Advancing the Urban Parcel Delivery System Using Crowdshipping

By

NABIN KAFLE

M.S., University of Illinois at Chicago, 2015

B.E., Institute of Engineering, Tribhuvan University, 2011

Dissertation

Submitted as partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Civil Engineering
in the Graduate College of the
University of Illinois at Chicago, 2017

Chicago, Illinois

Defense Committee:
Professor Bo Zou, Chair and Advisor
Professor Abolfazl Mohammadian
Professor Jane Lin
Professor Sybil Derrible
Professor Kazuya Kawamura, College of Urban Planning and Public Affairs
This thesis is dedicated to
my wonderful family,

and my fiancée Peniza Thapa

without whom I would have

never made it a success
ACKNOWLEDGEMENTS

It was the January of 2013 when it all started. On a chilly morning, I landed in the Windy City, unaware of the brutal cold the city would be welcoming me with. But the beauty of the city covered in a white blanket of snow made up for its bitter welcome. The cold of the city was the very first inspiration for me to complete my Ph.D. as soon as possible and leave (although I can’t deny that the summer that followed induced the eternal love for the Chi-town). I owe a big time to Chicago, the city that housed the university where I got my Ph.D. degree, the city that inspired me to be who I am today, and the city that taught me important life lessons.

Fast-forwarding the time 4.5 years, today I am defending my Ph.D. thesis. This journey would not have been complete without the boundless support from my Ph.D. advisor Bo Zou, who not only helped me in my research but also provided the moral support to steer my life into good directions. Bo, I still remember those encouraging words, when I was down with failure in the research and was on the verge of quitting. Every time I find myself in similar situations, I go back to your email and find the strength to march along my journey. To be honest, I have had many times where I felt like quitting the Ph.D. journey but the difference from the first time was that I was more experienced on overcoming the challenges.

I am also grateful to all the instructors whose classes I have taken. Specially, the classes of Operations Research I & II, Transportation System Analysis, Infrastructure Management, and Urban Travel Forecasting were some of the important classes that I found to have the direct implication on my research. Although challenging, these classes were utterly interesting, rewarding, and were the ones that kept me tied to books during the initial years. Adding to the list of classes are the Cities and Sustainability, and Sustainable Transportation, which helped me realize the importance and need to think of sustainability issues in the transportation systems.
I would also like to extend my gratefulness to my thesis committee members Jane Lin, Abolfazl (Kouros) Mohammadian, Sybil Derible and Kazuya Kawamura. Thank you for your valuable suggestions that helped to significantly improve the content of this thesis.

Next, I would like to express my gratitude to my lab mates Hossein, Nohseen, Ahmadreza, Farid, Amir, Nasir, Dongwoo, John, Thanakorn, Ashkan and Negar from SEL 4266. Guys, 4266 will never be ‘problem-free’! Good Luck in the future with the heating and cooling problems. Also I wish all the best to other mates Seyyed, Tanvir and Ali from the new lab-ERF 2054. Thanakorn, I am glad that you finally got an escape from the chilly weather of Chicago and are enjoying the sunshine at Los Angeles. We will toast for it when I visit you at Cali, or when we meet again at my next destination.

Last but not the least, this list of acknowledgment would not be complete without mentioning my parents and family members. Dad, I pray every day for your speedy recovery and wish to have you visit here to show the city I lived for 4 and half years. Yeah, the city can be cruel during winter but believe me it is very good during the summer. Mom, you are always the sweetheart. Brother and sister-in-law, I appreciate all your help and company. You two are the sweetest couple. Peniza, words cannot express my appreciation for you.
# TABLE OF CONTENT

1  Introduction .......................................................................................................................... 1

2  Design and Modeling of a Crowdsourced-Enabled System for Urban Parcel Relay and Delivery .......................................................................................................................... 5
   2.1  Introduction ...................................................................................................................... 5
   2.2  Literature review ............................................................................................................ 10
   2.3  Conceptual description of the system ............................................................................ 13
   2.4  Mathematical formulation ............................................................................................. 15
      2.4.1  Crowdsourced decision-making: bid generation and submission ....................... 15
      2.4.2  Truck carrier decision-making: bid selection and truck routes and schedule design .......................................................................................................................... 16
   2.5  Solution approach ........................................................................................................ 27
      2.5.1  Solving WDP ........................................................................................................... 29
      2.5.2  Solving SPDPSTW ............................................................................................... 30
   2.6  Numerical experiments ................................................................................................. 37
      2.6.1  Value for model parameters ...................................................................................... 38
      2.6.2  Small-size problems ................................................................................................. 39
      2.6.3  Large-size problems ................................................................................................. 41
   2.7  Conclusion ..................................................................................................................... 47

Appendix 2A: Proof of equivalence between $\pi(T_u, e_u, l_u)$ and (1.23)-(1.26) ............... 50
Appendix 2B: Choice of $w, c, n, N$ values ........................................................................ 51

3  Designing Mechanisms for Efficient Crowdsourced Urban Parcel Delivery .......... 54
   3.1  Introduction ................................................................................................................... 54
   3.2  Literature Review ......................................................................................................... 57
   3.3  The static problem ....................................................................................................... 60
      3.3.1  Generating feasible shipping jobs ........................................................................... 61
      3.3.2  Shipping job assignment .......................................................................................... 68
      3.3.3  Payment Calculation ............................................................................................... 71
      3.3.4  Optimal bidding strategy in absence of incentive compatible mechanism .... 78
   3.4  Dynamic Case ............................................................................................................. 82
   3.5  Numerical analysis ...................................................................................................... 86
      3.5.1  Static Case .............................................................................................................. 87
**TABLE OF CONTENT (continued)**

3.5.2 Dynamic Case .................................................................................................................. 96
3.6 Conclusion .......................................................................................................................... 101
Appendix 3A ........................................................................................................................... 104

4 Permission for reuse of the previously published materials .......................................... 106
References ................................................................................................................................ 107
VITA ............................................................................................................................................ 114
Table 2-1: Comparison of the results

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

Table 2-3: 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 3-1: DSP shipping cost with mechanism and the optimal cost

Table 3-2: Saving in shipping cost compared to the traditional auction

Table 3-3: Computation time comparison for different processes
Figure 2-1: An illustration of crowdsource-enabled urban parcel relay and delivery .............. 7
Figure 2-2: 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........................................... 34
Figure 2-3: An example of customer, relay point, and crowdsourcee distributions on a four-quadrant plane.......................................................... 42
Figure 2-4: Pure-truck vs. crowdsource-enabled delivery in terms of (a) total cost (b) total truck VMT (c) total periods of service time violations........................................ 43
Figure 2-5: Total periods of service time violations using crowdsource-enabled delivery, with different penalty rates .......................................................... 44
Figure 2-6: Percentage of cost savings using crowdsource-enabled vs. pure-truck delivery, with different penalty rates .......................................................... 45
Figure 2-7: Percentage of cost savings using crowdsource-enabled vs. pure-truck delivery with different truck unit operating costs as compared to the value in Ford Torrey IV and Dan (2016) ........................................................................... 46
Figure 2-8: Percentage of cost savings using crowdsource-enabled vs. pure-truck delivery as time value of crowdsourcees increases .......................................................... 47
Figure 2-9: 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) ........... 52
Figure 3-1: An example showing the how feasible shipping jobs are generated for a crowdsourcee ........................................................................................................ 66
Figure 3-2: Illustration of the periodic crowdsource solicitation and shipment assignment under the dynamic mechanism ................................................................. 84
Figure 3-3: Asymptotic property of the proposed mechanism ........................................... 92
Figure 3-4: Total shipping cost for different values of fixed rate ....................................... 93
Figure 3-5: Comparison of the crowdshipping cost with the traditional truck delivery cost ...... 94
Figure 3-6: Sensitivity of total cost with the maximum number of shipments allowed in a job . 96
Figure 3-7: Near north side neighborhood of Chicago ..................................................... 97
Figure 3-8: Representation of the results for the dynamic case study ................................. 99
Figure 3-9: Sensitivity of the total cost to the length of the periods \( T \) ............................ 99
Figure 3-10: Sensitivity of Total Cost with the guaranteed service time \( G \) and the size of the service area ..................................................................................... 100
Figure 3-11: Variations in the total cost with respect to the DSP location .......................... 101
Figure 3-12 (a) Savings from the IC mechanism for varying number of crowdsourcees, (b) cost of truthfulness for varying number of crowdsourcees ........................................... 105
SUMMARY

This research presents a methodological investigation on the design, modeling, and evaluation of two types of crowdsource-enabled urban parcel delivery systems. The first system couples the existing truck-delivery practice with the locally available crowdsourcees to minimize the cost of last mile delivery. The second system considers the delivery of parcels without the use of any intermediate relay point i.e. crowdsourcees are solely responsible for the pickup of parcels and delivering it to customers’ doorsteps.

For the first system, we consider cyclists and pedestrians as crowdsourcees 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 crowdsourcees express their interests in doing so by submitting bids to the truck carrier. The truck carrier then selects bids and coordinates crowdsourcees’ 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 crowdsourcees to complete the last-leg delivery (first-leg pickup) between customers and selected points for crowdsourcee-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’
SUMMARY (continued)

desired time windows, truck unit operating cost, time value of crowdsourcers, and the crowdsourcer mode.

For the second system, a new mechanism design based model is proposed in which a Delivery Service Provider (DSP) solicits ordinary individuals, i.e., crowdsourcers, who walk, bike, or drive to do delivery in urban areas. As an essential part of the model, the DSP collects private information such as one’s willingness-to-do-crowdshipping (WTDC) and available time window from crowdsourcers in order to assign shipments with the minimum cost. The mechanisms embedded in the crowdshipping model recognize that crowdsourcers may strategically misreport their private information to gain self-interest, and devises a joint shipment assignment-payment scheme that aligns the self-interest of the crowdsourcers with the objective of the DSP. Both static and dynamic cases are investigated. Numerical results demonstrate that the proposed mechanisms will lead to reduced shipping cost compared to the state-of-the-practice crowdshipping. A case study of North Chicago is also presented to further demonstrate the promise of the proposed mechanism to the on-demand delivery in urban areas.
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; Holguín-Veras et al., 2008). The payment is even higher in Manhattan, where a truck accumulates $750 weekly parking fines (Rodrique 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.

In contrast to the push from e-commerce explosion, urban parcel delivery is also pulled by the development of livable urban communities which requires a significant reduction of truck traffic. Many of the pulls are in the form of city ordinances such as restrictions on truck delivery time (Holguín-Veras et al., 2014), routes (Holguín-Veras et al., 2015), size and weight (Qureshi et al., 2013; Holguín-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 (Taniguchi, 2012; PIARC, 2015), and implementing dynamic vehicle routing techniques (Kritzinger et al., 2012; Wolfe and Troup, 2013).

To reconcile the above push and pull, this research introduces two types of crowdshipping or crowdsourced-enabled delivery models. Crowdsources in this research refers to ordinary individuals who are contracted to do short term delivery jobs. Any individual who possesses a car, bike or is willing to do delivery on foot can be a crowdsourced. Substituting the traditional truck delivery with crowdsources is based on the following economic rationale. Truck delivery is not efficient in urban neighborhoods due to the constraints imposed by city ordinances and geographic conditions such as limited truck access time and the presence of one-way roads. However, the constraints generally do not apply to crowdsourced cyclists, drivers, and pedestrians, who have high maneuverability in urban neighborhoods and low cost in delivery.

The first system considers a parcel relay system where truck and crowdsources (only cyclists and pedestrians) coordinate with each other to complete the last mile delivery. The design is from the perspective of a truck carrier, with the focus on operational (e.g., day-before) planning. The core of the proposed system is the substitution of trucks by local crowdsources for the last leg of the last-mile delivery, which occurs through parcel relay at relay points lying closer to the city area. Relay points are necessary as the crowdsources, cyclists and pedestrians in this case, can only travel shorter distances. 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. The design thus leverages the advantages of crowdshipping and truck modes
in the last-leg and the remaining part of the last-mile delivery respectively and is expected to improve the overall delivery efficiency compared to the pure-truck delivery. Further details can be found in Chapter 2 of this dissertation.

The second crowdshipping model, presented in Chapter 3, considers the direct delivery of parcels from a delivery service provider (DSP) to the customers without intermediate relay points. DSPs are the establishments responsible for delivering the online purchased items to the customers—Amazon fulfillment center for example. The crowdsourcers considered for this Chapter are car drivers, cyclists, and pedestrians. The DSP solicits crowdsourcers and assigns shipment demands to minimize the total shipping cost. However, crowdsourcers have the heterogeneous cost of completing the delivery and time availability, making it a non-trivial task for the DSP to determine the optimal crowdsourcer-shipment assignment. We, therefore, present an auction-based platform where crowdsourcers can bid their available time and the cost for which they are willing to do the delivery. While auction based assignments are frequently used in the transportation market (Figliozi et al., 2003; Mesa-Arango and Ukkusuri, 2013), the present study differs in that it specifically considers the strategic behavior of the crowdsourcers who can manipulate their bids to gain a higher payment from the DSP. To prevent such strategic behavior of crowdsourcers, we introduce a mechanism design based assignment and pricing approach that makes truth telling the dominant strategy for crowdsourcers. With the truthful revelation of crowdsourcers’ bids, the DSP can now attain the true optimal cost for the parcel delivery. Further, we also consider the dynamic arrival of the shipment demands and crowdsourcers over the time. To our knowledge, this is the first research of its type to integrate the truthful mechanism for the
dynamic crowdshipping of parcels. Details about the proposed mechanism design based approach can be found in Chapter 3.
2 Design and Modeling of a Crowdsourced-Enabled System for Urban Parcel Relay and Delivery


Permission for reuse of the above publication in the dissertation is obtained from Elsevier under License Number 4047840004845 dated February 14, 2017.

2.1 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; Holguín-Veras et al., 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.

In contrast to the push from e-commerce explosion, urban parcel delivery is also pulled by the development of livable urban communities which requires significant reduction of truck traffic. Many of the pulls are in the form of city ordinances such as restrictions on truck delivery time (Holguín-Veras et al., 2014), routes (Holguín-Veras et al., 2015), size and weight (Qureshi et al., 2013; Holguín-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 (Taniguchi, 2012; PIARC, 2015), and implementing dynamic vehicle routing techniques (Kritzinger et al., 2012; Wolfe and Troup, 2013).

The present study 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 study cyclists and pedestrians who are close to customers – to undertake the last leg of the last-mile delivery.¹ The last leg is connected to the

¹ Unless “pickup” is explicitly mentioned, in the rest of the chapter 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.
rest of the last mile at points where parcel relay between trucks and crowdsourcers 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 crowdsourcers with winning bids about the time that a truck relays parcels at each relay point; (5) crowdsourcers and trucks execute delivery based on the bid selection results and the planned routes and schedule (Figure 2-1).

![Figure 2-1: An illustration of crowdsourcer-enabled urban parcel relay and delivery](image)

The core of the proposed system is the substitution of trucks by local crowdsourcers 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 and low cost in delivery. On the other hand, cyclists and pedestrians can only travel shorter distances, making parcel relay necessary between crowdsourcers 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 trucks 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 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. (Rougès 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 crowdsourcer 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 crowdsourcers and
determine its own truck routes and schedule with the objective of minimizing the overall cost. Third, we consider both facts that multiple crowdsourcers may compete for one delivery job and a crowdsourcer may deliver multiple parcels in a single tour, and consequently introduce a combinatorial bidding process to select crowdsourcers.

