https://doi.org/10.1016/j.trb.2023.102831

An exact algorithm for the pickup and delivery problem with crowdsourced bids and transshipment

E Su[b], Hu Qin[b], Jiliu Li[a], Kai Pan[c]

[a] *School of Management, Northwestern Polytechnical University, Xi'an 710072, China*
[b] *School of Management, Huazhong University of Science and Technology, Wuhan 430074, China*
[c] *Department of Logistics and Maritime Studies, Faculty of Business, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong*

Abstract

This paper addresses a pickup and delivery problem with crowdsourced bids and transshipment (PDPCBT) in last-mile delivery, where all requests can be satisfied by either using the own vehicle fleet or outsourcing with a small compensation to crowdshippers through transshipment facilities. The crowdshippers show their willingness to deliver by submitting bids to the e-commerce company. To minimize both the travel cost of vehicles and the compensation of crowdshippers, the routes of vehicles and the selection of bids need to be optimized simultaneously. We formulate the PDPCBT into an arc-based formulation and a route-based formulation, where the latter is strengthened by the subset row inequalities. Based on the route-based formulation, we present a branch-and-price-and-cut algorithm to solve it exactly. To deal with two possible ways of serving requests, we first decompose the corresponding pricing problem into a shortest path problem and a knapsack problem, and then tackle them in the same bi-directional labeling algorithm framework. We also discuss acceleration techniques and implementation details to speed up the performance of the overall procedure. Computational experiments are conducted on a set of classic request instances, together with a set of randomly generated bid instances. With a time limit of two hours, numerical results validate the efficiency and effectiveness of the proposed algorithm. Finally, sensitivity analysis and managerial findings are also provided on the PDPCBT.

Keywords: vehicle routing; pickup and delivery; crowdsourced delivery; transshipment facilities; branch-and-price-and-cut.

1. Introduction

Urban logistics has recently been changing dramatically with the explosion of e-commerce, especially the last-mile delivery between a local depot and customers. Aized and Srai (2014)

Email addresses: es1996@hust.edu.cn (E Su), tigerqin1980@qq.com (Hu Qin), lijiliu1992@outlook.com (Jiliu Li), kai.pan@polyu.edu.hk (Kai Pan)

emphasized that the cost of this leg accounts for 13% up to 75%, which is the most expensive part of the whole supply chain. Worse still, companies are facing an embarrassment where the needs of their customers cannot be satisfied in time with increasing order volumes. Besides, truck-only transportation has led to traffic congestion and other serious impacts on the environment. In the United States, for example, truck activities contributed 28.5% of greenhouse gas emissions in 2016 (Qin et al., 2021). To mitigate these problems, e-commerce companies have sought a new delivery way called crowdshipping, where companies outsource their deliveries to crowdshippers who are willing to earn extra compensation (Alnaggar et al., 2019).

Naturally, crowdshipping offers numerous advantages for e-commerce companies. First and foremost, it can dramatically save operating costs in last-mile delivery because the compensation paid for crowdshippers is much less than the travel cost of trucks (Le et al., 2019). Secondly, rather than scheduling a special trip, companies can be more efficient in dealing with remote and urgent deliveries with the help of crowdshippers (Boysen et al., 2021). Finally, society benefits from crowdshipping by utilizing excess transportation capacity for freight deliveries without employing more trucks in local areas. Therefore, it can provide potential chances for social collaboration and reduce pollutant emissions to some extent (Pourrahmani and Jaller, 2021).

However, crowdsourced delivery is different from specialized distribution in which carriers are dedicated to executing delivery activities. Instead, crowdshippers are only hired for a short time to make a detour without a long-term salary. The short-tour delivery makes it critical for convenient package transshipment between trucks and crowdshippers. Macrina et al. (2020) indicated that crowdshippers prefer detouring to a transshipment facility closer to the delivery area rather than directly visiting the depot, which also needs less compensation for the deviation. In this way, trucks and crowdshippers must transfer packages at the facility to complete the final delivery for customers. Such efficient systems require coordination between trucks and crowdshippers when decision-making, inevitably leading to a more complex situation in logistics transportation.

This paper contributes to a pickup and delivery problem with crowdsourced service and package transshipment in last-mile delivery. Each request considered in the problem originates from a sender who requires a given quantity of packages to be picked up first and ends up with a receiver who then receives them. With the participation of crowdshippers, there are two possible options for serving requests, namely direct and indirect deliveries. The former type indicates that both the sender and receiver of the same request are visited by only one vehicle or one crowdshipper, whereas the latter allows a request to be served by a vehicle and a crowdshipper through transshipment.

Figure 1 presents an illustration of the proposed system where two trucks, three transship-

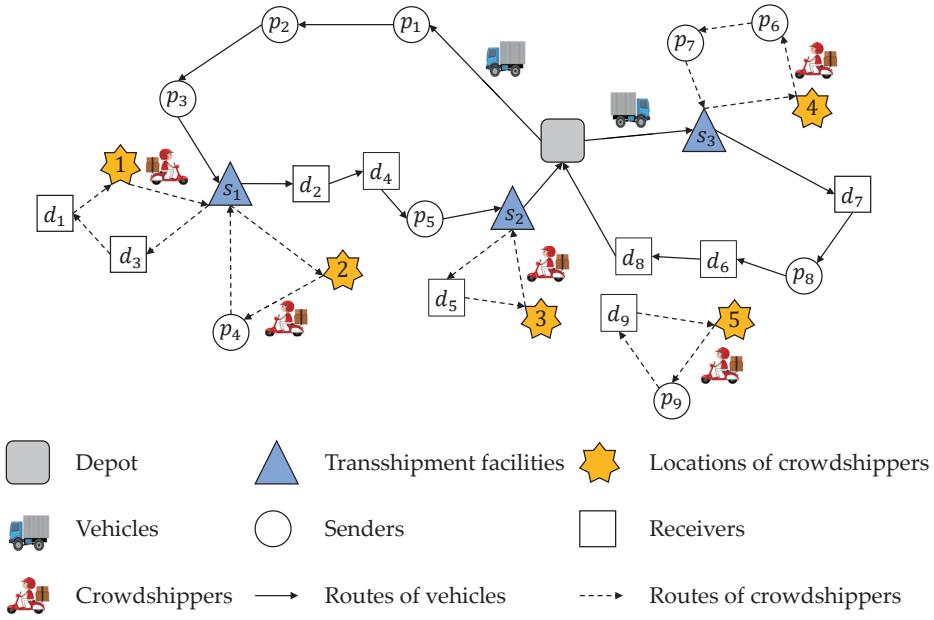


Figure 1: An illustration of the transportation system with crowdshipping.

