic algorithm (Wang and Chen, 2013), and a parallel Simulated Annealing method (Wang et al., 2015). However, we are not aware of previous efforts considering soft time windows for simultaneous pickup and delivery, although VRP with soft time windows has been studied in Balakrishnan (1993), Qureshi et al. (2009), and Figliozzi (2010). Compared to delivery with hard time windows, SPDPSTW has the advantage of reducing truck fleet size and improving vehicle capacity utilization, both of which reduce truck carrier cost.

Our algorithm for solving SPDPSTW uses the Simulated Annealing principle to improve or fine tune routes through various node moves. Simulated Annealing belongs to the class of “probabilistic hill climbing” algorithm to approximate the global optima for complex combinatorial optimization problems (Daganzo, 2005). It mimics the cooling of material in a heat bath. The theoretical background to Simulated
Annealing is derived from the concepts in statistical mechanism and Markov Chains (Dowsland and Thompson, 2012). With sound generate, accept, and update functions, Simulated Annealing is shown to converge the solution in probability to a minimum cost configuration (Mitra et al., 1986; Laarhoven and Aarts, 1987; Daganzo, 2005). It is simple, flexible, well suited for solving VRPs, and close to being an ideal general purpose tool for fine tuning (Robuste et al., 1990). It has also been reported that Simulated Annealing produces reasonably good solutions for large routing problems faster than other heuristics such as Genetic Algorithm (Tan et al., 2001; Adewole et al., 2012; Antosiewicz et al., 2013).

Below we present first the components (i.e., initial solution generation, route improvement, and acceptance criteria) and then the overall flow of the algorithm.

5.2.1 Initial solution construction

To generate the initial truck routes, we consider scheduling trucks to first fulfill delivery demands and then pickup demands. Trucks need to visit all relay points in the selected bids and customers uncovered by the selected bids. A truck route is progressively constructed using the nearest neighbor heuristic, i.e., from the current end node of the route to an unvisited node that has the lowest generalized link cost. The generalized cost on a truck link \((v, u)\), \(GC_{vu}\), is the sum of truck operating cost \(C_{vu}\) and time penalty cost for visiting the node \(\pi(T_u, e_u, l_u)\):

\[
GC_{vu} = C_{vu} + \pi(T_u, e_u, l_u) \tag{3.1}
\]

Recall that our previous definition of \(\pi(T_u, e_u, l_u)\) is for customer nodes (equation (1.22)). If \(u\) is a relay point in \(A'\), we further define \(\pi(T_u, e_u, l_u)\) as the sum of time penalty cost for all customers in the associated bid.

The nearest neighbor heuristic proceeds as follows. We start from the first truck. Among the relay points in the selected bids and the customers uncovered by the selected bids, the point having the lowest generalized cost from the departing depot is added to the truck’s route. We then connect this added point to the next point among the remaining relay points in the selected bids and the customers uncovered by the selected bids, such that the generalized cost from the added point to the next point is the lowest. We repeat this until no more delivery load can be added due to truck capacity constraint. Then we start constructing the second truck’s route, again from the depot. This process ends when all delivery demands are assigned to truck routes.

After completing delivery, the above trucks are reused for pickups. Pickup truck routes are formed in a similar fashion as delivery truck routes. However, the pickup route of a truck starts from the last delivery node. If additional trucks are needed due to capacity restrictions, these additional trucks will start from the depot.

5.2.2 Route improvement

An iterative process is adopted to improve the initial truck routes. In each iteration, three neighborhood search based node moves are performed: 2-opt move, node interchange and node relocation (Figure 2). The first move is an intra-route improvement, whereas the second and third are inter-route improvements.

In a 2-opt move, two non-adjacent links within a truck route are reshuffled (Figure 2(a)). A reshuffle changes the node visiting orders, and consequently the truck capacity constraint may be violated. If the move is feasible, i.e., truck capacity is not exceeded, and the total cost for the truck is reduced, then the move is definitely accepted. If the move is feasible but the total cost for the truck is increased, then additional acceptance criteria (Subsection 5.2.3) are applied to determine whether to accept the move. The 2-opt move is performed for every truck route and every possible non-adjacent link pair in a truck route.