Methodologically, this study makes two major contributions. The first contribution is system design that involves crowdsourcing and parcel relay between crowdsourcers 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 crowdsourcer bids selection and relay points locationing with truck routes design. We theoretically prove that under the new design the truck carrier is always no worse off than under pure-truck delivery. 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 10% cost saving and 25% truck vehicle miles traveled (VMT) reduction can be achieved.
We further find that the advantage of the crowdsourced-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 crowdsourced workers, and the crowdsourced mode.

The chapter continues with a review of the literature in Section 2.2 on last-mile delivery that bears relevance to the crowdsourced-enabled system. Section 2.3 provides a conceptual description of the proposed system design, followed by its mathematical formulations in Section 2.4. The solution approach is presented in Section 2.5. Section 0 offers numerical experiments and discussions of the results. Summary of major findings and directions for future research are given in Section 2.7.

2.2 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, 1997; Drezner, 1996; Drezner and Hamacher, 2001) 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, first introduced by (Taniguchi et al., 1999) and suitable for small-sized 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-crowdsourced relay points resemble satellites in 2E-LRP; truck routes are similar to first-echelon routes; and crowdsourced 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 crowdsourced bids and design truck routes and schedule.

2E-LRP is well recognized as an NP-hard problem (Crainic et al., 2009; Nguyen et al., 2012b), 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 (Clarke 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. (2012b) 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., 2012a). 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 study we also seek to develop a heuristic approach to solve the crowdsourcing-enabled parcel relay and delivery system design.

### 2.3 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 2.1 that the functioning of the system consists of five steps, in which decision-making occurs in steps 2 and 3, by crowdsourcerees 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 crowdsourceree 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 crowdsourceree decision making, we consider that each bid includes five pieces of information: (1) the customer(s) that the crowdsourceree plans to serve; (2) the relay point where parcels are transferred between the crowdsourceree and a truck (we term the combination of the customer(s) and the relay point in a bid a **bundle**); (3) the crowdsourceree routing which starts and ends at the crowdsourceree’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 crowdsourceree mode (cycling or walking). The last piece is used to infer the crowdsourceree’s travel speed.

Each crowdsourceree can have a combinatorial number of possible bundles. With the aid of personal computing devices (e.g., mobile Apps), we assume that each crowdsourceree can and will enumerate all feasible bundles. The feasibility of a bundle is constrained by the maximum distance
a crowdsourcee can travel, his/her carrying capacity, and the bidding rule. Subsection 2.4.1 will provide further detail. Provided that bundles are feasible, a crowdsourcee considers two aspects when submitting bids. First, the bid price should be competitive against bids from other crowdsourcees for a given bundle. This requires the route covering the chosen customers and the relay point to be the cost-minimum route for the crowdsourcee. Second, if feasible bundles are all competitively priced and each crowdsourcee is allowed to submit a limited number of bids, then the bids submitted by a crowdsourcee 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 crowdsourcee 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 crowdsourcees, 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 crowdsourcee and the truck carrier.
2.4 Mathematical formulation

2.4.1 Crowdsourcee decision-making: bid generation and submission

As mentioned in Section 2.3, on seeing the posted pickup and delivery jobs each crowdsourcee 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 crowdsourcee’s bid to be within an $R$-mile radius from the crowdsourcee’s origin. Second, a bid can only include a maximum number of customers, and there is a limit on the total parcel weight a crowdsourcee 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, crowdsourcees 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 crowdsourcee.

Note that the generated bids do not involve the arrival time of the crowdsourcee at a customer, but only the routes. Again, this is because at the bid generation step crowdsourcees 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 crowdsourcee 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 crowdsourcee). In this sense, the system will be mostly attractive to crowdsourcees 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 crowdsourcer 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 crowdsourcer. 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 crowdsourcer, because the crowdsourcer 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 2.3 a crowdsourcer will submit $B$ highest priced bids to the truck carrier, where $B$ is the maximum number of bids a crowdsourcer is allowed to submit.

### 2.4.2 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\(^2\) 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 crowdsourcee;

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: Crowdsourcees are not expected to stop en route (from the planning perspective), which can be justified by the fact that in bid generation each crowdsourcee determines bundle bid price assuming no en-route stops. As utility maximizers, crowdsourcees 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

\(^2\) The terms location, node and point are used interchangeably throughout the text.
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 customer 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 \( A' \).^3

### 2.4.2.1 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**

\[ S \quad \text{Set of crowdsourcers} \]

---

^3 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 \( y_{b1}^s = 1 \) and \( y_{b2}^s = 0 \) is symmetric to a solution with \( y_{b1}^s = 0 \) and \( y_{b2}^s = 1 \) (see Section 2.4.2.1 for description of variable \( y_{b}^s \)). 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 (Margot, 2010; Ostrowski et al., 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.
\( L \) 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

**Parameters**

\( B \) Maximum number of bids a crowdsourcer is allowed to submit

\( t_{uv} \) Truck travel time between nodes \( u \) and \( v \)

\( t_{vi}^{bs} \) Time required for crowdsourcer \( s \) to travel from relay point \( v \) to customer \( i \) included in the crowdsourcer’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 crowdsourcer $s$ 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 crowdsourcer) arrival time $T_u$, $e_u$ and $l_u$ (for details see Subsection 0)

**Decision variables**

$T_u$ Service time at node $u$ (by either a truck or a crowdsourcer)

$T^l_d$ Departure time of truck $l$ from the departing depot $d$

$T^l_r$ Arrival time of truck $l$ at the returning depot $r$

$Q_u$ Truck load after serving node $u$

$Q^l_d$ Load of Truck $l$ when leaving the departing depot (node) $d$

$x^l_{uv}$ Binary variable taking value 1 if truck $l$ travels consecutively from node $u$ to $v$ and 0 otherwise

$y^s_{bv}$ Binary variable taking value 1 if the bid $b$ submitted by crowdsourcer $s$ 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):
\[
\min \sum_{u \in N \cup A'} \sum_{v \in N \cup A' \cup \{d\}} C_{uv} x_{uv}^l + \sum_{u \in A'} \sum_{s \in S} \sum_{b=1}^{B} P_{bu}^s y_{bu}^s + \sum_{u \in N \cup A' \cup \{r\}} \pi(T_u, e, l_u)
\]

s.t. Truck routes constraints

\[\sum_{u \in N \cup A'} \sum_{l \in L} x_{uv}^l \leq 1 \quad \forall v \in N \cup A' \cup \{d\}\]
\[\sum_{v \in N \cup A' \cup \{r\}} \sum_{l \in L} x_{uv}^l \leq 1 \quad \forall u \in N \cup A' \cup \{d\}\]
\[\sum_{u \in N \cup A' \cup \{d\}} \sum_{l \in L} x_{uv}^l \leq |L|\]
\[\sum_{u \in N \cup A'} \sum_{l \in L} x_{uv}^l = \sum_{v \in N \cup A' \cup \{r\}} \sum_{l \in L} x_{vw}^l\quad \forall u \in N \cup A', l \in L\]

Truck schedule constraints

\[(T_u + \xi + t_{uv} - T_v) \sum_{l \in L} x_{uv}^l \leq 0 \quad \forall u \in N \cup A', v \in N \cup A'\]
\[(T_d + t_{dv} - T_v) x_{dv}^l \leq 0 \quad \forall v \in N \cup A', l \in L\]
\[(T_u + \xi + t_{ur} - T_r) x_{ur}^l \leq 0 \quad \forall u \in N \cup A', l \in L\]
\[\delta_i^b s(T_v + \xi + t_{vi} - T_i) y_{bv}^s = 0 \quad \forall s \in S, b = (1 \ldots B), v \in A', i \in N_D\]
\[\delta_i^b s(T_v - t_{vi} - T_i) y_{bv}^s = 0 \quad \forall s \in S, b = (1 \ldots B), v \in A', i \in N_P\]

Truck load and capacity constraints

\[Q_d^l = \sum_{u \in N \cup A' \cup \{d\}} \sum_{v \in N \cup A'} x_{uv}^l q_{vu}^D \quad \forall l \in L\]
\[Q_d^l \leq K \quad \forall l \in L\]
\[(Q_d^l + q_v - Q_v) \sum_{l \in L} x_{dv}^l \leq 0 \quad \forall v \in N \cup A'\]
\[(Q_u + q_v - Q_v) \sum_{l \in L} x_{uv}^l \leq 0 \quad \forall u \in N \cup A', v \in N \cup A'\]
\[Q_u \leq K \quad \forall u \in N \cup A'\]
Bid selection constraints

$$\sum_{v \in N \cup A'} \sum_{l \in L} x_{lv} + \sum_{v \in A'} \sum_{s \in S} \sum_{b=1}^B \delta_{b} y_{bv}^s = 1 \quad \forall i \in N$$ (1.16)

$$\sum_{b=1}^B \sum_{s \in S} y_{bv}^s \leq \sum_{u \in N \cup A'(d)} \sum_{l \in L} x_{uv}^l \quad \forall v \in A'$$ (1.17)

$$\sum_{b=1}^B \sum_{v \in A'} y_{bv}^s \leq 1 \quad \forall s \in S$$ (1.18)

Non-negativity and binary constraints of decision variables

$$x_{uv}^l \in \{0,1\} \quad \forall u \in N \cup A' \cup \{d\}, v \in N \cup A' \cup \{r\}, l \in L$$ (1.19)

$$y_{bv}^s \in \{0,1\} \quad \forall b = (1...B), s \in S, v \in A'$$ (1.20)

$$Q_u, Q_d^l, T_u, T_d^l, T_r^l \geq 0 \quad \forall u \in N \cup A', l \in L$$ (1.21)

The objective (1.1) minimizes the sum of truck operating cost (1st term), payment to selected crowdsourcers (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 2.4.2.2.

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. 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 crowdsourcer to 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 crowdsourcer). The equality sign reflects that crowdsourcers do not stop en route. Constraint (1.10) does not involve \( \xi_v \) because parcel transfer is made after a parcel-pickup crowdsourcer arrives at the relay point.

The third group of constraints relates to truck load and capacity, which follows Desrochers et al. (1988). 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 crowdsourcee. 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 crowdsourcee wins at most one bid. This is to avoid the circumstance that when a crowdsourcee wins multiple bids, it may require pickup/delivery jobs with overlapping (thus conflicting) times.

The last group of constraints specifies the binary conditions of $x^l_{uv}$’s and $y^g_{bv}$’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.

### 2.4.2.2 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, $\varepsilon_u$ 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)

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

(1.24)

$$\tau_u \geq T_u - l_u \quad \forall u \in N$$

(1.25)

$$\varepsilon_u, \tau_u \geq 0 \quad \forall u \in N$$

(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 2A.

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):

$$(T_u + \xi_u + t_{uv} - T_v) \leq \left(1 - \sum_{l \in L} x_{uv}^l\right) M \quad \forall u \in N \cup A', v \in N \cup A'$$

(1.6a)
\[(T_d^l + t_{dv} - T_v) \leq (1 - x_{dv}^l)M \quad v \in N \cup A', l \in L \] (1.7a)

\[(T_u^l + \xi_u + t_{ur} - T_u) \leq (1 - x_{ur}^l)M \quad u \in N \cup A', l \in L \] (1.8a)

\[(Q_d^l + q_v - Q_v) \leq (1 - \sum_{l \in L} x_{dv}^l)M \quad \forall v \in N \cup A' \] (1.13a)

\[(Q_u + q_v - Q_v) \leq (1 - \sum_{l \in L} x_{uv}^l)M \quad \forall u \in N \cup A', v \in N \cup A' \] (1.14a)

where \(M\) is a large constant.

Non-linear constraints (1.9)-(1.10), which have equality signs, will each require an additional constraint to be linearized:

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

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

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

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

**MILP-(BS+TRS)**

\[
\begin{align*}
\min & \sum_{u \in N \cup A'} \sum_{b \in \mathcal{U}(d)} C_{uv}^l x_{dv}^l + \sum_{u \in A'} \sum_{s \in S} \sum_{b \in B} P_b^s y_{bu}^s + \\
& \sum_{u \in N} \left( \epsilon_u + \tau_u \right)
\end{align*}
\] (1.1a)

s.t (1.2)-(1.5)
(1.6a)-(1.8a)
(1.9a,b)-(1.10a,b)
(1.11)-(1.12)
(1.13a)-(1.14a)
(1.15)-(1.21)
(1.24)-(1.26)

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.

### 2.5 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 crowdsourcées to serve customers. The solution indicates the relay points to be used by the selected crowdsourcées. Based on the solution, calculate the payment to each selected crowdsourcer.

Step 2: Solve SPDPSTW in which trucks visit the selected relay points as well as customers not covered by the selected crowdsourcées. Add the resulting truck operating cost and time penalty cost to the payment to crowdsourcées 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.

2.5.1 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:

\[
\begin{align*}
\min & \sum_{u \in A} \sum_{s \in S} \sum_{b \in B} p_{bu}^s y_{bu}^s + M \sum_{i \in N} z_i \\
& \quad z_i + \sum_{v \in A} \sum_{s \in S} \sum_{b \in B} \delta_i^{bs} y_{bv}^s = 1 \quad \forall i \in N \quad (2.2) \\
& \quad \sum_{v \in A} \sum_{b \in B} y_{bv}^s \leq 1 \quad \forall s \in S \quad (2.3) \\
& \quad y_{bv}^s \in \{0, 1\} \quad \forall b \in B, s \in S, v \in A \quad (2.4) \\
& \quad z_i \in \{0, 1\} \quad \forall i \in N \quad (2.5)
\end{align*}
\]

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 crowdsourcee 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 crowdsourcer 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 2.4.1, 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.

2.5.2 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 crowdsourcer 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 crowdsourced travel time from \( u' \) to \( u \). \( e_u = l_u \) 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 (2002) 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 (Daganzo, 2005; Mitra et al., 1985; Van Laarhoven and Aarts, 1987). It is simple, flexible, well suited for solving VRPs, and close to being an ideal general purpose tool for fine tuning (Robust 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.

2.5.2.1 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 \(\pi(T_u, e_u, l_u)\):

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

2.5.2.2 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 2-2). 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 2-2 (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 2.5.2.3) 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 2-2 (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.

![Image](image.png)

Figure 2-2: 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

### 2.5.2.3 Acceptance criteria

The acceptance criteria for a move when truck cost increases reflects the Simulated Annealing (SA) 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 $\Delta cost$, the probability of acceptance $e^{\frac{\Delta cost}{T}}$ 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 $\Delta cost$. 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 SA 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 0 will provide further details on how we choose $w$ and $c$.