ment facilities, five crowdshippers, and nine requests are considered. Requests 2 and 8 are directly served by trucks, and request 9 is completed by crowdshipper 5. The packages of these three requests are first collected from their senders and then delivered to the corresponding receivers. Note that no transshipment occurs for the packages, even if they pass through the facility. The other six requests correspond to indirect deliveries. Take the left truck and transshipment facility s_1 as an example. The truck first picks up the packages at senders p_1 and p_3 to s_1, and crowdshipper 1 subsequently takes the packages to their receivers d_1 and d_3. While crowdshipper 2 first visits sender p_4 and transports the packages to s_1, from which the truck continues delivering them to its receiver d_4. Compared with the locations of the crowdsourced customers, the locations of transshipment facilities are closer to that of the truck-visited customers. Therefore, instead of directly visiting these customers, integrating crowdshipping into a truck-only transportation system would potentially lead to cost-saving by reducing the travel distances of trucks.

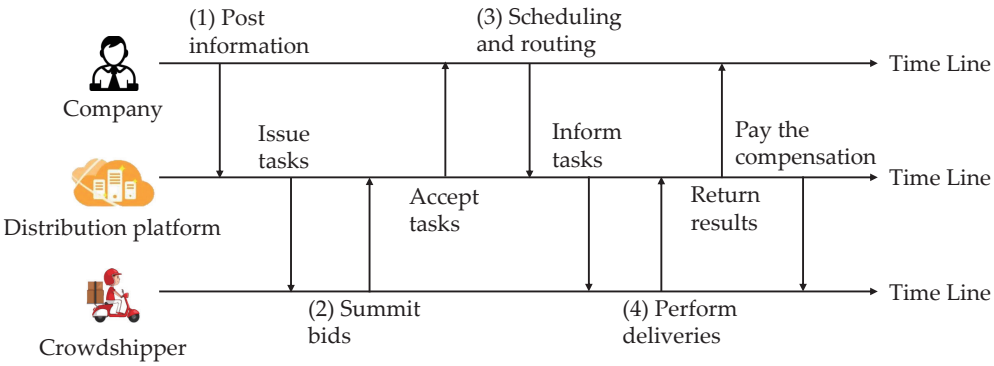


Figure 2: How the crowdshipping distribution platform works.

The implementation of crowdshipping relies on an online distribution platform where e-commerce companies can plan their day-ahead operation. In addition to the system introduced by Kafle et al. (2017), such crowdsourced delivery platforms have also emerged in China, such as Yun Niao (Zhang and Li, 2017; Lu, 2022) and Jing Dong (Wang, 2018). Intuitively, we summarize how a platform works in Figure 2. First, the e-commerce company posts information about all tasks to attract crowdshippers. Then crowdshippers submit bids for transportation tasks to the platform. Each generated bid specifies task information, such as the customers to be served, the transshipment facility, expected compensation, etc. If a bid is selected in decision-making, the corresponding crowdshipper would be informed of the execution time and receive compensation for his work. The company thereby needs to determine (i) the set of customers served by crowdshipping and (ii) the routes to visit those transshipment facilities associated with the selected bids and visit the customers who are not served by crowdshipping. We regard this specific delivery practice as the crowdshipping problem with transshipment (PDPCBT) and aim to achieve the optimal solution by minimizing the total shipping cost of vehicle transportation and crowdshippers' compensation.

Despite the introduction of the transshipment facility, it would be counterproductive to the cost-saving benefit when poorly scheduling routes of trucks or randomly selecting outsourced customers (Huang and Ardiansyah, 2019). Therefore, it is necessary to maintain a balance between the utilization of trucks and the selection of crowdsourced bids, which inevitably complicates the decision process. Addressing the pickup and delivery problem with crowdshipping and transshipment has great industrial practice. To the best of our knowledge, however, the existing research on crowdshipping with transshipment mainly focuses on heuristic algorithms. In this paper, we endeavor to propose an exact algorithm to fill this research gap and the contributions of our study are threefold:

1. We formulate two formulations for the PDPCBT. The first one is an arc-based formulation based on a logical network. It is straightforward, however, the computational effort to obtain an optimal solution increases exponentially with the problem size. Therefore, we reformulate the PDPCBT as a route-based formulation depending on a physical network.

2. We propose a branch-and-price-and-cut algorithm to exactly solve the route-based problem, in which the problem is decomposed into a shortest path problem and a knapsack problem. To deal with these two problems simultaneously in the same algorithm framework, we develop a bi-directional labeling algorithm with novel acceleration techniques. Subset row cuts as valid inequalities are also introduced to accelerate algorithm convergence.

3. We generate a set of test instances for the PDPCBT based on the requests introduced by

Ropke and Cordeau (2009) and random bid information. To validate the effectiveness of the proposed algorithm, extensive computational experiments have been conducted, and sensitivity analysis and managerial findings on the PDPCBT are also provided.

The remainder of this paper is organized as follows. We present a brief review of related studies in Section 2. Section 3 introduces the definition and assumption of the PDPCBT, followed by two different mathematical formulations. Section 4 presents the details of the branch-and-price-and-cut algorithm framework. Section 5 provides computational results on extensive experiments. We end up with conclusions and future research in Section 6.

2. Literature review

Deriving from the concept of sharing economy, crowdshipping has been widely integrated into the daily truck-only transportation of e-commerce companies, where delivery activities are allowed to be outsourced to crowdshippers (Mourad et al., 2019). Meanwhile, a transfer opportunity between trucks and crowdshippers at transshipment facilities has also been considered in last-mile delivery. In this section, we will introduce the literature on crowdshipping and utilizing transshipment facilities from the view of operations research.