In a node interchange/relocation move, two nodes which belong to two truck routes are exchanged/moved from one route to another (Figure 2 (b)/(c)). If the move is the feasible, i.e., truck capacity is not exceeded on both truck routes, and the total cost for both trucks is reduced, then the move is definitely
accepted. If the move is feasible but the total cost for both trucks is increased, then the same additional acceptance criteria mentioned above are applied to determine whether to accept the move. The node interchange/relocation move is performed for all possible truck route pairs, and given a route pair, all possible node pairs.

The iterative process stops when no reduction in total cost can be found for \( n \) consecutive steps. We also employ an upper bound \( N \) as the maximum number iterations to be performed.

![Route improvement moves](image)

**Figure 2:** Route improvement moves: (a) 2-opt move: two links connecting the black nodes are reshuffled; (b) Node interchange: black nodes are interchanged between two routes; (c) Node relocation: black node is relocated from one route to another

### 5.2.3 Acceptance criteria

The acceptance criteria for a move when truck cost increases reflects the Simulated Annealing principle that accepting some non-improving solutions makes the search more extensive to avoid being trapped in local minimum. Specifically, the acceptance criteria are based on a probabilistic approach. When a move leads to higher truck cost, the move would still be accepted with a probability of \( e^{\frac{-\Delta \text{cost}}{T}} \), where \( \Delta \text{cost} = \text{cost before the move} - \text{cost after the move} \), which is negative (note that for a positive \( \Delta \text{cost} \), the probability of acceptance \( e^{\frac{-\Delta \text{cost}}{T}} \) will be greater than 1, in which case the solution will always be accepted). \( T \) is the current temperature which is greater than 0. The temperature decreases at a given cooling rate \( c < 1 \) at every iteration. At the beginning of the iterative process, the temperature is high, meaning that the probability of accepting a non-improving solution is high for a given negative \( \Delta \text{cost} \). As the iterative process proceeds, both temperature and the probability of accepting a cost-increasing move decrease.

It has been found that the quality of Simulated Annealing solutions depends on the initial temperature and cooling rate \( c \) (Ropke and Pisinger, 2006). In this study, the initial temperature is set such that total cost that is \( w\% \) greater than the initial total cost is accepted with 0.5 probability, where \( w \) is a pre-specified parameter (Ropke and Pisinger, 2006). For \( c \), existing research does not suggest any definite values. Section 6 will provide further details on how we choose \( w \) and \( c \).

### 5.2.4 Overall algorithm to solving SPDSPTW

Based on the discussions in Subsections 5.2.1-5.2.3, Algorithm 2 below gives the pseudo code to solve SPDSPTW. Four functions: initial_solution, 2-opt_move, node_interchange, and node_relocation are involved. The variables appearing in the parentheses after each function name specify outputs of the function.

In Algorithm 2, lines 1-3 generate initial truck routes. The initial routes and costs are labeled as the best found at iteration 0. Line 4 sets up the initial temperature (such that the solution that is \( w\% \) worse than the initial solution is accepted with 0.5 probability). Line 5 stipulates that the algorithm would proceed up to \( N \) iterations (note that it can terminate earlier, as specified in line 36). At the start of each iteration, the temperature is updated (lowered) as in line 6.

A 2-opt move is performed in line 7. Line 8 calculates the acceptance probability based on the cost at the current iteration and the best cost found so far. If the acceptance probability is greater than a randomly
generated number in \([0,1]\), then the best routes and cost are updated (lines 9-11). Otherwise, the best routes and cost remain unchanged (lines 12-15).

After completing the 2-opt move, node interchange is performed (line 16). The acceptance criteria are similar to that of 2-opt move (lines 17-24). After the completion of node interchange, node relocation is applied in line 25, again with similar acceptance criteria (lines 26-33). Lines 34-37 state when the iteration will stop. When more than \(n\) iterations are performed, the best costs of the \(n\) latest iterations are compared. If these costs are identical, the algorithm exits the for-loop (line 36) (the value of \(n\) is typically greater than 2, as \(n = 2\) often leads to pre-mature termination of the algorithm due to rounding errors). Line 39 outputs the best routes and cost.

Algorithm 2: Solving SPDPSTW

1. **Function** initial_solution (route, cost)
2.  
3.  
4.  
5.  
6.  
7.  
8.  
9.  
10.  
11.  
12.  
13.  
14.  
15.  
16. **Function** 2-opt_move(route, cost)
17.  
18.  
19.  
20.  
21.  
22.  
23.  
24.  
25. **Function** node_interchange(route, cost)
26.  
27.  
28.  
29.  
30.  
31.  
32.  
33.  
34.  
35.  