### 2.5.2.4 Overall algorithm to solving SPDSPTW

Based on the discussions in Subsections 2.5.2.1-2.5.2.3, 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. \( \text{best\_route}_0 = \text{route} \)
3. \( \text{best\_cost}_0 = \text{cost} \)
4. \( \text{Temperature} = -w/100\cdot(cost)/\ln(0.5) \)
5. \( \text{for } i = 1 \text{ to } N \)
6. \( \text{Temperature} = \text{Temperature} \cdot \text{cooling\_rate}^{(i-1)} \)
7. Function 2-opt_move (route, cost)
8. \( \text{Acceptance\_probability} = \exp[(\text{best\_cost}_{i,1} - \text{cost})/\text{Temperature}] \)
9. if \( \text{Acceptance\_probability} > \text{rand}[0,1] \)
10. \( \text{best\_route}_i = \text{route} \)
11. \( \text{best\_cost}_i = \text{cost} \)
12. else
13. \( \text{best\_route}_i = \text{best\_route}_{i-1} \)
14. \( \text{best\_cost}_i = \text{best\_cost}_{i-1} \)
15. End if

\( ^4 \) 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.
16. **Function** node_interchange(route, cost)
17. Acceptance_probability=exp((best_cost_i-1-cost)/Temperature)
18. if Acceptance_probability > rand[0,1]
19. best_route_i = route
20. best_cost_i = cost
21. Else
22. best_route_i = best_route_i-1
23. best_cost_i = best_cost_i-1
24. End if
25. **Function** node_relocation(route, cost)
26. Acceptance_probability=exp((best_cost_i-1-cost)/Temperature)
27. if Acceptance_probability > rand[0,1]
28. best_route_i = route
29. best_cost_i = cost
30. else
31. best_route_i = best_route_i-1
32. best_cost_i = best_cost_i-1
33. End if
34. if i>n
35. best_cost_i = best_cost_i-1 =...= best_cost_i-n+1
36. End for
37. End if
38. End
39. Output best_route_i and best_cost_i

### 2.6 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.

2.6.1 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 (2016). We consider a unit truck operating cost of $68.09/hour following (Ford Torrey IV and Dan, 2016). 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 2.6.3, 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 crowdsourcing 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 crowdsourcee is allowed to submit 2 bids; in a large-size problem, the maximum number of bids allowed for a crowdsourcee is 5. The values for \( w, c, n, \) and \( N \) in Algorithm 2 are \((w,c,n,N) = (10, 0.96, 50, 1000)\). This parameter choice is the result of jointly considering solution quality and computation time, as further shown in Appendix 2B.

### 2.6.2 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 crowdsourcees ranging from 2 to 8 (Table 2-1). 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 crowdsourcee origins are randomly generated within a 2-mile radius of \((0, 0)\). Crowdsourcees 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 2-1 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 2-1. 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 2.5.2.2), 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 2-1: 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></td>
<td></td>
<td></td>
<td>Total Cost</td>
<td>CPU time (sec)</td>
<td>Total Cost</td>
<td>CPU time (sec)</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.
2.6.3 Large-size problems

2.6.3.1 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 2-3). 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 crowdsources, which are also randomly scattered around the centroid with the same radius. Crowdsources 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.
2.6.3.2 Results and sensitivity analysis

Figure 2-4 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 crowdsourceree-enabled delivery. At the median, total cost saving is around 9.25% with crowdsourcing.\(^5\) Median truck VMT will be reduced even more significantly, by about 24%. These are in spite of some slight increase in the

\(^5\) By Observation 1, the truck carrier will not be worse off using crowdsourcerees. On the other hand, because here heuristic solution approaches are used, it cannot be guaranteed that the total cost of crowdsourceree-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 crowdsourceree-enabled delivery. Nevertheless, for these four instances the cost difference between crowdsourceree-enabled and pure-truck deliveries is very small, lesser than 2%. This also applies to explaining the occurrence of cost negative cost savings in Figure 2-6 to Figure 2-8.
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 2-2). Averaged over the 100 instances, the number of customers served by crowdsourcers is 54 (out of 100) and the median total payment from the truck carrier to the crowdsourcers is $128.

![Figure 2-4](image)

**Figure 2-4:** Pure-truck vs. crowdsource-enabled delivery in terms of (a) total cost (b) total truck VMT (c) total periods of service time violations

<table>
<thead>
<tr>
<th>Null hypothesis (H$_0$)</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 H$_0$</td>
</tr>
<tr>
<td>Equal mean of total truck VMTs</td>
<td>-24.21</td>
<td>Reject H$_0$</td>
</tr>
<tr>
<td>Equal mean of total periods of service time violations</td>
<td>0.089</td>
<td>Fail to reject H$_0$</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 $184 to the truck carrier (recall that we use a penalty rate of $2/hr). This total time periods violated will, intuitively, decrease as the penalty rate increases. Figure 2-5 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 2-6, the median percentage cost savings using crowdsourcing drop from 9.25% with a penalty rate of $2/hr to only 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.

Figure 2-5: Total periods of service time violations using crowdsourceree-enabled delivery, with different penalty rates
Besides penalty rate, a few other model parameters may also affect the attractiveness of crowdsourced-enabled delivery. In the remainder of this subsection, we investigate the impact of truck unit operating cost, the time value of crowdsources, and crowdsourced delivery mode on system performance. In the previous experiments, the truck unit operating cost is adopted from Ford Torrey IV and Dan (2016). 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 Ford Torrey IV and Dan (2016) value. Because trucking is more expensive than cycling/walking, we expect greater cost advantage of crowdsourced-enabled delivery with higher truck unit operating cost. This is confirmed in Figure 2-7: the median percentage cost savings with crowdsourced-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 2-6: Percentage of cost savings using crowdsourced-enabled vs. pure-truck delivery, with different penalty rates
Figure 2-7: Percentage of cost savings using crowdsource-enabled vs. pure-truck delivery with different truck unit operating costs as compared to the value in Ford Torrey IV and Dan (2016)

The time value of crowdsourcleees 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 crowdsourcleees. Holding truck unit operating cost constant, an increase in crowdsourcee time value will compromise the competitiveness of crowdsource-enabled delivery. As shown in Figure 2-8, the median percentage cost savings would decrease from 9.25% to 6.5%, 5% and 2% if the time value of crowdsourcleees is increased by 25%, 50%, and 75% respectively.
So far we have considered cycling as the crowdsourcee mode. If crowdsourcees are instead pedestrians, who are slower and have less carrying capacity, then crowdsourcee-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 crowdsourcees will be used, with median total payment from the truck carrier to the selected crowdsourcees decreased from $128 to $29. The median percentage cost savings is only 2%.

2.7 Conclusion

In this study, we propose a crowdsourcee-enabled system design for urban parcel delivery. In this design, a truck carrier posts pickup and delivery jobs and relay points online. Local crowdsourcees 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 crowdsourcees for the last leg, total cost to
the truck carrier, which consists of truck operating cost, payment to crowdsourcers, 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 crowdsourcer-enabled system involves solving Undirected Travel Salesman’s Problems for crowdsourcer 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 crowdsourcer mode. Over half of the customers will be served by crowdsourcers. The attractiveness of the crowdsourcer-enabled system depends on factors such as penalty rate for serving customers outside the desired time windows, truck unit operating cost, time value of crowdsourcers, and crowdsourcer mode. Overall, lower penalty rate, greater truck unit operating cost, lower time value of crowdsourcers, and using cyclists instead of pedestrians will enhance the cost competitiveness of the crowdsourcer-enabled system.
This study 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 study we assume that crowdsourcing undertakes the assigned delivery jobs irrespective of the time of a day. This may be justified on the ground that crowdsourcers, whose time value is low, typically have flexible schedules. On the other hand, it may be possible that crowdsourcers have limited time availabilities. In this case, either crowdsourcers will submit their time availability information as part of the bids, or the bidding process may take multiple rounds to meet the constraints on crowdsourcee time availability. Second, although the present study investigates pre-operation 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 crowdsource-enabled system is receptive by local communities and customers.
Appendix 2A: 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 $\epsilon_u \geq e_u - T_u$, which is negative because $e_u < l_u < T_u$. Combining with (1.26), the overall constraint for $\epsilon_u$ is $\epsilon_u \geq 0$. 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 $P(\epsilon_u + \tau_u)$) given $T_u, e_u, l_u$, it must be that $\epsilon_u = 0$ and $\tau_u = T_u - l_u$. 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 $\epsilon_u$ and $\tau_u$ will be $\epsilon_u \geq 0$ and $\tau_u \geq 0$. To minimize time penalty for vehicle $u$ given $T_u, e_u, l_u$, it must be that $\epsilon_u = \tau_u = 0$. 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 $\epsilon_u$ and $\tau_u$ will be $\epsilon_u \geq e_u - T_u$ and $\tau_u \geq 0$. To minimize the time penalty for vehicle $u$ given $T_u, e_u, l_u$, it must be that $\epsilon_u = e_u - T_u$ 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.$\square$
Appendix 2B: 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 crowdsourcées (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 2-9(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 2-9(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 2-9 (b). The simple descent approach terminates quickly within 30 iteration but the
resulting cost is 20% higher than the benchmark solution. The conclusions are similar for other test problems.

Figure 2-9: 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 2-3 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 2-3: 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>
3 Designing Mechanisms for Efficient Crowdsourced Urban Parcel Delivery

3.1 Introduction

Driven by the rapid e-commerce growth, city logistics has become increasingly on-demand with small package size. For example, more than 1 and 30 million of items ordered on Amazon – many of them placed by residents in urban areas – are available for the same-day and two-day delivery (Wahba, 2016). Among these items, 86% weigh less than 5 pounds (Guglielmo, 2013). To accommodate the trend and thanks to recent technological advances in ubiquitous mobile computing and communication, crowdshipping has emerged as an attractive new form of urban delivery (Postmates, Deliv, PiggyBaggy, Amazon Flex, Jing Dong, to name a few). Different from traditional delivery which typically relies on full-time employees and self-owned vehicle assets, in crowdshipping a delivery service provider (DSP) solicits ordinary people who have available time and can walk, bike, or drive a car to perform delivery in exchange for some money. This gives crowdshipping a cost advantage over traditional truck-based delivery due to reduced needs for maintaining vehicle fleets, fuel costs, employed drivers, and warehouse infrastructure. In addition, crowdshipping is highly flexible in time, maneuverable in urban space (e.g., narrow streets, alleys), and environmentally friendly, which further its appeal for urban delivery.

To solicit ordinary people, hereafter termed crowdsourcers, to perform delivery, the state of the practice for crowdshipping either have a DSP offer a fixed pay rate to crowdsourcers (Archetti et al., 2016; Arslan et al., 2016) or through auction (Zipments, 2016; Caldwell, 2014; FBIC group, 2015; Rougès and Montreuil, 2014). In the auction approach, private information on the willingness-to-do-crowdshipping (WTDC) is submitted by crowdsourcers to the DSP who then uses the information to select crowdsourcers and assign shipments to them. However, both
approaches can entail inefficiencies. For the fixed rate approach, setting an appropriate, uniform pay rate is critical given that the WTDC is heterogeneous among crowdsourcers. A too high rate obviously increases total shipping cost. On the other hand, a too low rate can result in the DSP not having enough crowdsourcers, yet crowdsourcers who are willing to perform delivery at a low pay rate will still be overpaid. The auction approach implicitly assumes that crowdsourcers truthfully report private information, which is not necessarily true. In fact, a crowdsourcer can strategically misreport private information to gain competitive advantage. Example 1 below illustrates this. The consequence of untruthful reporting of private information will be borne by the DSP, also leading to greater overall shipping cost.

**Example 1:** We consider two crowdsourcers A and B available at the DSP location. Both are interested in deliver a shipment. The DSP solicits the Value of Time (VOT) from each crowdsourcer, which is a proxy for WDTC, and chooses the crowdsourcer with the lower VOT. The true VOT for A and B are $5/hr and $10/hr respectively. If both report their VOT truthfully, crowdsourcer A would be chosen to perform the delivery. But A can be strategic to gain further. Having some probabilistic knowledge of B’s VOT, A may report an inflated value (Section 3.3.4 offers some further mathematical detail on how to determine the optimal information reporting strategy of a crowdsourcer). For instance, if A reports $9/hr (and B reports truthfully), then the DSP still chooses A but ends up paying $4 more each hour for the delivery.

The present paper contributes to the crowdshipping literature by proposing a new, mechanism design based approach which has the potential to significantly reduce the inefficiencies in soliciting crowdsourcers. This approach explicitly accounts for the heterogeneity in the WTDC and available time window among crowdsourcers and – through the combined design of a crowdsourcer-shipment assignment scheme and a crowdsourcer payment rule – aligns the self-
interest of crowdsourcers with the objective of the DSP, which is to minimize total shipping cost. Under the mechanism, a crowdsourcee will find it his/her best interest to truthfully report private information. The consequent cost reduction from implementing the proposed mechanisms compared to fixed rate and traditional auction approaches is demonstrated through extensive numerical experiments.

We begin our investigation with a static problem in which all shipment demands and crowdsourcers who are willing to perform delivery are known in advance. In this case, the crowdsourcee selection and the assignment of crowdsourcers to shipments will be done in one shot. Because it is possible for a crowdsourcer to deliver multiple shipments in a tour, the premise of crowdsourcee selection and crowdsourcee-shipment assignment will be to understand what combinations of shipments, which we term shipping jobs, will be feasible for each crowdsourcee. To this end, a process to efficiently generate feasible shipping jobs is introduced. To have crowdsourcers report truthfully their private information, a DSP must provide incentives – in the form of compensation under the mechanism. In addition to show how the incentives induce truthful information reporting, we theoretically investigate the asymptotic properties of the compensation with respect to the size of the crowdshipping system and the motivation of crowdsourcers to misreport. The theoretic investigation is accompanied by extensive numerical analysis of the crowdshipping system performance.

The static case lays the foundation for the more complex situation that shipment demands show up over time, and crowdsourcers also dynamically enter and exit the crowdshipping system because of their different time availability. Simple application of the mechanism in the static case no longer guarantees that crowdsourcers do not misreport private information. To still ensure
truthful reporting, an extended mechanism with additional return requirement is conceived. Under this mechanism, the relationship among the guaranteed time for delivery, the frequency of dispatching crowdsources and the service coverage of crowdshipping, and the tradeoffs are explored both theoretically and numerically using the north side of Chicago as the case study area. The results offer useful insights into future design of on-demand delivery service in terms of the delivery time guarantee and the optimal depot/fulfillment center location.

The remainder of the chapter is organized as follows. Section 3.2 provides the review of related works. The formulation of the static mechanism problem and its solution procedure are introduced in Section 3.3. Section 3.4 details the dynamic mechanism. Results of numerical experiments are presented and discussed in Section 3.5. Conclusion and directions for further research are offered in the end.

3.2 Literature Review

This section reviews literature in two relevant fields. The first field is about different approaches that exist or have been studied for crowdshipping in the context of urban parcel delivery. The second field pertains to combinatorial auction and mechanism design, which have garnered increasing attention in transportation resource allocation.