2.1. Crowdshipping in city logistics

Based on the crowdshipping concept proposed by Walmart, Archetti et al. (2016) presented a vehicle routing problem with occasional drivers (VRPOD), which is the first one to incorporate crowdshipping into the logistics network. This problem allows requests to be accomplished by either the own vehicle fleet or crowdshippers called occasional drivers (ODs) who may detour to transport packages if the location of the request is near their destination and receive compensation in return. For example, in-store customers are willing to deliver items ordered by online customers. No routing is scheduled for ODs since they only perform a single delivery. To solve the proposed VRPOD, a multi-start heuristic was designed and implemented. The results showed significant benefits for a company to permit crowdshipping and indicated that employing an appropriate compensation scheme for ODs is important for cost savings. Macrina et al. (2017) extended this work by considering time windows and presented two models where one introduces the split delivery policy for ODs, and another allows ODs to implement multiple deliveries. Boysen et al. (2022) investigated a new crowdsourced delivery pattern in which employees in-store act as crowdshippers to transport packages from the depot. To maximize the number of shipments assigned to employees, a matching problem is presented by considering multiple deliveries limited by capacity and routing for each crowdshipper. The above researches assume that crowdshippers only ship packages from the depot to customers. Instead of picking items up from

the depot, Behrend et al. (2019) studied a form of crowdshipping where crowdshippers detour to deliver multiple requests from suppliers to consumers in need in the context of item-sharing. Compared with a pure item-sharing platform where the consumer must collect the items themselves, the extensive experiments indicated that integrating crowdshipping with item-sharing is more profitable and customer-friendly.

Our work can be viewed as an extension of the pickup and delivery problem (PDP), where requests require vehicles to pick up packages from an origin location and deliver them to a destination. As a class of the vehicle routing problem (VRP), the PDP further imposes precedence and pairing constraints on each request, i.e., the origin must precede the destination, and both locations must be visited by the same vehicle (Ropke and Cordeau, 2009; Baldacci et al., 2011). Dahle et al. (2019) focused on the pickup and delivery problem with time windows and occasional drivers, in which the design of compensation schemes for ODs is considered. Two formulations modeled the ODs' behavior and experiment results showed that using ODs can obtain substantial cost savings, even when the scheme is not optimal. Arslan et al. (2019) studied a new variant of the dynamic pickup and delivery problem where both delivery tasks and the crowdshippers arrive dynamically over time. They modeled it as a matching problem and proposed a rolling horizon framework for its solution. Results suggested a 37% reduction in costs using such a dynamic crowdshipping system compared to a traditional truck-only system. All these studies require crowdshippers to deliver packages directly without any transfer, and therefore the collaboration between trucks and crowdshippers is not considered.

2.2. Transshipment facilities for urban delivery

Several studies on integrating transshipment facilities in a distribution system have recently been presented without crowdshipping. Motivated by the production and distribution of a real-world company, Baldacci et al. (2017) addressed the vehicle routing problem with transshipment facilities (VRPTF), in which each customer can be served by a private vehicle or through an intermediate facility. In the latter case, the demand is first transferred to a facility on a vehicle route, and then it is fully outsourced to complete the final delivery. A similar transshipment setting is also studied by Alcaraz et al. (2019), where they designed a set of heuristic algorithms to solve a rich vehicle routing problem for long-haul transport with transshipment options in last-mile delivery. Friedrich and Elbert (2022) proposed an adaptive large neighborhood search (ALNS) for the VRPTF variant by incorporating time windows, heterogeneous fleets, and different cost functions such as distance-dependent, time-dependent, and fixed cost. Zhou et al. (2018) considered a two-level distribution system in which the first level requires vehicles to deliver the demands to the transshipment facilities while the second level designs routes to serve customers from the

facilities. In addition to direct parcel delivery, customers may pick up their demands at the intermediate facilities. A hybrid multi-population genetic algorithm is developed to minimize the total distribution cost.

As for the PDP, by allowing transshipment within vehicle routes, Cortés et al. (2010) introduced the PDP with transfer (PDPT), where passengers can transfer between vehicles. In the PDPT, the pairing constraint is relaxed to that a request can be transferred from one vehicle to another at a transfer point, which still respects the precedence constraints. They addressed a comprehensive arc-flow model and optimally solved small instances with a branch-and-cut algorithm. Since the exact algorithm is unable to solve large instances, Masson et al. (2013) developed an efficient ALNS algorithm for solving the PDPT within a reasonable time. Their results also confirmed that the introduction of transshipment facilities could reduce costs, although these improvements depend on the location of transfer points in the instances. Ghilas et al. (2018) studied a hybrid transportation system that synchronizes pickup and delivery vehicles with scheduled lines like buses where requests are allowed to be transshipped between two terminals.

2.3. Crowdshipping with transshipment

When looking into crowdshipping with transshipment, Macrina et al. (2020) worked on a variant where each crowdshipper performs a single delivery by relaying parcels from either the depot or those transshipment facilities at which trucks have delivered the parcels. They developed a variable neighborhood search heuristic and highlighted the benefit of introducing transshipment facilities in the logistics network. By allowing multiple deliveries for crowdshippers and an additional delivery option that customers can pick up at intermediate facilities, Yu et al. (2022) proposed an ALNS heuristic to solve their mixed integer nonlinear programming model. dos Santos et al. (2022) considered three delivery options simultaneously: parcel lockers, crowdshippers, and dedicated trucks in a distribution model. Parcel lockers are not only used for consolidation but also parcel transshipment. After collecting their own parcels from the lockers, customers can act as crowdshippers to execute single delivery to others en route to their destination in this case. Results show that integrating crowdshippers and parcel lockers can achieve cost-savings even when customers only detour no more than 25% of their route.

The crowdsourced distribution platform in this paper is similar to the work of Kafle et al. (2017), in which they considered both trucks and crowdshippers to execute the last-mile delivery between customers and transshipment facilities. The e-commerce company first posts requests on the platform, and crowdshippers submit available bids that are limited by capacity and travel distance. Then the truck routes visit several necessary transshipment facilities associated with the winning bids and customers who are not covered by bids. A tabu search based algorithm

Table 1: Summary of the PDPCBT and related variants in crowdshipping.

Works	Problem Feature	Transshipment Facility	Package Place	Delivery Way	Time Windows	Model Formulation	Algorithm
Archetti et al. (2016)	VRP		D	Single		ABF	Heuristic
Macrina et al. (2017)	VRP		D	Multiple	✓	ABF	Solver
Kafle et al. (2017)	VRP	✓	T, C	Multiple	✓	ABF	Heuristic
Huang and Ardiansyah (2019)	VRP	✓	T, C	Multiple	✓	ABF	Heuristic
Macrina et al. (2020)	VRP	✓	D, T	Single	✓	ABF	Heuristic
Yu et al. (2022)	VRP	✓	D, T	Multiple	✓	ABF	Heuristic
dos Santos et al. (2022)	VRP	✓	D, T	Single		ABF	Solver
Dahle et al. (2019)	PDP		C	Multiple	✓	ABF	Solver
Sampaio et al. (2020)	PDP	✓	T, C	Multiple	✓	ABF	Heuristic
Voigt and Kuhn (2021)	PDP	✓	T, C	Multiple	✓	ABF	Heuristic
This paper	PDP	✓	T, C	Multiple	✓	ABF, RBF	Exact

D: Depot; T: Transshipment facility; C: Customer; ABF: Arc-Based Formulation; RBF: Route-Based Formulation.

was proposed to minimize the sum of truck travel cost, crowdsourced compensation, and delay penalties for servicing outside customers' desired time windows. Huang and Ardiansyah (2019) followed this work by simultaneously specializing in the routing for vehicles and crowdshippers. Since they imposed time window constraints on customers, their objective excludes the time penalty cost while the compensation of each crowdshippers includes a fixed cost and a time-based variable cost. Later, Feng et al. (2021) investigated an integrated production and transportation scheduling problem in the context of crowdshipping. Taking into account both the shipping cost and customer service level, they proposed a mixed integer programming model and then efficiently solved it with a decomposition-based genetic algorithm.