Note that if the current cost is less than the previous best cost, the acceptance probability will be greater than one, and the best routes and cost will surely get updated.

5
6 Numerical experiments

This section implements the proposed design and solves both small- and large-size problems. For the small size problems, exact optimality can be reached using off-the-shelf solvers. These solutions are used to evaluate the quality of solutions generated from Algorithm 1. For large-size problems, we can only obtain results from Algorithm 1. Monte Carlo simulations are performed for the large-size problems to gain robust insights about savings in total cost and truck VMT with the proposed design as compared to pure-truck delivery. We also investigate the sensitivity of the results to different model parameters.

The proposed design is coded and different problems are solved using Matlab 2012b on an Intel Core i7 3630 2.4 GHz machine with 8 GB RAM. For obtaining exact optimal solutions, ILOG CPLEX v12.6 is used in the Matlab environment.

6.1 Value for model parameters

Values for model parameters are obtained from existing studies wherever possible. The average truck speed is assumed 20 mph (Lee et al., 2013). Similar truck speed values are also reported in Barnitt (2011) and EPA (2009). We consider a unit truck operating cost of $68.09/hour following ATRI (2015). The unit cost includes the cost of leasing or purchasing a truck, compensation and benefits to the driver, fuel cost, tolls, tires, permits, and insurances. The travel speed of a crowdsourcer is assumed 10 mph for cycling (Jensen et al., 2010) and 2.5 mph for walking. We consider the time value of a crowdsourcer to be $10/hr, which is comparable to the US minimum wage rate (DOL, 2016). Each customer has either a parcel to be picked up or delivered. The weight of a parcel is drawn from an exponential distribution with a mean value of 10 lbs. For service outside the desired time window, a penalty rate of $2/hr is applied. The choice of the penalty rate is inferred from the shipping fees currently charged in practice (e.g., $2.99 per delivery by Postmates (2016) and $5.99 for the same-day delivery by Amazon (2016)). Our assumption is that lateness of delivery would result in partial or full refund of the shipping fees. In Subsection 6.3.2, alternative rates from $3/hour to $10/hour are tested to further understand the impact of the penalty rate on the crowdsourced shipping system performance.

In the crowdsourcer bidding, we consider that a cyclist (pedestrian) only considers customers and relay points within a 2-mile (0.75-mile) radius from his/her origin. A bundle can include at most 5 (3) customers in a cyclist (pedestrian) bid due to carrying capacity limit. In addition, we limit the maximum weight a cyclist (pedestrian) can carry to be 20 (10) lbs. The time required for parcel transfer at a relay point is assumed 5 minutes. In a small-size problem, each crowdsourcer is allowed to submit 2 bids; in a large-size problem, the maximum number of bids allowed for a crowdsourcer is 5. The values for $\omega$, $c$, $n$, and $N$ in Algorithm 2 are $(\omega, c, n, N) = (10, 0.96, 50, 1000)$. This parameter choice is the result of jointly considering solution quality and computation time, as further shown in Appendix B.

6.2 Small-size problems

Eight small-size problems are tested, with the number of customers ranging from 10 to 30 and the number of relay points and crowdsourses ranging from 2 to 8 (Table 1). These problems are solved using both CPLEX (to optimality) and Algorithm 1. For each problem, the depot is set at (0, 0) of a plane. Customers, relay points, and crowdsourcer origins are randomly generated within a 2-mile radius of (0, 0). Crowdsources are all cyclists. The beginning of the desired service time window for each customer is randomly drawn from integer numbers 1-12, with each number corresponding to a non-overlapping 15-min interval. Each time window is assumed to be 30 min long. The number of trucks when solving a problem...
using CPLEX is equal to the number of trucks obtained from Algorithm 1. We consider two capacity values for a truck: 0.25 times and 2 times the total weight of customer demands. These two values are chosen to test multi- and single-truck use scenarios.