Despite the rapid development of crowdshipping in practice, the scientific literature remains scarce, most of which has appeared only in the past couple of years. In general, there are two types of crowdsources: dedicated and opportunistic. The first type of crowdsources self-determine their schedule for performing delivery and may consider it as either a part-time or a full-time job. Rougès and Montreuil, (2014) argue that the most promising crowdshipping start-ups rely primarily on dedicated crowdsources. The second type of crowdsources are those who are
already in their journey and are willing to make small detours to pick up and deliver packages and gain some extra money on the way. The majority of the literature focuses on the second type of crowdsourcees. Archetti et al. (2016) investigate the possibility of having opportunistic crowdsourcees each performing up to one delivery on their way. The problem is formulated as a variant of the capacitated vehicle routing problem and solved with a multi-start heuristic. It is found that delivery cost and VMT can be reduced by up to 40% and 68%. Wang et al. (2016) also consider opportunistic crowdsourcees but allow more than one package to be delivered by a crowdsourcee. A network min-cost flow problem along with various pruning techniques is proposed to efficiently solve large-scale instances. Arslan et al. (2016) model the match of shipping tasks and drivers as a new pickup and delivery problem in a rolling horizon framework, and conclude that using crowdsourcees brings economic and environmental benefits particularly for fast delivery services. Kafle et al. (2017) look into a different context. Instead of having crowdsourcees perform delivery all the way from pickup to delivery, a relay-based design is explored in which trucks are still used for line-haul transportation between a depot and relay points, where crowdsourcees exchange packages with trucks and perform the first/last leg of the pickup/delivery. Under the design, total delivery cost will be reduced by 10% and truck VMT by 25%. Note that the above studies either use fixed pay rate (Archetti et al., 2016; Wang et al., 2016; Arslan et al., 2016) or assume that crowdsourcees truthfully report private information (Kafle et al., 2017). Lastly, a platform model based on game-theoretic analysis is formulated and analyzed in Kung and Zhong (2016). Focusing on platform profitability, the authors find that membership-based pricing generally yields greater profit than the transaction-based pricing and cross subsidization.
Combinatorial auction has been widely considered in truckload procurement (Figliozzi et al., 2003; Mesa-Arango and Ukkusuri, 2013; Caplice and Sheffi, 2006; Figliozzi et al., 2005; Ledyard et al., 2002; Song and Regan, 2005; Song and Regan, 2003; Hernández et al., 2011; Wang and Xia, 2005). The problem is a “reverse auction” in which the winning bidders, which are truck carriers, receive the payment from the auctioneer (a shipper or a third party on the shipper’s behalf) for performing assignments. In participating in the auction, a truck carrier needs to generate bids for each possible combination of shipments. The bid generation problem has been discussed in detail in (Song and Regan, 2005; Lee et al., 2007; Chang, 2009). In our problem, an analogy may be made between a truck carrier and a crowdsourcer, who also needs to know all the feasible shipping jobs to him/her and the associated cost.

An implicit assumption underlying the above combinatorial auction studies are that truck carriers bid based on their true cost information, which is private. Since the objective of a truck carrier (maximizing net revenue, which is payment from the auctioneer minus cost) and that of the auctioneer (minimizing total payment to the selected truck carriers) are different, truck carriers may bid strategically resulting in inefficient allocation of shipping lanes to truck carriers (Xu and Huang, 2014). To overcome this, Huang and Xu (2013) propose a one-sided combinatorial VCG (Vickrey, 1961; Clarke, 1971; Groves, 1973) auction which is both allocatively efficient (AE), i.e., social welfare is maximized, and incentive compatible (IC), i.e., truthful bid is the dominant strategy for each bidder. However, carriers are restricted to expressing a bid for a single bundle. Later, Xu and Huang (2014) consider the problem where carriers can express bids for any bundles they are interested in. The authors develop a VCG auction and a sequential primal-dual Vickrey
auction, both of which are AE and IC. The work is further extended to intermodal e-commerce logistics accounting for the transaction costs of intermodal services in Xu et al. (2015).

Yet an important gap still exists in the literature of reverse combinatorial auction: AE and IC auctions in a dynamic environment. Dynamics is reflected in the arrival of both truck carriers and truckloads. Xu and Huang (2014) state that it may be extremely difficult to obtain AE and IC reverse combinatorial auction when future demands are uncertain. This suggests a severe limitation for the existing AE and IC mechanisms to be directly applied to crowdshipping, which has crowdsourcers and shipment demands constantly arriving over time. Predicting future crowdsourcers and shipment demands with accuracy is generally difficult. In addition, active crowdsourcers also dynamically leave the crowdshipping system when reaching the end of their available time. In this paper, a new mechanism is designed that accounts for the above features, in the context of time guaranteed delivery.

3.3 The static problem

This section investigates the static problem where all information about shipment demands and crowdsourcers is known. The static problem may apply to day-before planning in the context of next-day delivery. Each shipment demand \( u \) has the delivery destination location, and the delivery time window \( (e_u, l_u) \). The DSP requires each crowdsourcer \( i \) who is interested in crowdshipping to submit his/her private information as a bid in a four-tuple \( \theta_i = (a_i, d_i, r_i, m_i) \).

\( a_i \) and \( d_i \) denote the start and the end of crowdsourcer \( i \)'s available time window. For \( a_i \), we consider that the availability is with respect to the DSP location. \( r_i \) is crowdsourcer \( i \)'s VOT in \$/hr reflecting his/her WTDC. \( m_i \) is the mode of crowdsourcer \( i \) (walking, cycling, driving). The DSP computes the cost of a shipping job assigned to crowdsourcer \( i \) by multiplying the trip
distance of the job, the reported $r_i$, and the speed of mode $m_i$. A shipping job may contain one or multiple shipments. Of course, the assignment of a shipping job to a crowdsourcee is constrained by the delivery time window of the shipments in the job and the available time window of the assigned crowdsourcee.

Recall that a mechanism combines a payment rule with a shipment assignment scheme. The premise of a mechanism is thus to generate feasible shipping jobs based on known shipment demands. Below we present in sequence the problems of generating feasible shipping jobs, assigning jobs to crowdsourcees, and designing the payment rule.

### 3.3.1 Generating feasible shipping jobs

The feasibility of shipping jobs is crowdsourcee-specific. Let us start by looking at what single-shipment jobs are feasible for each crowdsourcee. For a given shipment-crowdsourcee pair, the DSP needs to check: (1) if the start of the crowdsourcee’s available time window plus the shipping time is no later than the end of the shipment’s delivery time window, and (2) the end of the crowdsourcee’s available time window is no earlier than the start of the shipment’s delivery time window. Here shipping time is the travel time from the DSP location to the shipment destination.

Once all feasible shipment-crowdsourcee pairs are identified, larger feasible shipping jobs will be formed by progressively adding shipments to the existing feasible jobs and checking the feasibility. The check involves solving a Traveling Salesman Problem with Time Windows and Crowdsourcee Availability (TSPTW&CA), as described below.
3.3.1.1 TSPTW&CA formulation

TSPTW&CA is a modified TSPTW problem which includes an additional constraint that all shipments contained in a shipping job must be delivered within the available time window of the crowdsourcee. The objective is to minimize the total routing distance from the DSP location and visiting all shipments destinations. For the crowdsourcee, distance minimization is equivalent to minimizing total routing cost. If no solution is found for the problem, then the shipping job is infeasible for the crowdsourcee.

The lists of notations and the formulation for the TSPTW&CA are given as follows:

Sets

- $C$: Set of shipments in a job
- $\{d, r\}$: Departing and returning location

Parameters

- $D_{uv}$: Distance between locations $u$ and $v$
- $t_{uv}$: Travel time between locations $u$ and $v$
- $e_u, l_u$: Early and latest allowable delivery time of shipment demand $u$
- $a_i, d_i$: Start and end of crowdsourcee $i$’s available time window

Decision Variables

- $X_{uv}$: Binary variable taking value 1 if the crowdsourcee travels consecutively from location $u$ to $v$ and 0 otherwise
- $T_u$: Continuous variable representing the service time at location $u$
With these notations, the TSPTW&CA problem for crowdsourcee $i$ is formulated as the following mixed integer nonlinear program (MINLP):

\[
\begin{align*}
\text{min} & \sum_{u \in C \cup \{d\}} \sum_{v \in C \cup \{r\}} D_{uv} x_{uv} \\
\text{s.t.} & \sum_{u \in C \cup \{d\}} x_{uv} = 1 \quad \forall v \in C \cup \{r\} \\
& \sum_{v \in C \cup \{r\}} x_{uv} = 1 \quad \forall u \in C \cup \{d\} \\
& x_{uv}(T_u + t_{uv} - T_v) \leq 0 \quad \forall u, v \in C \\
& e_u \leq T_u \leq l_u \quad \forall u \in C \\
& T_d \geq a_i \\
& T_r \leq d_i \\
& x_{uv} \in \{0, 1\} \quad \forall u \in C \cup \{d\}, \forall v \in C \cup \{d\} \\
& T_u \geq 0 \quad \forall u \in C \cup \{d, r\}
\end{align*}
\]

The objective function (1.1) minimizes the total distance traveled by crowdsourcee $i$ to visit all shipments contained in the shipping job, $C$. Constraints (1.2) and (1.3) indicate that all shipments in the job must be fulfilled exactly once. Constraint (1.4) calculates the change in time when a crowdsourcee travels from $u$ to $v$. This is a nonlinear constraint and valid only when $x_{uv}$ is equal to 1. Constraint (1.4) is also responsible for the sub-tour elimination (Kafle et al., 2017). Constraint (1.5) specifies each shipment must be served within their service time window $[e_u, l_u]$. Constraint (1.6) specifies that crowdsourcee $i$ can leave the departing location only after his/her early available time $a_i$. Similarly, constraint (1.7) specifies that crowdsourcee $i$ should arrive at
the return location by his/her latest available time \( d_i \). Constraint (1.8) imposes the binary condition for the decision variable \( x_{uv} \) and constraint (1.9) specifies that the time of visit at any shipment destination including the departing and returning depot should be greater than 0.

Recall that we consider all crowdsources are available at the DSP location at \( a_i \), thus the departing location is the location of the DSP for every crowdsources. Similarly, we consider each crowdsourcee’s final location as the location of the last customer to be served in their shipping job. This can be done by making the distance from every customer to the returning depot \( r \) equal to zero. The formulation can also be easily adapted to life-style crowdsources by changing the departure and returning depot to their respective origin and the destination location and recalculating \( D_{uv} \)s accordingly.

The TSPTW&CA problem needs to be solved for each crowdsourcee and for every job feasible to the crowdsourcee. As the TSPTW&CA problem can be reduced to both the TSPTW problem (by setting \( a_i = 0 \) and \( d_i = \infty \)) and the TSP problem (by setting all \( e_u \)s to 0 and \( l_u \)s to \( \infty \) along with \( a_i = 0 \) and \( d_i = \infty \)), TSPTW&CA can be shown to be NP-complete. Savelsbergh (1985) showed that even finding a feasible solution for a TSPTW is NP-complete. The problem of testing the feasibility of jobs (and hence generating all feasible jobs) is therefore a NP-complete problem and no algorithm exists to solve the problem within the polynomial time. However, in our case due to the limited time availability of crowdsources, the number of shipments that can be included in a job is relatively small. In addition, we also impose a hard constraint on the maximum number of shipments that can be carried by a crowdsourcee. This will be reflected later in the Algorithm 1. Practically, this is a valid assumption because the local crowdsources have the limited geographic coverage and weight limitation obscuring them to serve a large number of
shipments. The TSPTW&CA problem with small number of shipments can be solved to optimality within a fraction of second with the commercially available MIP solvers like CPLEX, after linearizing the constraint (1.4). Constraint (1.4) can be linearized as follow with the help of the big M method (Desrochers et al., 1988).

\[ T_u + t_{uv} - T_v \leq M(1 - X_{uv}) \quad \forall u, v \in \mathcal{C} \cup \{d, r\} \]  

(1.4a)

In addition to above assumptions, the following proposition is useful to reduce the number of shipping jobs to be enumerated.

**Proposition 1:** If a shipping job \( j \) is infeasible, no other jobs that include \( j \) as a subset are feasible.

Proof: We prove this by contradiction. Consider a job \( j \) is infeasible but a job \( j \cup j' \) is feasible. If a job \( j \cup j' \) is feasible, removing one or more shipment from the job \( j \cup j' \) would not violate any constraints of TSPTW&CA problem (1.1)-(1.9). This contradicts our initial assumption. □

With the TSPTW&CA formulation and Proposition 1, we now explain the process of generating all feasible shipping jobs. We start with an example below and then summarize it as Algorithm 1.
Figure 3-1: An example showing the how feasible shipping jobs are generated for a crowdsourcee

In the example shown in Figure 3-1, “level” indicates the number of shipments present in a job. For example, all jobs in level 1 consist of only 1 shipment and so on. It should be noted that for level 1 the terms “shipment” and “job” are analogous i.e. one can interchangeably use both the terms for the items in level 1. The first level jobs are enumerated as described earlier in Section 3.3.1 (see Algorithm 2 for the implementation). Each job in the level can branch out several (n + 1)th level job by appending a shipment from level 1 that is not already present in the current job. If the addition of a new shipment does not make the job infeasible (recall the feasibility of the new job is checked by solving the TSPTW&CA problem), it is added to the list of feasible jobs. Else, the job is infeasible and by proposition 1, no further branching from the job is necessary. In the example above, this means that if the first highlighted job at level 2 is not feasible, none of the further branching jobs (highlighted in the example) should be enumerated.

The detailed algorithm is presented below.

Algorithm 1: JOB_ENUMERATOR

1. For each crowdsourcee i
2. \(J_i^1\) = Function FIRST_LEVEL_JOB (all shipment demands)
3. For \(w = 2\) to \(\min \{|J_i^1|, \max\_allow\}\)
4. \(k = 1\)
5. For \(j = 1\) to \(|J_i^{w-1}|\)
6. \(z=\text{position of } J_i^{w-1}(j, w-1)\)th shipment in \(J_i^1\)
7. For \(l = z+1\) to \(|J_i^1|\)
8. Function TSPTW&CA(\(J_i^{w-1}(j, 1:w-1) \cup J_i^1(l, 1)\)
9. If a solution exists

10. \[ J_i^w(k, 1:w) = J_i^{w-1}(j, 1:w-1) \cup J_i^1(l, 1) \]

11. \[ k = k + 1 \]

12. End If

13. End For

14. End For

15. End For

16. End For

The algorithm JOB_ENUMERATOR is executed for each crowdsourcer as indicated in line 1. This process can be done in parallel for each crowdsourcer. The first level jobs are enumerated in line 2 \((J_i^w)\) represents the set of \(w^{th}\) level jobs for crowdsourcer \(i\)), by the function FIRST_LEVEL_JOB. FIRST_LEVEL_JOB is described in Algorithm 2 later. The inputs to a function are denoted by the variables inside the parenthesis. In line 3, we define an index \(w\) that indicates the level of the job. The value for \(w\) starts from 2, as the first level jobs are already enumerated, and ends at the maximum number of shipments allowed in a shipping job. The minimum operator is required for cases where the number of first level jobs is less than the maximum number of shipments allowed. The index \(k\) in line 4 denotes the job number, which starts at 1 for each new level. Line 6 and 7 indicate that we sequentially select a new shipment from the first level that are not already included in the current job and add it to make a new shipping job. Line 8 checks the feasibility of the job after the addition of the new shipment by solving the TSPTW&CA problem. If a solution for the TSPTW&CA exists, then a new feasible job is formed (line 10) and the index \(k\) is increased by 1 (line 11). If no solution is found, then no new feasible job is formed and eventually no branching from the infeasible job will take place, complying with
the proposition 1. For each feasible job, the TSPTW&CA gives the route and schedule that the
crowdsourc ee should to follow, if assigned to complete that job. The cost of performing a job is
calculated as the total distance covered by a crowdsourc ee for the job divided by the speed of their
mode (walking, cycling or driving) and multiplied by their expressed VOT $\eta_i$.

The algorithm for enumerating the first level job is provided next, which is simply the
implementation of the discussion earlier in Section 3.3.1.