Apart from the above, the PDP with crowdshipping and transshipment has also been proposed but occurs rarely. Sampaio et al. (2020) analyzed the potential benefits of the PDPT with time windows in a crowdsourced system. They assumed that crowdshippers express their availability to work only for a given amount of time and implemented an ALNS to solve the problem efficiently. Contrasting Sampaio et al. (2020), Voigt and Kuhn (2021) considered that crowdshippers could carry out the tasks anytime and incorporated the own vehicle fleet as an alternative option to serve requests. They also proposed an extension of the ALNS for the problem. Table 1 summarizes ten mostly related variants to our work in crowdshipping. As can be seen, heuristic algorithms and commercial solvers are often presented to address these characteristics in one problem. To the best of our knowledge, no exact algorithm has been developed, and we mainly attempt to address the gap based on a route-based formulation in this work.

3. Problem description, assumption, and formulation

This section first introduces the problem setting in Section 3.1, and then formally define the PDPCBT in Section 3.2. We end this section with two formulations of the PDPCBT, namely an arc-based formulation in Section 3.3 and a route-based formulation in Section 3.4. To help better track the narrative, Table 2 list the notations defined in this section.

Table 2: Notations used in Section 3.

Notation	Definition	First appeared section
Sets		
$\{0\}/\{0^+, 0^-\}$	Depot	3.2/3.3
V	Set of vertices in a physical network	3.2
A	Set of arcs in a physical network	3.2
V'	Set of vertices in a logical network	3.3
A'	Set of arcs in a logical network	3.3
A_L	Set of logical arcs in a logical network	3.3
A_P	Set of physical arcs in a logical network	3.3
P	Set of senders (pickup vertices) or requests	3.1
D	Set of receivers (delivery vertices)	3.1
S	Set of facilities	3.1
B	Set of bids	3.1
B^r	Set of request bids	3.3
$B^r(i)$	Set of request bids that serve request i	3.3
B^p	Set of pickup bids	3.3
B^d	Set of delivery bids	3.3
$B(i)$	Set of pickup/delivery bids that serve customer i	3.3
C_b	Set of customers served by bid b	3.1
K	Set of vehicles	3.2
R	Set of feasible routes	3.4
Indices		
i	Index of customers (senders or receivers) or requests	3.1
b	Index of bids	3.1
(i, j)	Index of arcs	3.2
k	Index of vehicles	3.3
r	Index of feasible routes	3.4
Parameters		
n	Number of requests	3.1
q_i	Demand of a customer $i \in P \cup D$ in kilograms	3.1
$[e_i, l_i]$	Time window of a customer $i \in P \cup D$	3.1
$[e_0, l_0]$	Planning horizon	3.2
ρ	Compensation coefficient of a crowdshipper	3.1
s_b	Facility for package transshipment in bid b	3.1
q_b	Load has to be carried by the crowdshipper in bid b in kilograms	3.1
p_b	Price of bid b in kilometers	3.1
τ_b	Release time for the pickup bid or due time for the delivery bid	3.1
Q	Vehicle capacity in kilograms	3.2
c_{ij}	Travel cost of arc $(i, j) \in A$ in kilometers	3.2

t_{ij}	Travel time of arc $(i,j) \in A$ in minutes	3.2
M	Large positive number	3.3
c_r	Cost of a route $r \in R$ in kilometers	3.4
α_{ijr}	Binary variable indicating whether or not route r travels arc $(i,j) \in A$	3.4
y_{br}	Binary variable indicating whether or not route r selects bid $b \in B^p \cup B^d$	3.4
λ_{ir}	Binary variable indicating whether or not route r contains request $i \in P$	3.4
Variables		
x_{ij}^k	Binary variable indicating whether or not vehicle k travels along arc $(i,j) \in A'$	3.3
ω_b	Binary variable indicating whether or not request bid b is selected	3.3
A_i^k	Continuous variable representing the time when vehicle k arrives at vertex i	3.3
T_i^k	Continuous variable representing the time when vehicle k starts service at vertex i	3.3
Q_i^k	Continuous variable representing the current load of vehicle k after serving vertex i	3.3
θ_r	Binary variable indicating whether or not route r is selected	3.4
μ_i	Dual variable associated with constraint (3b)	3.4
μ_0	Dual variable associated with constraint (3c)	3.4

3.1. Tasks, transshipment facilities and crowdsourced bids

Considering a set of n requests as distribution tasks, each request i has a sender corresponding to a pickup vertex i and a receiver associated with a delivery vertex $n+i$. We also refer to a specific request i by its sender for simplicity. Denote by $P = \{1, \ldots, n\}$ and $D = \{n+1, \ldots, 2n\}$ the set of pickup and delivery vertices, respectively. Each sender $i \in P$ has a positive demand q_i, which should be delivered to its corresponding receiver $n+i$, namely $q_{n+i} = -q_i$. Let $[e_i, l_i]$ denote the time window of the vertex $i \in P \cup D$ within which service must be started. We allow vehicles or crowdshippers to wait before the time window but do not allow arriving later than it. The set of transshipment facilities (or facilities, for short) is denoted by S. Assume that package's size is small and each facility can provide large enough room to temporarily store all transferred packages as in Voigt and Kuhn (2021) and dos Santos et al. (2022).

We now present a conceptual picture of crowdsourced bids based on a real-world application in China and the crowdsourced setting described by Kafle et al. (2017). As shown in Figure 2, the posted tasks become visible on the distribution platform, and crowdshippers bid for tasks in Step (2). The details to simulate the result of submitting feasible bids are described in the sequel.