Table 1 shows the total costs and CPU time using CPLEX and Algorithm 1 for the eight problems. To capture the randomness in accepting a non-improving solution, each problem is solved 5 times using Algorithm 1. The average values are reported in Table 1. We also present the optimality gap, defined as the difference between the solutions from Algorithm 1 and CPLEX, divided by the CPLEX solution. For each problem, the coefficient of variation (COV) for the heuristic solutions is also presented. It is found that Algorithm 1 is very robust (small COVs) and produces quality solutions with small optimality gaps. The optimality gap increases somewhat with problem size, and is smaller with two trucks than one truck. The latter fact is not surprising, since having multiple trucks permits all three types of node moves to improve truck routes (Subsection 5.2.2), whereas having one truck only allows for 2-opt moves. Using CPLEX, the computation time explodes quickly with problem size – we are not able to solve the problems with 30 customers to optimality in 4 hours. In contrast, Algorithm 1 solves the problems in 2 min and yields comparable results to those using CPLEX. The results suggest the suitability of the Tabu Search and Simulated Annealing based algorithm for the small-size problems. In what follows, we further apply the developed heuristics solution algorithms to large-size problems.

### Table 1: Comparison of the results

<table>
<thead>
<tr>
<th>Problem</th>
<th># of Customers</th>
<th># of relay points</th>
<th># of total crowd-sources</th>
<th># of trucks used (ratio of truck capacity and total customer demand)</th>
<th>CPLEX</th>
<th>Algorithm 1</th>
<th>% Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total Cost</td>
<td>CPU time (sec)</td>
<td>Total Cost (COV)</td>
<td>CPU time (sec)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>2</td>
<td>2</td>
<td>1 (2)</td>
<td>44.08</td>
<td>3.67</td>
<td>44.99 (1.32%)</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>4</td>
<td>4</td>
<td>2 (0.25)</td>
<td>43.16</td>
<td>15.2</td>
<td>43.87 (0.89%)</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>2</td>
<td>2</td>
<td>1 (2)</td>
<td>42.24</td>
<td>4.98</td>
<td>44.77 (1.12%)</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td>4</td>
<td>4</td>
<td>2 (0.25)</td>
<td>48.33</td>
<td>19.5</td>
<td>50.00 (1.31%)</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>2</td>
<td>2</td>
<td>1 (2)</td>
<td>58.46</td>
<td>29.1</td>
<td>64.04 (2.01%)</td>
</tr>
<tr>
<td>6</td>
<td>20</td>
<td>4</td>
<td>4</td>
<td>2 (0.25)</td>
<td>54.51</td>
<td>112</td>
<td>59.08 (1.87%)</td>
</tr>
<tr>
<td>7</td>
<td>30</td>
<td>4</td>
<td>4</td>
<td>1 (2)</td>
<td>93.96</td>
<td>4 hr</td>
<td>102.93 (3.03%)</td>
</tr>
<tr>
<td>8</td>
<td>30</td>
<td>8</td>
<td>8</td>
<td>2 (0.25)</td>
<td>90.99</td>
<td>4 hr</td>
<td>98.34 (3.01%)</td>
</tr>
</tbody>
</table>

*Values obtained at the end of the 4-hr running time.

### 6.3 Large-size problems

#### 6.3.1 Network setup

The large-size problems consider 25 customers scattered in each of the four quadrants of a plane, thus in total 100 customers (Figure 3). Each quadrant has a square shape of 10 miles × 10 miles. In each quadrant, customers are randomly scattered around the centroid of the quadrant with a 3.5-mile radius. Each quadrant has 5 relay points and 15 crowd-sources, which are also randomly scattered around the centroid with the same radius. Crowd-sources are all cyclists. The depot again lies at the center of the plane.

Customers’ desired time windows are more spread out than in the small-size problems. Specifically, the beginning of the desired service time window for a customer is randomly generated from 1 to 32, again with each number corresponding to a 15-min interval. Each time window is assumed to be 2 hours long. The capacity of a truck is set to one fourth of the total weight of customer demands, which ensures that at least two trucks are needed to fulfill the pickups and deliveries.
Because customers, relay points, and crowdsources are randomly scattered, we generate and solve 100 problem instances using Algorithm 1. For comparison, we also solve each problem instance with only trucks, using Algorithm 2.