---

**Algorithm 2: FIRST_LEVEL_JOB (all shipment demands)**

1. For each crowdsourcee $i$
2. $J_i^1 = \emptyset$
3. For each shipment $u$
4. If $\text{NOR} \left( a_i + t_{du} > l_u, d_i < e_u \right)$
5. $J_i^1 = J_i^1 \cup \{u\}$
6. End If
7. End For
8. End For

---

**3.3.2 Shipping job assignment**

Once all feasible shipping jobs for each crowdsourcee are generated, the DSP needs to assign
jobs to crowdsourc ees such that: (i) no crowdsourc ees get more than one job, (ii) every shipment
is fulfilled, and (iii) the total cost to the DSP is minimized. The problem can be formulated as a set
packing problem. However, it may not always be guaranteed that every shipment demand can be
served by a crowdsourcee. In such cases, the DSP maintains backup vehicles that can promptly
fulfill such shipments. The lists of notations used for the formulation of the shipping job assignment are given as follows:

**Sets**

- \( I \)  
  Set of crowdsourcers

- \( P \)  
  Set of all shipment demands

- \( J_i \)  
  Set of shipping jobs feasible to crowsourcee \( i \) (generated as described in Section 3.3.1)

- \( J_u \)  
  Set of shipping jobs that include shipment demand \( u \)

**Parameters**

- \( C_{ij} \)  
  Cost of crowsourcee \( i \) to perform job \( j \) (calculated in section 3.3.1)

- \( B_u \)  
  Cost of fulfilling shipment demand \( u \) using the backup vehicle

**Decision Variables**

- \( X_{ij} \)  
  Binary variable taking value 1 if crowsourcee \( i \) is assigned to shipping job \( j \) and 0 otherwise

- \( y_u \)  
  Binary variable taking value 1 if shipment demand \( u \) is fulfilled using the backup vehicle and 0 otherwise

With these notations, the shipping job assignment problem can be formulated as the following Binary Integer Program (BIP):

\[
\begin{align*}
\text{min} \quad & \sum_{i \in I} \sum_{j \in J_i} C_{ij} X_{ij} + \sum_{p \in P} B_p y_p \\
\text{s.t.} \quad & \sum_{j \in J_i} X_{ij} \leq 1 \quad \forall \, i \in I
\end{align*}
\]
\[
\sum_{j \in J} \sum_{i \in I} X_{ij} + y_p = 1 \forall p \in P
\]  \hspace{1cm} (2.3)

\[X_{ij} \in \{0,1\} \forall i \in I, j \in J\]
\hspace{1cm} (2.4)

\[y_p \in \{0,1\} \forall p \in P\]
\hspace{1cm} (2.5)

The objective function (2.1) minimizes the cost of assigning jobs to crowdsources \((\sum_{i \in I} \sum_{j \in J} C_{ij} X_{ij})\) and the cost of assigning backup vehicles \((\sum_{p \in P} B_p y_p)\). Constraint (2.2) states that a crowdsourcer cannot be assigned more than one job and constraint (2.3) states that every shipment demands must be fulfilled exactly once either by a crowdsourcer or by a backup vehicle. Here we assume that a backup vehicle leaves from the DSP directly to the shipment location and returns back to the DSP. Completing more than one shipment in the trip of backup vehicles is not considered. This can be justified based on the following rationale. We assume that the crowdshipping market is thick and crowdsources are present in abundant number throughout the operation period. Given this assumption, there will be very less number of shipments that cannot be completed by one of the crowdsources. Such demands occur if they have very uncommon parameters, for example they are located very far from the DSP, or they need the service at very uncommon hours. As the occurrences of these cases are rare, having a backup vehicle for the direct delivery of such demands is justified. Constraint (2.4) and (2.5) specify the binary condition of decision variables.

The linear program (2.1-2.5) can be solved to optimality using the commercially available solvers such as CPLEX.
3.3.3 Payment Calculation

The parameter $C_{ij} \forall j \in J_i$ and $\forall i \in I$ in the job assignment problem depends upon the bidding information $\theta_i = (a_i, d_i, r_i, m_i)$ submitted by the crowdsorucee. More specifically, the set of feasible jobs for $i, J_i,$ is the function of $a_i$ and $d_i$, and $i$’s cost of performing job $j \in J_i$, $C_{ij}$, is the function of his/her stated VOT $r_i$ and the mode information $m_i$. We therefore denote the cost for crowdsorucee $i$ to complete the job $j$ as $C_{i,j} \in J_i(a_i,d_i)(r_i,m_i)$ or sometimes simply $C_{ij}(\theta_i)$. As shown in example 1, a crowdsorucee can manipulate his/her VOT $r_i$ to increase the payment derived. A crowdsorucee can also manipulate any of the other bid information besides $r_i$ to gain a higher payment from the DSP. Intuitively, a crowdsorucee can strategically submit false values of $a_i$ and $d_i$ to limit his/her competition only to crowdsorucees arriving with high VOT. To prevent such strategicbehavior, we introduce a payment rule that makes it the dominant strategy for every rational crowdsorucee to state their true bidding information. The payment rule is the generalization of the well-celebrated Vickrey-Clarke-Grove (VCG) pricing mechanism to the crowdshipping context where a crowdsorucee’s bid includes both monetary and non-monetary information.

To formally introduce the payment rule, let us consider true bids of all crowdsorucees are represented by $(\theta_i, \theta_{-i})$, where $\theta_i$ is the bid of crowdsorucee $i$ and $\theta_{-i}$s are the bids of all crowdsorucees other than $i$. Let $V(A^*)$ and $A^* = (X^*_{11} ... X^*_{ij} ... X^*_{ij}, y^*_1, ..., y^*_p)$ denote the optimal cost and the optimal assignment decision for the BIP (2.1-2.5). Let $V(A^*_{-i})$ represent the optimal cost of the BIP (2.1-2.5) when crowdsorucee $i$ is not present. $V(A^*_{-i})$ can be obtained by simply

6 This representation does not affect our formulations in earlier sections.
removing \( i \) from the set \( I \) and resolving the BIP (2.1-2.5). With these notations, we can now define the payment to each crowdsourcee and the utility they derive as,

**Definition 1:** The payment to a crowdsourcee is equal to his/her (stated) cost of performing the assigned job plus the difference in the total shipping cost to the DSP without and with his/her presence. It can be given as follows.

\[
p_i = \sum_{j \in J_{i}(a_i, d_i)} C_{ij}(r_i, m_i)X_{ij}^* + [V(A^*_{-i}) - V(A^*)] \quad \forall i \in I
\]  

where \( p_i \) represents the payment to crowdsourcee \( i \). The first term on the right-hand side (RHS) of Eq. (3) represents the crowdsourcee’s own cost of performing the assigned job (which is the function of his/her stated bid). It should be noted that when crowdsourcee \( i \) submits a different bid, it might change the optimal assignment \( A^* \), cost of the optimal assignment \( V(A^*) \), and \( i \)’s own cost of performing the assigned job. However, the term \( V(A^*_{-i}) \) remains unchanged for any bid submitted by crowdsourcee \( i \).

**Definition 2:** The utility derived by a crowdsourcee for doing the assigned job is the payment s/he receives from the DSP minus his/her own (true) cost of performing the assigned job. It can be given as:

\[
u_i(\theta_i) = p_i - \sum_{j \in J_{i}(a_i, d_i)} C_{ij}(r_i, m_i)X_{ij}^* \quad \forall i \in I
\]  

where \( u_i(\theta_i) \) represents the utility derived by the crowdsourcee. It is obvious that every rational crowdsourcee would like to increase the utility s/he derives from the DSP. In other words, s/he would like to increase the payment received from the DSP while reducing his/her own cost of doing the job. Recall, the cost of the crowdsourcee for completing the shipping job is the time spent in doing the assigned job multiplied by his/her VOT. This can also be described as the
opportunity cost of the crowdsourcee. It should be noted that the opportunity cost of the crowdsourcee does not depend on the VOT he submits and depends on his/her true perceived VOT.

Replacing the value of $p_i$ from (3.1) to (4.1) we get

$$u_i(\theta_i) = V(A_+^i) - V(A^*) \quad \forall i \in I$$

(4.2)

The utility of a crowdsourcee, under the proposed payment (3) can, therefore, also be defined as the decrease in the total shipping cost due to his/her presence. Recall $V(A_+^i)$ is a constant to crowdsourcee $i$. Since every crowdsourcee tries to maximize his/her utility, from (4.2) it can be seen that due to the payment rule (3), each crowdsourcee indeed try to maximize the negative of $V(A^*)$ which aligns with the objective of the DSP—minimizing the total shipping cost.

Next, we prove that under the proposed payment rule, it is the dominant strategy for every crowdsourcee to report their bid truthfully. In other words, a crowdsourcee cannot increase his/her utility by submitting a false bid. Recall, that a crowdsourcee needs to report a four-tuple information which includes the earliest and the latest time available to work, the VOT denoting the WTDC, and the mode used. Among others, a crowdsourcee cannot lie about his/her mode as the DSP can physically observe this information. For the other three, we prove that the payment rule described by the definition 1 can induce the truthful bidding behavior for every crowdsourcee.

**Proposition 2**: Reporting a longer than actual available time window i.e. $(\tilde{a}_i \leq a_i$ or $\tilde{d}_i \geq d_i)$ never makes a crowdsourcee better off.

**Proof**: Fix all other crowdsourcees’ bids $\theta_{-i}s$. Reporting a longer available time window for crowdsourcee $i$ generates all the jobs feasible while reporting the true time window plus the “extra jobs” that were infeasible while reporting true time window. By having the extra feasible jobs for
crowdsourcer, two things can happen. Either the assignment will not change leaving the utility unchanged for \( i \) or the assignment will change and the crowdsourcer will be assigned one of the “extra job.” But these extra job are actually infeasible to crowdsourcer \( i \). The DSP will in turn charge a heavy penalty for not being able to complete the assigned job, making his/her utility zero or even negative. Therefore, reporting \( \hat{a}_i \leq a_i \) or \( \hat{d}_i \geq d_i \) never makes a crowdsourcer better off. □

**Proposition 3:** Reporting a shorter than actual available time window i.e. \( (\hat{a}_i \geq a_i \text{ or } \hat{d}_i \leq d_i) \) never makes a crowdsourcer better off.

**Proof:** Fix all other crowdsourcers’ bids, \( \theta_{-i} \)s and assume \( i \) reports \( \theta'_i = (\hat{a}_i, \hat{d}_i, r_i, m_i) \) with a shorter available time window. Reporting a shorter than actual available time would generate the set of shipping jobs \( J'_i \) which is the subset of \( J_i \)— set of shipping jobs while reporting the true \( a_i \) and \( d_i \). Similar to the previous case, two outcomes are possible. First, the assignment does not change leaving the utility unchanged for crowdsourcer \( i \). Second, the assignment changes to \( A' = (X'_{11} \ldots X'_{ij} \ldots X'_{l1}, y'_1, \ldots y'_p) \) with the total cost of \( V(A') \). The total cost \( V(A') \) is always greater than or equal to \( V(A^*) \) because the new assignment is more restrictive given the fewer number of feasible jobs for crowdsourcer \( i \). Following the definition 2 for the utility and replacing the payment term following definition 1 we can write the following:

\[
\begin{align*}
    u_i(\theta'_i) &= \sum_{j \in J'_i(a_i, \hat{d}_i)} C_{ij}(r_i, m_i)X'_{ij} + [V(A^*) - V(A')] - \\
    &\sum_{j \in J'_i(a_i, \hat{d}_i)} C_{ij}(r_i, m_i)X'_{ij} \\
    u_i(\theta'_i) &= V(A^*) - V(A')
\end{align*}
\]  

(4.3)

(4.4)
From (4.4) it is clear that $i$’s utility is highest when $V(A')$ is minimum, which occurs when s/he reports the available time no shorter than the actual available time. This concludes that reporting a shorter than actual available time window i.e. ($\tilde{a}_i \geq a_i$ or $\tilde{a}_i \leq d_i$) never makes a crowdsourceree gain higher utility. □

**Proposition 4:** A crowdsourceree cannot improve his/her utility by reporting a bid with a false VOT.

**Proof:** Fix all other crowdsourcerees’ bids $\theta_{-i}$s. Let $\theta_i'' = (a_i, d_i, \tilde{r}_i, m_i)$ represent crowdsourceree $i$’s bid with the false VOT. If truthful reporting is not the dominant strategy for crowdsourceree $i$, then

$$u_i(\theta_i) > u_i(\theta_i'') \quad (5.1)$$

When $i$ reports $\theta_i''$, assume the assignment changes to $A'' = (X_{11}'' ... X_{ij}'' ... X_{IJ}'', y_1'', ... y_p'')$. From definition 2 the utility of the crowdsourceree can be given as $p_i - \sum_{j \in I_i(a_i,d_i)} C_{ij}(r_i, m_i)X_{ij}''$. Recall, although reporting an untrue VOT may change the overall assignment and the payment received, the cost incurred by the crowdsourceree will still be the function his/her true VOT. Eq. (5.1) may be rewritten as:

$$p_i - \sum_{j \in I_i(a_i,d_i)} C_{ij}(r_i, m_i)X_{ij}'' > V(A_{-i}^-) - V(A^*) \quad (5.2)$$

where the RHS is replaced from (4.2) when $i$ reports the true bid. Replacing the payment term following the definition 1 we get:

$$\sum_{j \in I_i(a_i,d_i)} C_{ij}(\tilde{r}_i, m_i)X_{ij}'' + [V(A_{-i}^-) - V(A'^*)] - \sum_{j \in I_i(a_i,d_i)} C_{ij}(r_i, m_i)X_{ij}'' > V(A_{-i}^-) - V(A^*) \quad (5.3)$$
$V(A^*_i)$ is the common term in both LHS and RHS and cancels out. Multiplying both sides by -1 and further simplifying $V(A'') - \sum_{j \in J_i(a_i,d_i)} C_{ij}(\bar{r}_i, m_i)X''_{ij}$ we get:

$$[\sum_{k \neq i} \sum_{j \in J_k(a_k,d_k)} C_{kj}(r_k, m_k)X''_{kj} + \sum_{p \in P} B_p y'''_p] + \sum_{j \in J_i(a_i,d_i)} C_{ij}(r_i, m_i)X''_{ij} < V(A^*)$$

(5.4)

Note that all cost parameters, $C_{ij}$s in the left-hand side (LHS) are the function of true bids ($\theta_i, \theta_{-i}$)s. The parameters $B_p$s are constants to the DSP. Therefore, Eq. (5.4) is a contradiction because $V(A^*)$ represents the minimum cost when every crowdsourcer express his/her true $\theta_i$ and no other assignment can be strictly lower than $V(A^*)$. This concludes the proof that no higher utility is possible by reporting the false VOT in the bid. □

With propositions 2, 3 and 4, we conclude that a crowdsourcer cannot increase his/her utility by misreporting the private information. This can be applied to any crowdsourcer thus making truthful reporting the dominant strategy for each rational crowdsourcer irrespective of what others report. A rational crowdsourcer will not be willing to take part in the assignment process if the dominant strategy results in negative utility. We show in the following proposition that this is not the case with the proposed payment scheme.

**Proposition 5:** All crowdsourcers are willing to participate in the shipping job as they get the non-negative utility.

**Proof:** While reporting the true bid, the utility of a crowdsourcer is given by Eq. (4.2). $V(A^*_i)$ can never be smaller than $V(A^*)$, because for everything else being constant, having a lesser number of crowdsourcers cannot decrease the total cost of shipping. In a case where $i$ is not assigned, $V(A^*_i)$ and $V(A^*)$ are equal giving the utility of 0. Therefore $u_i(\theta_i) \geq 0$ for every $i$. □
With the payment calculation, we now shift our attention towards the total shipping cost that is borne by the DSP. The total cost is the payment made to crowdsourcers in addition to the cost of shipments fulfilled by backup vehicles. This can be given as,

\[ TC = V(A^*) - \sum_{i \in I} \sum_{j \in J} C_{ij}(r_i, m_i) X^*_ij + \sum_{i \in I} p_i \]  

where \( TC \) represents the total cost for the DSP. \( V(A^*) - \sum_{i \in I} \sum_{j \in J} C_{ij}(r_i, m_i) X^*_ij \) represents the shipping cost using the backup vehicle in the optimal solution and \( \sum_{i \in I} p_i \) represents the sum of payments to each crowdsourcer. Replacing the value of \( p_i \) from Eq. (3) and simplifying we get:

\[ TC = V(A^*) + \sum_{i \in I} [V(A^* - i) - V(A^*)] \]  

The term inside the square bracket on the RHS of Eq. (6.2) is the summation of non-negative numbers (refer to proposition 5) and thus a non-negative number. This value represents the additional cost borne by the DSP to induce the truthful bidding behavior among crowdsourcers, and henceforth referred to as the cost of truthfulness. Two natural questions that have not yet been answered are (i) how large can the cost of truthfulness be, and (ii) if the proposed IC mechanism generates savings in the shipping cost to the DSP.