Before the bidding process, crowdshippers have their own private information including the origin, available time, and willingness to do crowdshipping. We entail the following assumptions on the information. First, as crowdshippers prefer submitting bids for a short tour, they only consider facilities and customers that are located within a predefined activity radius of the origin. Moreover, we assume that crowdshippers are reliable, i.e., they have expressed their willingness truthfully and planned their spatial flexibility because requests are known beforehand. Finally, due to the heterogeneity of crowdshippers, we use a random compensation coefficient ρ representing the willingness of different crowdshipper to make a detour. Suppose that the higher

the willingness, the less the expected compensation. Then the bid price is the summation of the detour distances multiplied by ρ, which differs from Kafle et al. (2017) where they calculate the price as the total travel time multiplied by a constant time value.

The bid generation and submission justify by the following constraints. First, we restrict the maximum number of customers in a bid given the limited capacity of crowdshippers and time windows of customers. Second, we consider two bidding rules for the customer type. One rule limits a bid can serve only senders or only receivers, which we respectively call *pickup bid* and *delivery bid* hereafter. Another allows a bid to be mixed only if it contains only one request, which we call *request bid*. The bidding rules state that it would be better to handle pickup and delivery tasks as separately as possible. As the platform has not yet scheduled the routes, crowdsourced delivery may bring more complexity and uncertainty when integrating pickups and deliveries simultaneously. Subjecting to the feasibility, crowdshippers specify the set C_b of customers to be served and facility s_b for package transshipment in each feasible bid b. Note that request bids do not require s_b since crowdshippers directly deliver packages from senders to their receivers.

To remain bid competitive with others and maximize one's utility, the route serving C_b must be cost-minimum for the crowdshipper. Let q_b be the total load that has to be carried in b. For these parameters, determining a minimum-cost route equates to a traveling salesman problem with time windows (TSPTW) that has been well-studied and can be solved to optimality for an instant using CPLEX (Laporte, 1992). The route starts and ends at the crowdshipper's origin, visiting the facility and all customers within their time windows in the bid. One more constraint for the route is that s_b must be the last visit for the pickup bids or the first visit for delivery bids. Whenever a TSPTW is solved, the corresponding bid price p_b is the consequent travel cost multiplied by the willingness. Moreover, the package available time τ_b at s_b can be easily obtained during calculation.

Consequently, each bid b is denoted by a tuple $\{C_b, s_b, q_b, p_b, \tau_b\}$. We distinguish the last attribute τ_b between the pickup and delivery bids. For a pickup bid, we assume τ_b is a release time representing the earliest time when the crowdshipper can deliver the packages to the facility and return to one's origin. Thus, the vehicle cannot take the stored packages at s_b until τ_b, otherwise the driver must wait for the crowdshipper if this pickup bid has been selected. For a delivery bid, a vehicle should deliver the packages to the stored room before the due time τ_b, otherwise the crowdshipper has no time to carry out this delivery bid. Indeed, a crowdshipper may not be always available to perform the delivery at times, so this setting is reasonable. According to the assumption about information, τ_b can include the possible service time duration at s_b.

After bid generation, we assume that each crowdshipper designates at most one preferable bid, avoiding who may win multiple bids with conflicting working times and cannot provide

dependable delivery services due to their non-professional (Feng et al., 2021). This assumption also helps reduce the computational burden (Kafle et al., 2017). Finally, the set of all submitted bids, denoted by B, is then served as input in the subsequent decision-making.

3.2. Problem definition

Formally, the PDPCBT is defined on a physical network $G^P = (V, A)$, where $V = \{0\} \cup P \cup D \cup S$ is the set of physical vertices and $A = \{(i, j)|i, j \in V, i \neq j\}$ is the set of physical arcs. Vertex 0 is a special location called depot, where a fleet K of homogeneous vehicles with a capacity $Q \geq \max_{i \in P}\{q_i\}$ for service. For notational convenience, the planning horizon is represented by a time window $[e_0, l_0]$, limiting the earliest and latest times at which vehicles can depart and return. Each arc $(i, j) \in A$ has a travel cost c_{ij} and a travel time t_{ij}. We assume that t_{ij} includes the service time at vertex i, and both the travel cost and time satisfy the triangle inequality.

The objective is to minimize both the compensation for crowdshippers and the travel cost of trucks. Particularly, there are two aspects of decisions of the considered problem: (i) bid selection to determine crowdsourced customers and corresponding facilities that need to be visited, and (ii) vehicle routing that visits facilities associated with the selected bids and customers who are not served by crowdshipping. The decisions impact each other and entail the following assumptions and constraints, which are grouped into three categories including requests, facilities, and routes:

Requests. Each request must respect the precedence and pairing constraints. The former restricts the pickup vertex must be visited before its delivery. While the latter can be satisfied by one of the following four cases: (1) Both pickup and delivery vertices are visited by the same vehicle, which is the same as the classical PDPTW; (2) Both pickup and delivery vertices are served by a request bid; (3) A crowdshipper first visits the sender and delivers the package to the facility (pickup bid), a vehicle then picks up the package at this facility and delivers it to the corresponding receiver; and (4) The package is picked up from a sender by a vehicle to a facility, then a crowdshipper visits this facility and delivers the package to its receiver (delivery bid). Unlike the first two so-called direct deliveries, the last two are indirect deliveries that require parcel transshipment. Given that serving the sender and receiver of the same request by two different crowdshippers may suffer unpredictable delays due to their unfamiliarity with each other, we do not consider a mode where a crowdshipper picks up the package for transshipment, and another then delivers the dropped package to complete the request. All requests must be satisfied exactly once, which means each customer is served only once by a vehicle or a certain bid.

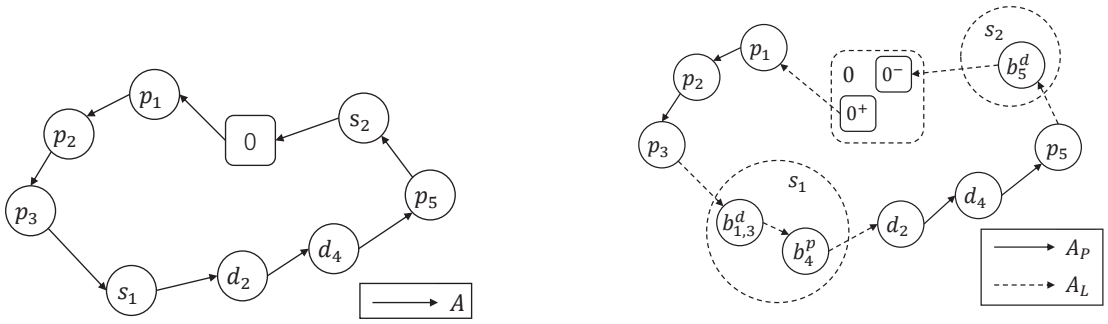
Facilities. According to the selected pickup and delivery bids, a facility may not be visited at all or at least once. Note that the coordination between vehicles and crowdshippers at the facility is one-to-one. Precisely, only the selected pickup bids assigned to this route, can the vehicle carry

the corresponding transshipment packages. In addition, the vehicle can transfer those loads to the facility only if the corresponding delivery bids are assigned to this route. In a word, packages in one bid or one vehicle are not allowed to be split and transferred to different vehicles or crowdshippers.