Figure 3: An example of customer, relay point, and crowdsource distributions on a four-quadrant plane

6.3.2 Results and sensitivity analysis

Figure 4 presents the results of total cost, total truck VMT, and total periods (i.e., 15-min intervals) of service time violations from the 100 instances using boxplot. In each graph, the first boxplot corresponds to pure-truck delivery and the second one the crowdsource-enabled delivery. At the median, total cost saving is around 9.25% with crowdsourcing. Median truck VMT will be reduced even more significantly, by about 24%. These are in spite of some slight increase in the total periods of service time violations (3%). Results from paired t-tests further show that the null hypotheses on the equal means of total cost and total truck VMT are rejected at 0.05 level of significance, whereas we cannot reject the null hypothesis on the equal means of total periods of service time violation (Table 2). Averaged over the 100 instances, the

---

6 From Observation 1, the truck carrier will not be worse off using crowdsources. On the other hand, because here heuristic solution approaches are used, it cannot be guaranteed that the total cost of crowdsource-enabled delivery using Algorithm 1 is always no greater than the total cost of pure-truck delivery using Algorithm 2. Among the 100 instances tested, four yield greater total cost with crowdsource-enabled delivery. Nevertheless, for these four instances the cost difference between crowdsource-enabled and pure-truck deliveries is very small, lesser than 2%. This also applies to explaining the occurrence of cost negative cost savings in Figures 6-8.
number of customers served by crowdsources is 52 (out of 100) and the median total payment from the truck carrier to the crowdsources is $128.

Figure 4: Pure-truck vs. crowdsource-enabled delivery in terms of (a) total cost (b) total truck VMT (c) total periods of service time violations

Table 2: Hypothesis test for the equal mean of total cost, truck VMT and periods of service time violation between crowdsource-enabled and pure-truck delivery

<table>
<thead>
<tr>
<th>Null hypothesis (H₀)</th>
<th>T-statistics</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equal mean of total cost</td>
<td>12.93</td>
<td>Reject H₀</td>
</tr>
<tr>
<td>Equal mean of total truck VMTs</td>
<td>-24.21</td>
<td>Reject H₀</td>
</tr>
<tr>
<td>Equal mean of total periods of service time violations</td>
<td>0.089</td>
<td>Fail to reject H₀</td>
</tr>
</tbody>
</table>

Looking into the service time violations, we find that the total periods amount to 370, or 92 hrs. In other words, an average customer is served 55 min (92/100 hours) outside one’s desired time window. This translates into a total of about $184 to the truck carrier (recall that we use a penalty rate of $2/hr). The total time periods violated will, intuitively, decrease as the penalty rate increases. Figure 5 shows the decrease in number of time periods violated as the penalty rate increases up to $10 per hour. However, with the penalty rate increase the cost advantage over pure-truck delivery will become less. As shown in Figure 6, the median percentage cost savings using crowdsourcing drop from 9.25% with a penalty rate of $2/hr to about 6% at the rate of $3/hr. At the penalty rate of $10/hr, the saving drops to only about 1.5% making it not very lucrative to use crowdsources. The higher penalty rates represent the scenario where the carrier cares more about its goodwill and is willing to maintain a greater on-time service performance.
Besides penalty rate, a few other model parameters may also affect the attractiveness of crowdsource-enabled delivery. In the remainder of this Subsection, we investigate the impact of truck unit operating cost, the time value of crowdsources, and crowdsource mode on system performance. In the previous experiments, the truck unit operating cost is adopted from ATRI (2015). However, uncertainties about future fuel prices and labor supply in the trucking industry lead us to consider variations of the truck unit operating cost. We simulate five scenarios with the truck unit operating cost ranging from 20% lower to 20% higher than the ATRI (2015) value. Because trucking is more expensive than cycling/walking, we expect greater cost advantage of crowdsource-enabled delivery with higher truck unit operating cost. This is confirmed in Figure 7: the median percentage cost savings with crowdsource-enabled delivery increases to almost 12.5% when truck unit operating cost is 20% higher; in contrast, when truck unit operating cost is reduced by 20%, the median percentage cost savings will be only 5%.
Figure 7: Percentage of cost savings using crowdsource-enabled vs. pure-truck delivery with different truck unit operating costs as compared to the value in ATRI (2015)

The time value of crowdsources also affects the cost competitiveness of crowdsource-enabled delivery. Previously, a time value of $10/hr is assumed. As the minimum wage rate rises in the future, so will the time value of crowdsources. Holding truck unit operating cost constant, an increase in crowdsources’ time value will compromise the competitiveness of crowdsource-enabled delivery. As shown in Figure 8, the median percentage cost savings would decrease from 9.25% to 6.5%, 5% and 2% if the time value of crowdsources is increased by 25%, 50%, and 75% respectively.