For cases where number of crowdsourcers \(|I|\) is sufficiently large and their bids are generated from a compact set, \( V(A^* - i) \) is very close to \( V(A^*) \) (De Vries and Vohra, 2003; Monderer and Tennenholtz, 2005). The cost of truthfulness therefore approaches to zero as the number of crowdsourcers become large and the DSP can achieve the minimum shipping cost by implementing the proposed mechanism. We will numerically investigate as to how large the pool of crowdsourcers should be for the cost of truthfulness to asymptotically approach to 0 in the later section.
On the other hand, when the number of crowdsourcées is not sufficiently large, the cost of truthfulness may form a significant portion of the total cost. However, we speculate that with this additional cost, the DSP can in fact realize the reduction in the total shipping cost. This is because in absence of IC mechanisms, all crowdsourcées are motivated to strategically misreport their VOT thereby increasing the total shipping cost by even higher amount. To assess the saving from the proposed mechanism we first need to derive the optimal bidding strategy when no IC mechanisms are present. It would be very difficult to derive the optimal bidding strategy when combinations of shipments are considered for the assignment. We, therefore, derive the optimal bidding strategy for the simple case with single shipment and analyze its relationship with the number of crowdsourcées present. The analysis should shed some intuition on the complex-bidding scenario considered in the study as well.

3.3.4 Optimal bidding strategy in absence of incentive compatible mechanism

In this section, we derive the optimal bidding strategy for crowdsourcées in absence of any IC mechanism. We consider a simplified scenario where there are \( n \) crowdsourcées (with same mode) and 1 shipment demand. The crowdsourcées only need to submit their VOT ($/hr) as their bid. The DSP then selects the crowdsourcée with the lowest VOT to complete the shipment job. For simplicity, let us consider that it takes exactly 1 hour for crowdsourcées to complete the shipment. The payment to the assigned crowdsourcée will then be equal to the VOT submitted in his/her bid. With this setup, we model the bidding behavior for each crowdsourcée. The assumptions made for developing the model are outlined below.

1. The total number of crowdsourcées present \( n \), is a common knowledge to all crowdsourcées.
2. The VOT of each crowdsourcer is a private information and is only known to himself. However, the distribution from which the exact VOT is drawn is the common knowledge for all (i.e. a crowdsourcer know the cumulative distribution function (CDF) $F_X$ and the probability density function (PDF) $f_x$ of VOT among the crowdsourcers).

3. Every crowdsourcer’s VOT follows an independent and identical distribution (i.i.d.).

4. Every crowdsourcer is rational and aim to maximize his/her expected utility.

5. The true and stated VOTs for every crowdsourcer are non-negative.

To maximize the utility in presence of the limited knowledge about rivals, crowdsourcers need to consider two things. First, their submitted VOT must be low enough to be assigned to the shipment, and second, their submitted VOT should be high enough to maximize their utility since the utility of crowdsourcer $i$ would now be $(\hat{r}_i - r_i)$ if selected and 0 otherwise. The probability of crowdsourcer $i$ being selected can be given as follows:

$$ P(i \text{ is selected}) = P(\hat{r}_i \text{ lower than } n - 1 \text{ other crowdsourcers}) \quad (7.1) $$

Let $X$ be the random variable denoting the stated VOTs of rival crowdsourcers. The probability of $i$ being selected would therefore be

$$ P(i \text{ is selected}) = P(X > \hat{r}_i)^{n-1} = [1 - F_X(\hat{r}_i)]^{n-1} \quad (7.2) $$

The superscript $n - 1$ in Eq. (7.2) appears due the i.i.d. condition of the rival crowdsourcer’s bid. The expected payoff of $i$, $E[\pi_i]$, can therefore be given as:

$$ E[\pi_i] = [1 - F_X(\hat{r}_i)]^{n-1}(\hat{r}_i - r_i) + 0 \ast F_X(\hat{r}_i)^{n-1} \quad (7.3) $$

In Eq. (7.3), $r_i$ is a constant value (i’s true VOT) and $\hat{r}_i$ is the stated VOT which $i$ can manipulate to increase his/her expected utility. One can therefore, differentiate Eq. (7.3) with respect to $\hat{r}_i$ and equate with 0 to get the optimal bidding value.
\[
\frac{\partial E[\pi_i]}{\partial \hat{r}_i} = [1 - F_X(\hat{r}_i)]^{n-1} - \hat{r}_i(n - 1)[1 - F_X(\hat{r}_i)]^{n-2} f_X(\hat{r}_i) + r_i(n - 1)[1 - F_X(\hat{r}_i)]^{n-2} f_X(\hat{r}_i)
\]

(7.4)

\[
[1 - F_X(\hat{r}_i)]^{n-1} - (n - 1)[1 - F_X(\hat{r}_i)]^{n-2} f_X(\hat{r}_i)(\hat{r}_i - r_i) = 0
\]

(7.5)

\[
\hat{r}_i = r_i + \frac{1}{(n-1)} \frac{[1 - F_X(\hat{r}_i)]}{f_X(\hat{r}_i)}
\]

(7.6)

Eq. (7.6) has two implications. First, since \(\frac{1}{(n-1)} \frac{[1 - F_X(\hat{r}_i)]}{f_X(\hat{r}_i)}\) is always positive (\(F_X(\hat{r}_i)\) can take the maximum value of 1, \((n - 1)\) and \(f_X(\hat{r}_i)\) cannot be negative), the optimal bidding strategy for crowdsourcers is always to overbid their VOT in the absence of IC mechanisms. Second, we can analyze the direction of movement of \(\hat{r}_i\) in response to the change in \(n\). When \(n\) decreases, keeping \(\hat{r}_i\) fixed, the term \(\frac{1}{(n-1)} \frac{[1 - F_X(\hat{r}_i)]}{f_X(\hat{r}_i)}\) becomes larger. This will lead to LHS<RHS in the Eq. (7.6). If one increases the \(\hat{r}_i\), the numerator of the second term in the RHS, \([1 - F_X(\hat{r}_i)]\), decreases. The direction of denominator \(f_X(\hat{r}_i)\) however is indeterminate for the increase in \(\hat{r}_i\). If we can make assumption about the direction of \(\frac{[1 - F_X(\hat{r}_i)]}{f_X(\hat{r}_i)}\) then we can make conclusion about the response to \(\hat{r}_i\) when \(n\) varies. The term \(\frac{[1 - F_X(\hat{r}_i)]}{f_X(\hat{r}_i)}\) represents the inverse of the hazard rate and for distributions with monotonic increasing hazard rate (distribution like normal and uniform have this property), we can conclude that for decrease in \(n\), \(\hat{r}_i\) is increasing. This translates that the optimal bidding strategy for a crowdsourcer is to increase his/her VOT as the number of competing crowdsourcers decreases. Eq. (7.6) is the optimal bidding strategy for every crowdsourcer.

On the other hand, the IC mechanisms incentivize every crowdsourcer to express their true VOT. For the current simplified scenario, the VCG mechanism we introduced will simply collapse to the second price auction, meaning that the winning crowdsourcer will be paid equal to the
second lowest VOT. The cost of truthfulness is, therefore, the difference between the second lowest and the lowest VOT among the crowdsourcées present. The cost saving realized from the introduction of IC mechanism is the difference between the lowest stated VOT $\hat{r}_i$ (absent the mechanism) and the second price payment (when the mechanism is present). The use of the mechanism is justifiable when the saving in cost is greater than 0. In the Appendix 3A, we present the numerical analysis to demonstrate the saving of the mechanism for the simplified case with single shipment demand.

The analysis shows that for the simplified condition, motivation to misreport the higher VOT increases as the pool of competing crowdsourcées decreases. We hypothesize that this motivation is even higher when the number of shipments increases and the complexities of combinatorial bidding, time window requirement of shipments, time availability of crowdsourcées and different modes are introduced further adding the importance of having the mechanism. This observation will be helpful to numerically analyze the cost saving from the proposed mechanism in the later section.
3.4 Dynamic Case

In the context of on-demand delivery, shipment demands arrive over time. With limited available time, crowdsourcers also dynamically enter and exit the crowdshipping system. In this case, the static mechanism no longer ensures truthful reporting of $a_i$ and $r_i$ from crowdsourcer $i$, as shown in example 2. In the literature of dynamic mechanism design, two approaches exist. The first approach assumes that the auctioneer has prior probabilistic knowledge about future agent arrivals and accordingly performs resource allocation and computes payment using dynamic programming (Parkes and Singh, 2004). In contrast, the second approach designs mechanisms to optimize allocation periodically using existing information (Zou et al., 2015; Chen et al., 2015;

Example 2: We consider two crowdsourcers and two shipments shown in the table below. A backup vehicle is available with the same travel time as the two crowdsourcers but at higher cost of $\$15$/period. Consider the DSP performs shipment assignment at the end of each period. Both crowdsourcers are present in period 1 along with shipment 1. Because $r_1 < r_2$, at the end of period 1 the DSP assigns crowdsourcer 1 to shipment 1. The payment, based on Eq. (3), will be 7.5. Then in period 2, shipment 2 arrives and only crowdsourcer 2 is available. The DSP assigns crowdsourcer 2 to shipment 2. Again using Eq. (3), the payment to crowdsourcer 2 will be 15. However, crowdsourcer 1 could have increased personal utility by misreporting $a_1$ or $r_1$. For example, if crowdsourcer 1 stated that $\hat{a}_1 = 2$, crowdsourcer 2 would be assigned to shipment 1 at the end of period 1. Crowdsourcer 1 would be assigned to shipment 2 in the next period with a greater payment ($\$15$). Similarly, s/he can also state $r_1 = 11$, keeping everything else constant, to be assigned at time period 2 with payment of 15.

<table>
<thead>
<tr>
<th>Crowdsourcer</th>
<th>$a_i$</th>
<th>$d_i$</th>
<th>$r_i$</th>
<th>$m_i$</th>
<th>Shipment</th>
<th>$e_u$</th>
<th>$l_u$</th>
<th>Travel time from the DSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>7</td>
<td>5</td>
<td>driving</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>0.75</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>7</td>
<td>10</td>
<td>driving</td>
<td>2</td>
<td>2</td>
<td>8</td>
<td>1</td>
</tr>
</tbody>
</table>
Robu et al., 2013). Given that prior probabilistic knowledge about future agent arrivals is often difficult to obtain, we take the second approach in this study.

Specifically, the following dynamic mechanism is considered, in the context of time-guaranteed delivery. A customer is promised to receive the ordered shipment within time $G$ after the order is placed, which is assumed the same as the time when the corresponding shipment demand is generated.\(^7\) In the mechanism, the DSP solicits crowdsourcers and assigns shipments once every $T$ periods (Figure 3-2). Each time (say $t_0 - T$), the generation of feasible shipping jobs\(^8\), their assignment to crowdsourcers, and the payment calculation follow the static case. The key difference from the static mechanism in Sec. 3.1 is that a crowdsourcer $i$ after completing the assigned job is required to return to the DSP by $t_0$. In fact, the return requirement is not foreign to crowdshipping and has been reported in practice (Botsman, 2014; Amazon Flex, 2016; Campbell, 2016). The returning crowdsourcers will automatically be considered for the next assignment period without further solicitation of their bids as long as they are still available (i.e., if $d_i > t_0$). The return to DSP requirement for all crowdsourcers can be implemented by specifying the DSP location as the returning depot in the TSPTW&CA problem (1.1-1.9) and replacing the constraint (1.7) as $T_r \leq \min(d_i, t_0)$ which states that the crowdsourcer must be able to return to the DSP before the next assignment period or the end of his/her available time whichever is earlier. A job, which can be completed by a crowdsourcer in time but does not leave enough time for the

\(^{7}\) Alternatively, a constant could be added to account for the time needed between order placement and generation of the shipment demand. This will not alter the general idea and insights presented here.

\(^{8}\) The shipping job generation will be more efficient than the static case. In the dynamic case, for all arriving crowdsourcers with the same mode and the end of available time greater than the next assignment period ($\forall i: d_i > t_0$), the feasible shipping job generation process (Algorithm 1 and 2) can be solved only once. The shipping cost for each crowdsourcer can then be calculated using their individual VOT. For crowdsourcers with the end of available time lesser than the next assignment period, feasible shipment job generation should be done individually.
crowdsourceree to return to the DSP location before the next assignment will not be counted as a feasible job. Requirement to serve shipments within $G$ periods of their arrival can be implemented by defining $l_u = e_u + G$ for every shipment.

![Diagram](image)

Figure 3-2: Illustration of the periodic crowdsourceree solicitation and shipment assignment under the dynamic mechanism

**Proposition 6:** The dynamic mechanism which assigns shipping jobs to crowdsourcerees once at every $T$ period’s interval with the requirement that every crowdsourceree must be able to return to the DSP no later than the next assignment period is able to induce truthful reporting for all attributes of crowdsourcerees’ bid.

**Proof:** We first prove about the truthfulness for the $a_i$. A crowdsourceree cannot gain by expressing $\hat{a}_i < a_i$ with the argument similar to that in Proposition 2. For $\hat{a}_i > a_i$, suppose that $t_0 - T < a_i < t_0$. If $\hat{a}_i < t_0$, then the shipment assignment and payment made at $t_0$ will be exactly the same as if crowdsourceree $i$ truthfully reports $a_i$. If $t_0 < \hat{a}_i < t_0 + T$, then crowdsourceree $i$ will lose the chance to be assigned at $t_0$ although the assignment at $t_0 + T$ will remain unchanged. The same applies if crowdsourceree $i$ misreports a later $\hat{a}_i$. Therefore, it is always the best strategy to report
the true \(a_i\) for any crowdsourcees. The reasoning of Propositions 2 and 3 will similarly apply here for the truthfulness of \(d_i\).