Routes. Each vehicle route must start at the depot, visit several requests or facilities, and finally returns to the depot. The vehicle capacity must be respected along the route. Because vehicles do not have to wait too long for the packages, and the facility has provided a temporary storage room, one facility can be visited multiple times at different times by the same vehicle if multiple pickup and delivery bids related to the facility have been assigned to this route.

3.3. Arc-based formulation

This section formulates the PDPCBT as an arc-based formulation (ABF). Under the assumption that each vehicle may visit the same facility multiple times, we must distinguish the load and time of each visit. Therefore, we duplicate a logical vertex set $V' = \{0^+, 0^-\} \cup P \cup D \cup B$, where 0^+ and 0^- represent the starting and returning depots, respectively, and the pickup and delivery bids share the same location as their transshipment facility. Then a set A_L of logical arcs connecting the duplicated depots and bids is created, and the calculations of the travel cost and travel time between vertices in V' also take into account the bid price and physical locations. Consequently, the ABF is defined on a logical network $G^L = (V', A')$, where the arc set A' consists of the logical arc set A_L and a physical arc subset $A_P = (P \cup D) \times (P \cup D)$. Figure 3 shows a solution example on the logical network corresponding to the route of the left truck in Figure 1. For the solid circles in Figure 3.(b), the subscripts match the index of request, and the letters p and d represent the pickup and delivery operations, respectively.



(a) the physical network (b) the logical network

Figure 3: A solution example for the physical and logical networks.

Note that symmetry could occur for the bids that belong to the same facility, which means different route structures express the same solution. For example, let b_i and b_j represent two different bids in the same physical location. As a route can select these two bids in the same visit

and travel between them takes no time, the visited sequences within two routes $(\ldots, b_i, b_j, \ldots)$ and $(\ldots, b_j, b_i, \ldots)$ have no difference. To break the symmetry, we impose the index of the delivery bid precedes that of the pickup bid and eliminate an arc set $\{(b_i, b_j) | i \geq j, s_{b_i} = s_{b_j}\}$.

Let x_{ij}^k be a binary variable that takes value 1 if vehicle k travels along arc $(i,j) \in A'$, and 0 otherwise; ω_b be a binary variable that takes value 1 if request bid b is selected, and 0 otherwise. Continuous variables A_i^k, T_i^k, and Q_i^k represent the time at which vehicle k arrives at vertex i, the time when service starts at vertex i by vehicle k, and the current load of vehicle k after serving the vertex i, respectively. Using the additional sets introduced in Table 2, the ABF is presented as follows:

$$(ABF)\ \min \sum_{k\in K}\sum_{i\in V'}\sum_{j\in V'} c_{ij}x_{ij}^k + \sum_{k\in K}\sum_{b\in B^p\cup B^d}\sum_{j\in V'} p_b x_{bj}^k + \sum_{b\in B^r} p_b w_b \tag{1a}$$

$$\text{s.t.} \sum_{k\in K}(\sum_{j\in V'} x_{ij}^k + \sum_{b\in B(i)}\sum_{j\in V'} x_{bj}^k) + \sum_{b\in B^r(i)} w_b = 1,\ \forall i \in P \tag{1b}$$

$$\sum_{j\in V'} x_{ij}^k + \sum_{b\in B(i)}\sum_{j\in V'} x_{bj}^k = \sum_{j\in V'} x_{n+i,j}^k + \sum_{b\in B(n+i)}\sum_{j\in V'} x_{bj}^k,\ \forall i \in P, k \in K \tag{1c}$$

$$\sum_{k\in K}\sum_{b\in B(i)}\sum_{j\in V'} x_{bj}^k + \sum_{k\in K}\sum_{b\in B(n+i)}\sum_{j\in V'} x_{bj}^k \leq 1,\ \forall i \in P \tag{1d}$$

$$\sum_{j\in V'} x_{0^+,j}^k = 1,\ \forall k \in K \tag{1e}$$

$$\sum_{j\in V'} x_{ji}^k - \sum_{j\in V'} x_{ij}^k = 0,\ \forall i \in P\cup D\cup B^p\cup B^d, k \in K \tag{1f}$$

$$\sum_{i\in V'} x_{i,0^-}^k = 1,\ \forall k \in K \tag{1g}$$

$$A_i^k \leq T_i^k,\ i \in V', k \in K \tag{1h}$$

$$T_i^k + t_{ij} - A_j^k \leq M(1 - x_{ij}^k),\ \forall i \in V', j \in V', k \in K \tag{1i}$$

$$T_i^k + t_{i,n+i} - T_{n+i}^k \leq M(2 - \sum_{j\in V'} x_{ij}^k - \sum_{j\in V'} x_{n+i,j}^k),\ \forall i \in P, k \in K \tag{1j}$$

$$T_b^k + t_{s_b,n+i} - T_{n+i}^k \leq M(2 - \sum_{j\in V'} x_{bj}^k - \sum_{j\in V'} x_{n+i,j}^k),\ \forall i \in P, b \in B(i), k \in K \tag{1k}$$

$$T_i^k + t_{i,s_b} - T_b^k \leq M(2 - \sum_{j\in V'} x_{ij}^k - \sum_{j\in V'} x_{bj}^k),\ \forall i \in P, b \in B(n+i), k \in K \tag{1l}$$

$$Q_i^k + q_j - Q_j^k \leq M(1 - x_{ij}^k),\ i \in V', j \in V', k \in K \tag{1m}$$

$$e_i \leq T_i^k \leq l_i,\ \forall i \in \{0\}\cup P\cup D, k \in K \tag{1n}$$

$$\tau_b \leq T_b^k,\ \forall b \in B^p, k \in K \tag{1o}$$

$$A_b^k \leq \tau_b,\ \forall b \in B^d, k \in K \tag{1p}$$

$$\max\{0, q_i\} \leq Q_i^k \leq \min\{Q, Q + q_i\},\ \forall i \in V', k \in K \tag{1q}$$