Figure 8: Percentage of cost savings using crowdsource-enabled vs. pure-truck delivery as time value of crowdsources increases

So far we have considered cycling as the crowdsource mode. If crowdsources are instead pedestrians, who are slower and have less carrying capacity, then crowdsource-based delivery will have reduced competitiveness. Our numerical experiments for the same 100 instances show that, compared to pure-truck delivery, the median truck VMT is reduced by 7% (compared to 24% with cycling). Fewer crowdsources will be used, with median total payment from the truck carrier to the selected crowdsources decreased from $128 to $29. The median percentage cost savings is only 2%.
7 Conclusion

In this paper, we propose a crowdsourcing-enabled system design for urban parcel delivery. In this design, a truck carrier posts pickup and delivery jobs and relay points online. Local crowdsources such as cyclists and pedestrians respond by generating and submitting bids. The truck carrier selects the bids and determines truck routes and schedule to coordinate with the crowdsourced jobs. By replacing trucking with local crowdsources for the last leg, total cost to the truck carrier, which consists of truck operating cost, payment to crowdsources, and time penalty for service outside customers’ desired time windows, can be reduced compared to pure-truck delivery. The new design also reduces truck VMTs, making it an attractive alternative not only from the economic perspective but also for developing urban livable communities.

Designing the crowdsourcing-enabled system involves solving Undirected Travel Salesman’s Problems for crowdsourcing bid generation, and solving a mixed integer non-linear program for the truck carrier to select bids and determine truck routes and schedule. For the truck carrier problem, we proposed a Tabu Search based algorithm which iteratively solves a Winner Determination Problem (WDP) and a Simultaneous Pickup and Delivery Problem with Soft Time Windows (SPDPSTW). In the algorithm, WDP is solved exactly using the branch-and-bound method; a sub-algorithm using the Simulated Annealing principle is developed to solve SPDPSTW. Overall, the Tabu Search based algorithm is able to yield solutions that are close to optimum in our small-size problems.

We implement the design in large-size instances, and find total cost and truck VMTs can be significantly reduced compared to pure-truck delivery with cycling as crowdsourc mode. Over half of the customers will be served by crowdsources. The attractiveness of the crowdsourcing-enabled system depends on factors such as penalty rate for serving customers outside the desired time windows, truck unit operating cost, time value of crowdsources, and crowdsourcing mode. Overall, lower penalty rate, greater truck unit operating cost, lower time value of crowdsources, and using cyclists instead of pedestrians will enhance the cost competitiveness of the crowdsourcing-enabled system.

This paper presents a promising beginning of using crowdsourcing to innovate existing truck-based urban delivery practices. Future research can be extended in a few directions. First, in this paper we assume that crowdsources undertake the assigned delivery jobs irrespective of the time of a day. This may be justified on the ground that crowdsources, whose time value is low, typically have flexible schedules. On the other hand, it may be possible that crowdsources have limited time availabilities. In this case, either crowdsources will submit their time availability information as part of the bids, or the bidding process may take multiple rounds to meet the constraints on crowdsourcing time availability. Second, although the present paper investigates operational planning, the crowdsourcing concept can apply to real-time or close-to-real-time planning for delivery as well. Then demand uncertainty and consequent route adjustment while trucks are already en route should be considered. Third, it would be interesting to compare the results using the Simulated Annealing based routing heuristics with results based on Adaptive Large Neighborhood Search Algorithms, which are also shown to be effective for solving vehicle routing problems. Fourth, public policy issues such as the location of relay points in neighborhoods and safety and privacy concerns over crowdsourced delivery should be investigated and addressed so that the crowdsourcing-enabled system is receptive by local communities and customers.

Acknowledgement

Earlier versions of this paper were presented at the Chicago Area Travel Model User Group (CATMUG) seminar organized by the Chicago Metropolitan Agency for Planning, and in the “Modeling and Analysis of Innovative Mobility Services” session of the 2016 INFORMS annual meeting. We are grateful to the anonymous reviewers for their constructive comments which helped us improve the presentation of the paper.
Appendix A: Proof of equivalence between $\pi(T_u, e_u, l_u)$ and (1.23)-(1.26).

We present three possible cases of $T_u$ with respect to $e_u$ and $l_u$.