With the requirement that every crowdsourcee should be able to return to the DSP no later than the start of next assignment period, the mechanism essentially eliminates the strategy for misreporting the arrival time. With this, the dynamic mechanism is indeed the static mechanism performed at each period. Note, that the dynamic mechanism allows crowdsourcees to submit their bid only once and they will be automatically reconsidered for the future assignments as long as they are still available. With proposition 4, no crowdsourcees will be better off at any periods by misreporting their bids. This concludes that the proposed dynamic mechanism is able to induce truthful reporting of crowdsourcees’ bid. □

With the proposition 6, we now look at the requirement for the length of \(T\). The length of \(T\) should be set to ensure time-guaranteed delivery in the worst case scenario: a shipment demand generated right after an assignment (so it has to wait for \(T\) till the next assignment) and needs to be delivered to a location that is farthest from the DSP location. At the next assignment, the DSP will use the fastest mode (e.g., a driving crowdsourcee or the DSP’s backup vehicle) to go directly to the shipment destination, so that the associated trip time \(t_w\) plus the waiting time \(T\) is less than \(G\): \(T + t_w \leq G\). Therefore, for a given \(G\) and \(t_w\), the maximum length of \(T\), denoted by \(T_{\text{max}}\), can be equal to \(G - t_w\) (the implicit assumption is that \(G \gg t_w\)). For the minimum length of \(T\), denoted by \(T_{\text{min}}\), it should be large enough to allow for the farthest shipment demand from the DSP location to be delivered, and the crowdsourcee after completing the delivery to return to the DSP location before the next assignment. In other words, the fastest mode should be able to make a roundtrip single shipment delivery to the farthest location from the DSP location within \(T_{\text{min}}\) (i.e., \(T_{\text{min}} \geq\)
Thus the minimum and maximum value for $T$ depends on the size and shape of the service area and the DSP location. As an example, for a squared service area with side length $L$ and the DSP located at the center, the farthest locations from the DSP are the four corners. Thus $T_{\text{min}} \geq \sqrt{2}L/v_f$, where $v_f$ is the speed of the quickest crowdsourcer mode (or of the DSP’s backup vehicle).

There is a tradeoff in the choice of $T \in [T_{\text{min}}, T_{\text{max}}]$: having a smaller $T$ reduces crowdsourcers’ time wasted in inter-assignment waiting and consequently increases crowdsourcers’ time availability for assignments. However, a smaller $T$ constrains the possibility to form larger shipping jobs. More jobs will be formed with fewer shipment demands, which increases the total shipping cost due to the reduced consolidation. We will numerically investigate how the choice of $T$, size of the service area and the location of the DSP affect the performance of the dynamic mechanism.

3.5 Numerical analysis

The section numerically investigates the performance of the static and dynamic mechanisms. The mechanisms are implemented in a Matlab 2016a environment on an Intel Core i7 3630 2.4 GHz machine with 8 GB RAM. The integer programs involved in the mechanisms are solved via ILOG CPLEX v12.6.

\footnote{The $T_{\text{min}} \geq 2t_w$ is not the absolute requirement as the mechanism can be truthful for value of $T$ lesser than $2t_w$. When $T < 2t_w$, the shipments requiring $t_w$ travel time can only be assigned to the backup vehicle which is not very desirable. Having $T \geq 2t_w$ will reduce occurrence of such instances.}
3.5.1 Static Case

In the static case, we consider a 5-mile-by-5-mile service area with the DSP located at the center. A suite of problem instances is tested with varying shipment demands and crowdsourcers. Three shipment demands are considered: 50, 75, and 100. For a given demand, we vary the number of crowdsourcers such that the ratio of crowdsourcers to shipments equals 0.5, 1.0, and 1.5. In total, nine scenarios are examined.

3.5.1.1 Setting up the analysis context

Shipment demands are randomly drawn from the service area. For a given demand, the beginning of its delivery window, which is assumed 2 hours, draws from a uniform distribution $[0, 600]$ (in min). The early available time of each crowdsourcer is also drawn from the same uniform distribution. The length of a crowdsourcer’s available time window is assumed uniformly distributed between 90 min and 150 min. The mode of a crowdsourcer is randomly picked from walking, biking, and driving. The speed of the modes is 3 mph (walking), 10 mph (biking), and 20 mph (driving). The speed of backup vehicles is also assumed 20 mph. The maximum number of shipments allowed in a shipping job is three.

The WTDC of a crowdsourcer depends on his/her mode. For walking and biking crowdsourcers, we assume that their WTDCs follow the truncated gamma distribution with shape and scale factors equal to 14 and 0.5, so that the mean and variance is equal to $7/hr$ and 3.5. A mean WTDC of $7/hr$ is comparable to the minimum wage rate in the US (about $7.25/hr$ (DOL, 2016)). The gamma distribution is truncated at two standard deviation above and below the mean because to low or too high WTDCs are unlikely among crowdsourcers. For driving crowdsourcers the shape and scale factors are 28 and 0.5 which give the mean and variance of $14/hr$ and 7. The
distribution is also truncated at two standard deviation above and below the mean. The mean WTDC for driving crowdsourcers is higher because a driving crowdsourcee must also account for fuel, car wear-and-tear, and insurance costs. Given that a driving crowdsourcee travels at a speed of 20 mph, the cost of using a driving crowdsourcee is $0.7/mile which is 30% lower than the cost of using a backup vehicle (assumed $1/mile). Because shipment demand, crowdsourcers, and crowdsourcee WTDCs are drawn from distributions, each of the nine scenarios is simulated five times. The results reported below are the averaged values.

### 3.5.1.2 Overall system performance

Table 3-1 presents the DSP total shipping cost with the static mechanism. For comparison, the optimal total shipping cost with full information (thus truth telling is not relevant) is also reported. The ratio in the last column of Table 3-1 show the extra cost needed to induce truthful information reporting from crowdsourcees. The extra cost decreases with the number of crowdsourcees per shipment demand. When the crowdsourcee-shipment demand ratio is 0.5, the extra cost accounts for 60-70% of the optimal total shipping cost. When the crowdsourcee-shipment demand ratio is 1.5, the extra cost will only be 12-15% of the optimal total.

<table>
<thead>
<tr>
<th># of shipment demands (1)</th>
<th># of crowdsourcees (2)</th>
<th>Total shipping cost with the static mechanism (3)</th>
<th>Optimal total shipping cost with full information $V(A^*)$ (4)</th>
<th>Ratio (3)/(4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>25</td>
<td>76</td>
<td>46</td>
<td>1.6</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>38</td>
<td>30</td>
<td>1.3</td>
</tr>
<tr>
<td>75</td>
<td>38</td>
<td>31</td>
<td>27</td>
<td>1.1</td>
</tr>
<tr>
<td>75</td>
<td>75</td>
<td>75</td>
<td>46</td>
<td>1.6</td>
</tr>
<tr>
<td>75</td>
<td>48</td>
<td>48</td>
<td>40</td>
<td>1.2</td>
</tr>
</tbody>
</table>
The static mechanism can be implemented in relatively short time (Table 3-2). The CPU times for feasible shipping job generation and shipping job assignment are within 1 min for eight out of nine scenarios. Payment calculation takes longer time. In addition, the CPU times increases more than proportionately with shipment demand but less than proportionately with the number of crowdsourcees. Since payment calculation can be done for each crowsourcee independently, the computation time can be reduced using parallel computing. Heuristics could be considered instead of the exact solution approach to further improve computational efficiency. It should be noted that heuristically computing payment will not alter the IC property of the mechanism, but may increase the cost of truthfulness (Nisan and Ronen, 2007; Zhou and Saigal, 2014).

Table 3-2: Computation time comparison for different processes

<table>
<thead>
<tr>
<th># of shipment demands</th>
<th># of crowdsourcees</th>
<th>CPU time (sec)</th>
<th>Generating feasible shipping jobs</th>
<th>Shipping job Assignment</th>
<th>Payment calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>25</td>
<td>1.2</td>
<td>0.4</td>
<td>6.6</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>2.6</td>
<td>0.7</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>75</td>
<td>3.6</td>
<td>0.6</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>38</td>
<td>6.6</td>
<td>3.2</td>
<td>85</td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>75</td>
<td>15</td>
<td>5.1</td>
<td>136</td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>113</td>
<td>22</td>
<td>5.5</td>
<td>142</td>
<td></td>
</tr>
</tbody>
</table>
### 3.5.1.3 Comparison with the traditional auction approach

A major objective of the numerical analysis is to understand to what extent the static mechanism will outperform the traditional auction and fixed rate approaches in terms of total shipping cost to the DSP. Let us first look at the comparison with traditional auction.

Recall in Section 3.3.4 that, absent the mechanism, a crowdsourcee have the incentive to misreport greater than his/her true WTDC. Since it is very difficult to know the exact amount of misreporting, we consider four different percentage values: 15%, 20%, 25%, and 30% by which a crowdsourcee overstates his/her true WTDC. Table 3-3 shows the percentage total cost savings to the DPS with the mechanism compared to the corresponding WTDC overstatement cases. We find that when the overstatement is low (15% or 20%) and there is a lack of crowdsourcees (crowdsourcee-shipment demand ratio at 0.5), the percentage cost saving will be negative, meaning that the extra cost to induce truthful information reporting is so large that it exceeds the cost increase due to overstated WDTCs absent the mechanism. However, in all other cases the static mechanism yields a lower total DSP shipping cost.

Table 3-3: Saving in shipping cost compared to the traditional auction

<table>
<thead>
<tr>
<th># of shipment demands</th>
<th># of crowdsourcees</th>
<th>Percentage saving when VOT is overstated (by different %)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>15%</td>
</tr>
<tr>
<td>50</td>
<td>25</td>
<td>-25%</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>1%</td>
</tr>
<tr>
<td></td>
<td>75</td>
<td>11%</td>
</tr>
<tr>
<td>-------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>38</td>
<td></td>
<td>-22%</td>
</tr>
<tr>
<td>75</td>
<td>75</td>
<td>5%</td>
</tr>
<tr>
<td>113</td>
<td></td>
<td>8%</td>
</tr>
<tr>
<td>50</td>
<td></td>
<td>-27%</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>7%</td>
</tr>
<tr>
<td>150</td>
<td></td>
<td>8%</td>
</tr>
</tbody>
</table>

Note: negative percent savings mean cost increase.

We further compare the total DSP shipping costs with the static mechanism and with an ideal auction where crowdsourcers report private information truthfully. This relates back to the earlier discussions on the asymptotics of the extra cost to induce truthful reporting (Section 3.3.3). Here we plot the above two shipping costs as a function of the number of crowdsourcers and 50 shipments. Figure 3-3 confirms the asymptotic property, showing that the difference between the two cost curves approaches zero as the number of crowdsourcers increases. When an abundant pool of crowdsourcers is present, the extra cost to induce truthful reporting will become almost zero.
3.5.1.4 Comparison with the fixed rate approach

In this subsection, we investigate the saving of the proposed mechanism compared to the fixed rate approach. When the fixed rate payment is provided, crowdsourcers with the VOT higher than the fixed rate do not take part in the crowdshipping. When the fixed rate is too low, only a few crowdsourcers will be interested in doing the crowdshipping which warrants the greater use of backup vehicles and increase the total cost to the DSP. On the contrary, if the rate is too high every crowdsourcer will be paid a higher value which also increases the DSP’s total shipping cost. As the fixed rate goes on increasing, the total cost curve follows a downward convex curve as shown in Figure 3-4.
Figure 3-4: Total shipping cost for different values of fixed rate

The horizontal axis in Figure 3-4 represents the fixed rates at which crowdsourcers are paid and the vertical axis represents the total shipping cost to the DSP. Three curves are plotted for different combination of shipment demands and crowdsourcers present (S and C denote respectively the numbers of shipments and crowdsourcers). The minimum shipping cost occurs at around $16/hr to $18/hr for different combination of shipments and crowdsourcers present. This minimum cost with the fixed rate is on average 21% higher than the cost with the proposed mechanism (see the total cost for proposed mechanism in Table 2-1).

3.5.1.5 Comparison with truck-only delivery

We compare the cost of crowdshipping with the cost while using the traditional truck delivery. For the truck delivery, we solve the vehicle routing problem with time windows (VRPTW) using the Nearest Neighborhood heuristics introduced in Solomon (1987). The per mile operating cost of the trucks is same as the backup vehicle but the trucks can now deliver more than a single shipment. We do not impose any capacity constraints for the truck, as we assume the sizes of parcels are small and all of them can be accommodated in a single truck. More than one truck
however may be required due to the time constraints of the shipments. Results are shown in Figure 3-5. The three clusters represent the cases with three different number of shipment demands. Inside each cluster, the X-axis represents the number of crowdsourcers present. The truck delivery cost is constant in each cluster as this cost would be invariant to the number of crowdsourcers present and only depends upon the number of shipment demands. The results show that truck delivery costs is significantly higher than the cost with the proposed crowdshipping model further signifying the usefulness of the crowdshipping for the urban parcel delivery.

![Graph showing comparison of crowdshipping cost with traditional truck delivery cost](image)

**Figure 3-5:** Comparison of the crowdshipping cost with the traditional truck delivery cost

Now we shift our attention to the running time of the solution. The CPU times for generating the feasible shipping jobs, solving the job assignment problem, and the payment calculation are
presented for problems with different combination of crowdsourcers and shipment demands. The results are average of 5 instances of each problem Table 3-2.

3.5.1.6 Cost sensitivity to the maximum number of shipments allowed in a job

Finally, we test the sensitivity of the total cost with the maximum number of shipments allowed in a shipping job. Allowing the higher number of shipments in a shipping job decreases the total cost due to the economies of consolidation but increases the solution time significantly. The results are shown in Figure 3-6. The X-axis represents the solution time required for solving a problem instance with 50 shipments and 25 crowdsourcers. The Y-axis represents the total shipping cost. Each dot represents the total cost and solution time for different number of shipments allowed in a job (the value in the balloon callout represents the maximum number of shipments allowed in a job). Figure 3-6 shows that the reduction in the total cost is very marginal as the maximum number of shipments allowed increases from 3 to 4 and 4 to 5 but the increase in solution time is very pronounced. This suggests allowing the maximum of 3 or 4 shipments gives a good tradeoff between the total cost and the solution time.

![Figure 3-6](image-url)
3.5.2 Dynamic Case

3.5.2.1 Setting up the analysis context

With the results for the static case, we now move towards the dynamic case where we intend to apply the mechanism to the real-world scenario. We consider a neighborhood called Near North Side of the Chicago downtown. The area of the neighborhood is 2.78 sq. miles with a very high population density of about 30,000 people per square mile (Statistical Atlas, 2016). The information on the number of shipments is derived with the help of data available on SimplyMap\textsuperscript{10}. SimplyMap provides information on the number of online shipment demands (per year) for retail stores like Walmart at the level of the census tract. Figure 3-7 shows the location of the neighborhood considered for the study, locations of shipment demands and the DSP, and the road network.

\textsuperscript{10} Available at \url{http://www.simplymap.com/index.html}. 

Figure 3-6: Sensitivity of total cost with the maximum number of shipments allowed in a job
The exact locations of shipment demands (shown in Figure 3-7) are generated at random with the spatial distribution proportional to that derived from SimplyMap. The shipment demands are assumed to arrive with a Poisson’s distribution with a mean of 20 arrivals per hour. The guaranteed service hour, $G$ is assumed to be 2 hours after the arrival time. The DSP is assumed to be the Walmart Store present in the neighborhood for the base case. The numbers of crowdsourcers are also assumed to arrive at the DSP with the Poisson’s arrival rate of 20 per hour. Other parameters are similar to those introduced in the static case.

The distance between any two points (to and from the DSP and between any two shipments) is calculated using the Open Street Map’s Application Program Interface (OSRM, 2016). The calculation involved finding the shortest path between the two points using the real road network. The trip time to the farthest point ($t_w$) is the time required to cover the routing distance from the DSP to the farthest point within the neighborhood. This is approximated by generating number of points at a regular interval along the borderline of the neighborhood and finding the longest routing distance from the DSP to those points including the existing shipment locations. The longest distance is then padded by 20% to account for any locations that might have been missed. The time required for reaching the farthest point ($t_w$) using the fastest vehicle travelling at 20 mph is found to be 9 minutes.