$$x_{ij}^k \in \{0,1\},\ \forall i \in V', j \in V', k \in K \tag{1r}$$

$$w_b \in \{0,1\}, \quad \forall b \in B^r \tag{1s}$$

Objective (1a) minimizes the total shipping cost consisting of the travel cost of vehicles and the compensation of the selected bids. Constraints (1b) guarantee that each request is served exactly once. Constraints (1c) and (1d) restrict that both sender and receiver of the same request are associated with the same vehicle but cannot completely be crowdsourced if the request is not directly visited by a request bid. Constraints (1e) and (1g) ensure that vehicles start and end at the depot. Constraints (1f) define the flow conservation at the other logical vertices. Constraints (1h) and (1i) express the relationship between the arrival time A_i^k and service start time T_i^k. Constraints (1j)-(1l) assure that for each request i, the pickup operation must be performed before the delivery. Constraints (1m) guarantee the consistency of the load variables. Constraints (1n) regard the time window constraints for the depot and customers. Constraints (1o) entail that the service can only start after the release times of the pickup bids, while constraints (1p) ensure that the delivery bids cannot be selected after their due times. Constraints (1q) enforce the capacity for all vehicles. Finally, constraints (1r) and (1s) state the domains of the decision variables. Note that a large positive number M is introduced for linearization, which can be respectively set to the maximum returning time to the depot l_0 in constraints (1i)-(1l) and the vehicle capacity Q in constraints (1m), respectively.

The ABF indicates that an optimal solution to the PDPTW is a valid upper bound for the PDPCBT when the feasible solution does not include any bid. As an extension of the PDPTW, our problem is also NP-hard. However, the variables and constraints of the PDPCBT depend not only on the number of requests but also on the bid size considered in the crowdsourced system. Therefore, the PDPCBT is more complex than the PDPTW.

3.4. Route-based formulation

Although the arc-based formulation is straightforward to define the problem, its application is quite limited by the instance size since it needs to create a mass of logical arcs. To overcome the such drawback, we note that except for the selection of request bids, the other decisions are included in each vehicle route. At this end, this section reformulates the PDPCBT into a route-based formulation (RBF), where a route is divided into two parts: traveling and bid selection. The traveling depicts the arc cost when the vehicle traverses along physical arc $(i, j) \in A$. While selecting pickup and delivery bids at the facility, we handle it as a knapsack problem where the feasibility constraints (1c)-(1r) should be still satisfied. Figure 4 demonstrates the new route representation by comparing the same solution in Figure 3. This divide is consistent because bid selection can be viewed as an unordered sequence, which is different from the traveling. Therefore, it is not

necessary to define the logical arcs since the travel cost among bids at one facility is zero. Instead, we only need to consider the feasibility of the assignment and the total price of selected bids.

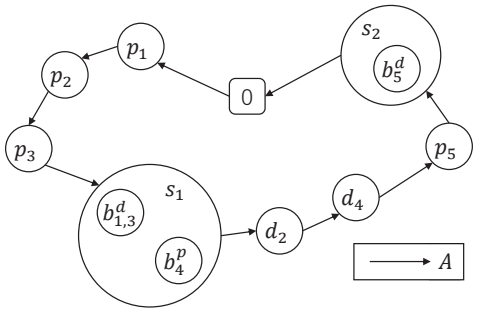


Figure 4: A same solution example for the route-based formulation.

Denote by R the set of feasible routes. Associate a binary decision variable θ_r with a route $r \in R$ indicating whether route r is selected or not. Let α_{ijr} be a binary variable specifying whether route r travels arc $(i,j) \in A$, and y_{br} be a binary variable indicating whether route r selects bid $b \in B^p \cup B^d$. Then the cost of a route r is calculated by

$$c_r = \sum_{i \in V} \sum_{j \in V} \alpha_{ijr} c_{ij} + \sum_{b \in B^p \cup B^d} y_{br} p_b. \tag{2}$$

In addition, let λ_{ir} be a binary variable that takes value 1 implying that request i is either directly visited by vehicle (cf. request 2 in Figure 4) or indirectly completed through a bid (cf. request 4 in Figure 4). The RBF is presented as follows:

$$(RBF) \ \min \ \sum_{r \in R} c_r \theta_r + \sum_{b \in B^r} p_b \omega_b \tag{3a}$$

$$\text{s.t.} \ \sum_{r \in R} \lambda_{ir} \theta_r + \sum_{b \in B^r(i)} \omega_b = 1, \ \forall i \in P \qquad (\mu_i) \tag{3b}$$

$$\sum_{r \in R} \theta_r \leq |K| \qquad (\mu_0) \tag{3c}$$

$$\theta_r \in \{0,1\}, \ \forall r \in R \tag{3d}$$

$$\omega_b \in \{0,1\}, \ \forall b \in B^r \tag{3e}$$

Objective (3a) minimizes the total shipping cost of the selected routes and bids. Constraints (3b) guarantee that every request should be visited exactly once. Constraint (3c) restricts the number of vehicles to be used, followed by constraints (3d) and (3e) that define the domain of the decision variables. Furthermore, let $\mu_i, i \in P$ be the dual variables associated with constraints (3b) and μ_0 be the dual variable associated with constraint (3c), respectively.

It is well-known that the resulting route-based formulation can provide a tighter bound than the arc-based one. However, it leads to an exponential number of variables that cannot be tackled directly by off-the-shelf solvers, like CPLEX. Therefore, we employ an exact branch-and-price-and-cut algorithm for solving, which is introduced in the next section.