Case a: $T_u > l_u$. Constraint (1.24) states that $\varepsilon_u \geq e_u - T_u$, which is negative because $e_u < l_u < T_u$. Combining with (1.26), the overall constraint for $\varepsilon_u$ is $\varepsilon_u \geq 0$. By the same token, the overall constraint for $\tau_u$ is $\tau_u \geq T_u - l_u$. To minimize the time penalty for vehicle $u$ (which is now expressed as $P(\varepsilon_u + \tau_u)$) given $T_u, e_u, l_u$, it must be that $\varepsilon_u = \tau_u = 0$. This leads to the same time penalty cost as the first line in (1.22).

Case b: $e_u < T_u < l_u$. The right-hand-sides of both (1.24) and (1.25) are negative. Combining the non-negative constraint (1.26), the overall constraints for $\varepsilon_u$ and $\tau_u$ will be $\varepsilon_u \geq 0$ and $\tau_u \geq 0$. To minimize time penalty for vehicle $u$ given $T_u, e_u, l_u$, it must be that $\varepsilon_u = \tau_u = 0$. This leads to the same time penalty cost as the second line in (1.22).

Case c: $T_u < e_u$. Discussions here will be similar to Case a. The overall constraints for $\varepsilon_u$ and $\tau_u$ will be $\varepsilon_u \geq e_u - T_u$ and $\tau_u \geq 0$. To minimize the time penalty for vehicle $u$ given $T_u, e_u, l_u$, it must be that $\varepsilon_u = e_u - T_u$ and $\tau_u = 0$. This leads to the same time penalty cost as the third line in (1.22).

Summing up the time penalties across customers gives total time penalty cost, as expressed (1.23). This completes the proof. □

Appendix B: Choice of $(w, c, n, N)$ values

The values of $(w, c, n, N)$ should be set such that Algorithm 2 terminates in a reasonable amount of time and with good solution quality. To assess the solution quality, 100 test problems with no crowdsources (thus pure-truck delivery) are generated each consisting of 100 customers randomly scattered around the depot with a 3.5-mile radius. We first consider a benchmark case where solution quality is emphasized but with no consideration of solution time. To this end, we set a high initial temperature by letting $w = 100$ and a very slow cooling rate $c = 0.99$. In addition, we let $N$ be a large number (10000) and do not specify the stopping criteria (thus lines 34-37 will not be used) in Algorithm 2. Figure B.1 (a) shows the resulting total cost in one of the randomly picked test problems. At the start of Algorithm 2, the initial cost is about 2000. After the first few iterations, the total cost goes up to 6000, due to very loose criteria for accepting worse solutions. As the iteration proceeds (temperature decreases), the criteria for accepting worse solutions becomes stricter. After 500 iterations, we achieve a relatively stable total cost, at around 1000.

After experimenting with different combinations of $(w, c, n, N)$ values, $(w, c, n, N) = (10, 0.96, 50, 1000)$ is considered as the final choice. With this combination, we are able to achieve a total cost only 2% higher than the benchmark solution (Figure B.1 (b)). For the same test problem above, the stable total cost is achieved in only one-sixth of the time the benchmark solution requires, in less than 140 iterations. For further comparison, the solution using a simple descent approach (i.e., not accepting worse solutions while iterating) is also tested and presented in Figure B.1 (b). The simple descent approach terminates quickly within 30 iterations but the resulting cost is 20% higher than the benchmark solution. The conclusions are similar for other test problems.
Figure B.1: Solution quality and computation time to solve SPDSTW in the benchmark (a), with \((w, c, n, N) = (10, 0.96, 50, 1000)\) and using a simple descent approach (b).

Table B.1 reports the averaged total cost values and number of iterations in the benchmark, with \((w, c, n, N) = (10, 0.96, 50, 1000)\), and using a simple descent approach.

Table B.1: Averaged total cost and number of iterations in the 100 test problems (a) in the benchmark, (b) with \((w, c, n, N) = (10, 0.96, 50, 1000)\), and (c) using a simple descent approach

<table>
<thead>
<tr>
<th>Cases</th>
<th>Average total cost</th>
<th>Average number of iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benchmark</td>
<td>887</td>
<td>754</td>
</tr>
<tr>
<td>((w, c, n, N) = (10, 0.96, 50, 1000))</td>
<td>921</td>
<td>120</td>
</tr>
<tr>
<td>Using Simple descent</td>
<td>1145</td>
<td>33</td>
</tr>
</tbody>
</table>

References