The operational period is assumed to be 10 hours in a day. For the base case, we assume that the length of $T$ is set to the maximum possible ($T_{max} = G - t_w = 111$ minutes) giving 6 assignment periods (the last period will have less than 111 minutes).
3.5.2.2 Overall system performance

Figure 3-8 below presents the number of shipment demands, active crowdsourcers, and the cumulative cost while using crowdsourcers and truck at each period. The total cost while using the crowdshipping is about $118 while the total cost with truck is more than $200 for the full operation period. The total cost will be sensitive to the length of the periods, which we investigate in Figure 3-9. The total cost shown in Figure 3-9 is the cumulative cost at the end of the operational period but with different length of periods ranging from $T_{min}$ to $T_{max}$ at the interval of 15 minutes. Results show that the total cost decreases initially when the length of period is increased from 18 minutes to 33 minutes. This is because when the length is too small crowdsourcers cannot consolidate larger number of shipments in a job as they need to return to the DSP before the next assignment period. As the length of the period increases, the total cost initially decreases due to increased consolidation opportunity. After 33 minutes, the total cost does not decrease any more. This is due to the following two reasons. First, further consolidation is not possible as we restrict the maximum number of shipments allowed in a job to be 3. Second, having a longer period translates to crowdsourcers not being utilized efficiently due to the time wasted in waiting between the inter-assignment periods. On contrary, a smaller length of periods mean a crowdsourcer can be re-assigned frequently thereby increasing the number of active crowdsourcers at each period. The analysis shows that for the considered area, the total cost will be lowest if the length of the period is set to about half an hour.
3.5.2.3 Cost sensitivity to delivery time guarantee and service area size

In this subsection, we analyze the sensitivity of the total cost with respect to the length of the guaranteed service time \( G \) and the size of the service area. As the size of the service area increases, the time required to travel to the farthest point \( t_w \) also increases. We consider two
hypothetical areas with the size two times and four times the base area. Different values of $G$ from 60 minutes to 240 minutes are investigated. The value of $T$ is set to $T_{max}$ at each case. The results are shown in Figure 3-10. For the area four times the size of the base area, the total cost is very high when $G = 60$ minutes as most of the shipments are infeasible to the crowdsourcers and have to be delivered using the backup vehicles. The crowdsourcers if assigned, only can deliver the shipments lower than the maximum allowed in a job which reduces the consolidation effect. When $G$ increases to 120 minutes, significant decrease in the total cost can be observed for the quadrupled area as this relaxes the time constraint and provides greater consolidation opportunities. Further increase in the value of $G$ leads to an increment in the total cost. This occurs because of the long reassignment period, which is fixed at $T_{max}$. Crowdsourcers cannot wait for such long time and the available crowdsourcers pool start to decrease giving the rise in the total cost.

Figure 3-10: Sensitivity of Total Cost with the guaranteed service time ($G$) and the size of the service area
3.5.2.4 Cost sensitivity to DSP location

Finally, the effect of the DSP’s location on the total cost is investigated. Four different locations, near four corners of the neighborhood boundary, are selected as the alternate DSP locations sequentially. The length of the period is fixed to be 33 minutes as this gives the lowest cost among different cases analyzed for the base area. All other parameters are kept unchanged. The total cost for different DSP locations are given in Figure 3-11. The total cost is highest when the DSP is located at the Northwest corner, which is intuitive given the low density of shipments in this area. The lowest cost occurs when the DSP is located at the Northeast corner, from where most of the shipment demands can be reached easily. The intuitive conclusion one can draw from this analysis is that the total shipping cost goes down when the DSP is located near the area with high shipment demands.

![Figure 3-11: Variations in the total cost with respect to the DSP location](image)

3.6 Conclusion

In this study, we proposed a mechanism design based approach for assigning urban shipment jobs to crowdsourcing. The shipments are carried out from the Delivery Service Provider (DSP) by the crowdsourcees who are pedestrians, bicyclists, or car drivers. The study advances the state-
of-art crowdshipping practice by introducing a mechanism design based auction to efficiently allocate shipping jobs to crowdsourcers. The proposed mechanism design based approach takes into account the heterogeneity of crowdsourcers’ willingness-to-do-crowdshipping (WTDC) or VOT and also considers the strategic behavior of the crowdsourcers. Under the proposed mechanism, every crowdsourcer find their dominant strategy to report their bids truthfully. Further, we extend the mechanism to the dynamic scenario where both shipment demands and crowdsourcers arrive over time. The dynamic mechanism myopically optimizes the shipping cost at each discrete period.

Numerical simulations were conducted for both static and dynamic mechanism. For the static mechanism, the performance of the mechanism for varying number of shipments and crowdsourcers are presented. Results show that the mechanism performs well when the numbers of crowdsourcers are no lesser than the number of shipments. As the number of crowdsourcers increases, the total shipping cost as well as the cost of truthfulness decreases. In other words, the DSP is able to attain the total cost closer to the minimum possible cost as the number of crowdsourcers increases. The proposed mechanism outperforms the fixed rate based assignment scheme, traditional auctions and the traditional-truck based delivery with significant percentages. The results also show that the best tradeoff between total shipment cost and solution time can be achieved when a crowdsourcer is allowed to include no more than 3 to 4 shipments in a job.

For the dynamic mechanism, we conducted a case study for a neighborhood in the city of Chicago. The total cost is sensitive to the length of the period considered for the reassignment. When the length of the period is small, the total cost is higher due to the lower consolidation of shipments. The total shipment cost initially reduces with the length of the period but later it again
starts to increase as the longer period means crowdsourcers not being utilized efficiently and spending most of their time waiting for the assignment. A good trade-off for the case study considered is at about half an hour period. We also present the sensitivity of the total cost with the size of the service area, guaranteed service time, and the location of the DSP.

The research can be extended in a few directions. First, one can extend the mechanism to allow crowdsourcers to pick up the shipments from multiple DSP locations. Next, for the greater computation efficiency, the job assignment and payment calculation can be parallelized. One can also implement the heuristics solution instead of the exact solution procedure for solving the payment calculation process.
Appendix 3A: Numerical results for optimal bidding strategy of crowdsourcers

In this appendix, we present the numerical results for the model derived in Section 3.3.4. In particular, we will look at the cost of truthfulness and the saving due to the IC mechanism for the varying number of crowdsourcers present. The distribution of the VOT of crowdsourcers is considered to be uniform in the range of [0,1]. For the uniform distribution, the pdf and CDF are given as:

\[ f_X(\hat{r}_i) = \begin{cases} 1 & \text{for } \hat{r}_i \in [0,1] \\ 0 & \text{otherwise} \end{cases} \]  \hspace{1cm} (A.1)

\[ F_X(\hat{r}_i) = \begin{cases} 0 & \text{for } \hat{r}_i < 0 \\ \hat{r}_i & \text{for } \hat{r}_i \in [0,1] \\ 1 & \text{for } \hat{r}_i > 1 \end{cases} \]  \hspace{1cm} (A.2)

Eq. (7.6) will simplify to the following given the pdf and CDF from A.1 and A.2.

\[ \hat{r}_i = \frac{1+r_i(n-1)}{n} \]  \hspace{1cm} (A.3)

From Eq. (A.3) it can be seen that as \( n \) decreases and approaches to 1, \( \hat{r}_i \) approaches to 1 (the upper bound of the distribution considered). For the numerical analysis, we consider different values of \( n \) (from 10 to 100 at an increment of 10) and generate their true bids as a random value from 0 to 1. The optimal bidding value, when no IC mechanism is present is calculated with the help of Eq. A.3. A.3 requires the true bid value \( (r_i) \) and number of crowdsourcers present \( (n) \) as input and gives the optimal bidding value \( (\hat{r}_i) \) as output for every crowdsourcer. The shipping cost absent the mechanism is therefore the lowest false bid among the bids submitted. When the second price mechanism is implemented, all crowdsourcers submit their true bid but the DSP pays the amount equal to second highest bid to the selected crowdsourcer. We present the saving from
the use of second price mechanism and the cost of truthfulness for implementing the second price auction for 100 random instances in Figure 3-12.

Figure 3-12 (a) Savings from the IC mechanism for varying number of crowdsources, (b) cost of truthfulness for varying number of crowdsources

The vertical axis in Figure 3-12 (a) represents the savings from the second price mechanism for different numbers of crowdsources present. The figure shows that the median saving is very high of about 9% when the number of crowdsources is 10 and goes on decreasing as the number of crowdsources increases, but the median value always remains greater than 0. Figure 3-12 (b) represents the cost of the truthfulness or the difference between the second lowest and lowest bid. Results show that the median value is very low of about 1% for any number of crowdsources present.
4  Permission for reuse of the previously published materials in Chapter 2

---

**ELSEVIER LICENSE TERMS AND CONDITIONS**

Feb 14, 2017

This Agreement between Nabin Kafle ("You") and Elsevier ("Elsevier") consists of your license details and the terms and conditions provided by Elsevier and Copyright Clearance Center.

<table>
<thead>
<tr>
<th>License Number</th>
<th>4047840004845</th>
</tr>
</thead>
<tbody>
<tr>
<td>License date</td>
<td>Feb 14, 2017</td>
</tr>
<tr>
<td>Licensed Content Publisher</td>
<td>Elsevier</td>
</tr>
<tr>
<td>Licensed Content Publication</td>
<td>Transportation Research Part B: Methodological</td>
</tr>
<tr>
<td>Licensed Content Title</td>
<td>Design and modeling of a crowdsourc-enabled system for urban parcel relay and delivery</td>
</tr>
<tr>
<td>Licensed Content Author</td>
<td>Nabin Kafle, Bo Zou, Jane Lin</td>
</tr>
<tr>
<td>Licensed Content Date</td>
<td>May 2017</td>
</tr>
<tr>
<td>Licensed Content Volume</td>
<td>99</td>
</tr>
<tr>
<td>Licensed Content Issue</td>
<td>n/a</td>
</tr>
<tr>
<td>Licensed Content Pages</td>
<td>21</td>
</tr>
<tr>
<td>Start Page</td>
<td>62</td>
</tr>
<tr>
<td>End Page</td>
<td>82</td>
</tr>
<tr>
<td>Type of Use</td>
<td>reuse in a thesis/dissertation</td>
</tr>
<tr>
<td>Portion</td>
<td>full article</td>
</tr>
<tr>
<td>Format</td>
<td>both print and electronic</td>
</tr>
<tr>
<td>Are you the author of this Elsevier article?</td>
<td>Yes</td>
</tr>
<tr>
<td>Will you be translating?</td>
<td>No</td>
</tr>
<tr>
<td>Order reference number</td>
<td></td>
</tr>
<tr>
<td>Title of your thesis/dissertation</td>
<td>Advancing the Urban Parcel Delivery System Using Crowdshipping</td>
</tr>
<tr>
<td>Expected completion date</td>
<td>May 2017</td>
</tr>
<tr>
<td>Estimated size (number of pages)</td>
<td>100</td>
</tr>
<tr>
<td>Elsevier VAT number</td>
<td>GB 494 6272 12</td>
</tr>
<tr>
<td>Requestor Location</td>
<td>Nabin Kafle 842 W Taylor St. (MC 246) 2095 ERF</td>
</tr>
<tr>
<td></td>
<td>CHICAGO, IL 60607 United States Attn: Nabin Kafle</td>
</tr>
<tr>
<td>Publisher Tax ID</td>
<td>98-0397604</td>
</tr>
<tr>
<td>Total</td>
<td>0.00 USD</td>
</tr>
<tr>
<td>Terms and Conditions</td>
<td></td>
</tr>
</tbody>
</table>
References

VITA

NABIN KAFLE

nkafle89@gmail.com

AREA OF SPECIALIZATION
Optimization, Operations Research, Statistics, Data Analysis, Machine Learning,
Transportation System Analysis, Supply Chain and Logistics

EDUCATION

University of Illinois, Chicago, IL
Ph.D. Candidate, Civil and Materials Engineering expected May 2017
• Thesis: Advancing the Urban Parcel Delivery System Using Crowdshipping
  • Advisor: Bo Zou, Ph.D.
M.S., Civil and Materials Engineering May 2015
• GPA: 4.0/4.0
  • Advisor: Bo Zou, Ph.D.

Institute of Engineering, Pulchowk, Nepal
B.E. in Civil Engineering Dec 2011
• GPA: 79/100

PUBLICATIONS

BOOK CHAPTERS
PRESENTATIONS


WORK EXPERIENCE

Research Assistant, University of Illinois, Chicago  
Jan 2013 to Present

- Designed innovative crowdsourced-based last mile logistical solutions for urban parcel delivery. Includes the formulation and solution of mixed integer problems using CPLEX and development of efficient heuristic solution methods. Monte-Carlo simulations and case studies demonstrate that the proposed methodologies can result up to 25% cost savings to Shippers. Earlier project also includes a dynamic pricing model for parking slot assignment problems. Three methodological papers published in Transportation Research Part B.
- Data analysis and application of machine learning tools to predict the shipping price for 3PL services for a logistics company.
- Developed a methodology for evaluating airline short and long-term fuel efficiency using the statistical modeling. The methodology is adopted by International Council on Clean Transportation to rate the airline’s fuel efficiency.
- Analyzed the data on flight delay to figure out the pattern and possible solution strategies for flight delay propagation. Includes the collection, filtering and advanced statistical analysis of large data (more than 700,000 observations).
- Evaluated the role of federal funding on the performance of the US primary airports and suggested financial reforms required to improve an airport’s performance. Includes the mathematical modeling and optimization of the linear program.
- Cost benefits analysis, economic impact analysis and social impact analysis of high-speed rails in the US.

Teaching Assistant, University of Illinois, Chicago  
Jan 2013 to Present

- Introduction to Transportation Engineering. 50+ students every Spring.
- Design of Reinforced Concrete Structures. 60 students, Fall 2015.

Consultant, Rensselaer Polytechnic Institute  
Summer 2015

- Led a team of 4 for an NSF-funded project to conduct field surveys and on-site interviews with disaster responders during the M7.8 Earthquake at Nepal on May 2015.
- Figured out the key challenges hindering the humanitarian logistics activities in the disaster-struck area and helped in documenting the findings in the form of a journal paper.

Design Engineer, Full Bright Consultancy, Kathmandu  
2012-2013

- Design and planning for 40,000-hectare land irrigation system.
- Analyze the lidar maps in ArcGIS, and prepare maps and layout of irrigation canal alignment in ArcGIS.
- Project Management and scheduling of personnel for different tasks.
- Modeling and forecast of water discharge on rivers.
- Design of hydraulic structures for irrigation.
- Cost analysis and logistics handling.

**Leadership Roles and Awards**

- ITE Illinois Section Graduate Student Award, 2017 (Statewide award).
- Student Presenter Award, 2017 (University-wide award).
- Graduate Student Council Travel Award, 2017 (University-wide award).
- UIUC Chancellor's Student Service and Leadership Award, 2016 (University-wide award).
- David Boyce Transportation Award, 2016 (Department-wide award).
- Student Presenter Award, 2016 (University-wide award).
- Graduate Student Council Travel Award, 2016 (University-wide award).
- President, ITE Student Chapter at University of Illinois at Chicago, 2016.
- President, Rotaract Club of Patan West, Kathmandu, Nepal 2012.

**Skills**

- Programming
  - Python including packages like Numpy, Scikit-learn, Matplotlib, and Pandas (Intermediate to Advanced), Matlab (Advanced), C++ (Beginner)
- Statistical software
  - Stata (Advanced), R (Intermediate)
- Others
  - CPLEX (Advanced), ArcGIS (Advanced), Excel (Advanced), MySQL (Intermediate)