4. Branch-and-price-and-cut algorithm

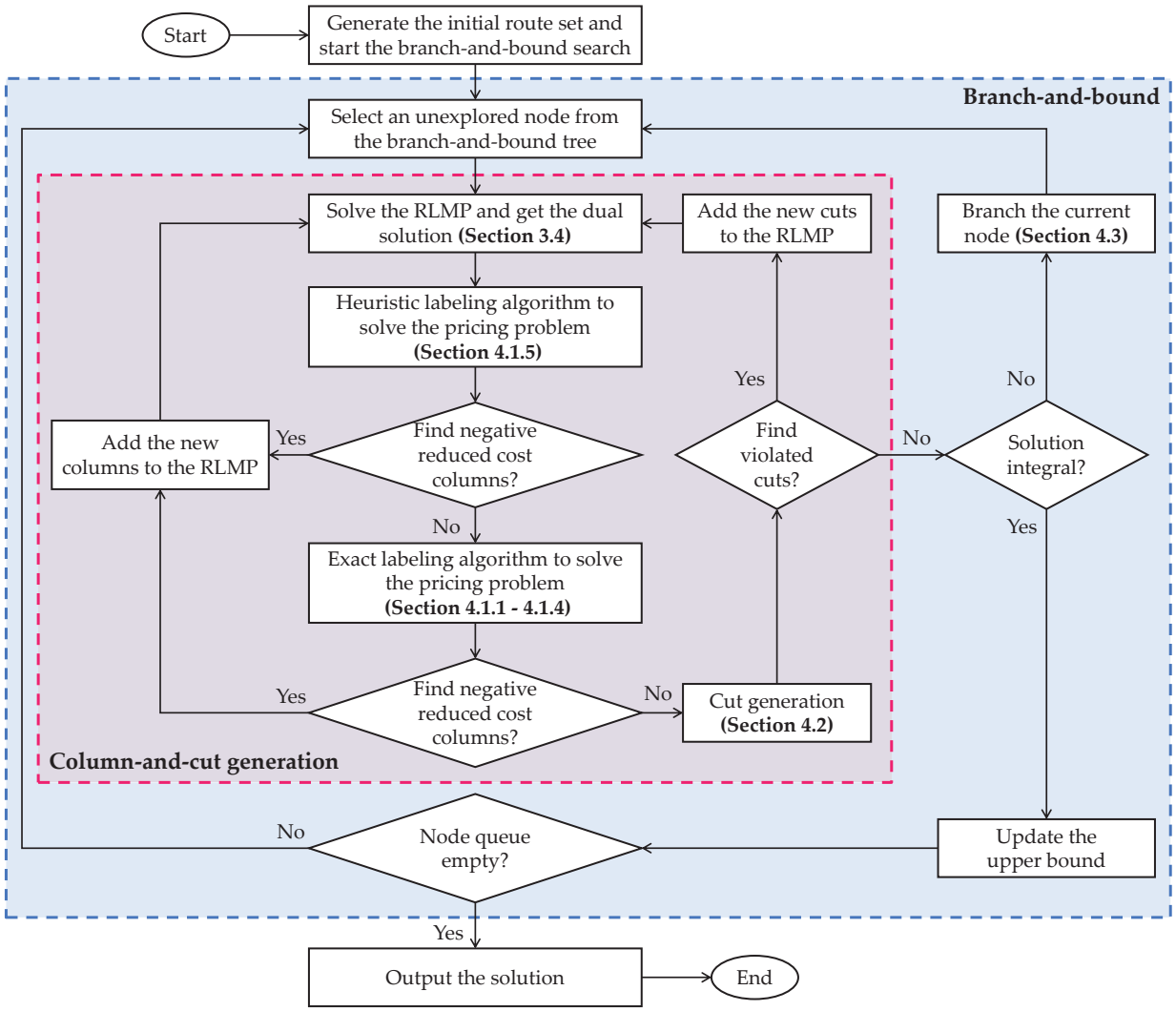


Figure 5: The flowchart of the proposed BPC algorithm.

The branch-and-price-and-cut (BPC) algorithm has been commonly used to exactly handle various vehicle routing problems (Sun et al., 2018; Yao et al., 2021; Qin et al., 2022). Before delving into the solution method, Figure 5 presents a flowchart as a big picture. The BPC algorithm solves each node of the branch-and-bound tree embedded with a column-and-cut generation. As the core of the algorithm, column generation is an iterative algorithm that solves the linear relaxation of the route-based model (3a)-(3e), which is then called the linear master problem (Desaulniers et al., 2005). It starts with a set of feasible routes generated by a simple greedy insertion as the initial set of columns, together with a dummy column that visits all requests with a high cost to guarantee the feasibility of the model. At each iteration, we optimally solve a restricted linear master problem (RLMP) that involves only a subset of route variables $R' \subseteq R$ by the simplex method, after which the dual solution is produced as inputs for the pricing problem. Afterward, we present a full-fledged labeling algorithm to solve the pricing problem for searching new columns with

negative reduced costs. To speed up the process, a heuristic pricing is first invoked, followed by an exact column generator (see Section 4.1). If such columns exist, we resolve the RLMP by adding them to R'. Otherwise, cut generation is invoked to separate violated valid inequalities to strengthen the lower bound of the RLMP (see Section 4.2). The column-and-cut generation terminates with an optimal solution to the master problem without finding any new columns or cuts to add to the RLMP. If the optimal solution is not fractional, we update the upper bound of the PDPCBT and check the terminal condition. Otherwise, four types of branching strategies are implemented to ensure that the algorithm finally yields the optimal integer solution (see Section 4.3). The details of the BPC algorithm are explained in the following.

4.1. Pricing problem

Depending on the optimal dual solution of the current RLMP, the reduced cost of a route $r \in R$ can be calculated by

$$\bar{c}_r = c_r - \sum_{i \in V} \lambda_{ir} \mu_i - \mu_0 \tag{4}$$

The pricing problem aiming at finding routes with negative reduced cost $\bar{c}_r$ can be addressed as a variant of the elementary shortest path problem with resource constraints (ESPPRC), where the arc cost d_{ij} is redefined as

$$d_{ij} = \begin{cases} c_{ij} - \frac{1}{2}\mu_i & \forall i \in P, j \in V \\ c_{ij} - \frac{1}{2}\mu_{i-n} & \forall i \in D, j \in V \\ c_{ij} & \forall i \in \{0\} \cup S, j \in V \end{cases} \tag{5}$$

Note that the dual value μ_i for fulfilling request $i \in P$ has been divided in half to balance the evaluation of visiting pickup vertex i and delivery vertex $n + i$. This is reasonable because the pairing constraints imply that the relationship between senders and receivers is one-to-one, the dual value of visiting a request can also associate with its pickup or delivery vertex. We have pointed out that determining which bids to be selected when visiting a facility is also a decision. The enumeration of all feasible bids for every visit corresponds to a kind of knapsack problem. Therefore, we define an assignment cost χ_b for each bid b:

$$\chi_b = \begin{cases} p_b - \frac{1}{2}\sum_{i \in C_b} \mu_i & \forall b \in B^p \\ p_b - \frac{1}{2}\sum_{i \in C_b} \mu_{i-n} & \forall b \in B^d \end{cases} \tag{6}$$

To this end, the pricing problem combines an elementary shortest path problem with pickup and delivery with a knapsack problem (ESPPPDKP), where the reduced cost of route r is determined by

$$\bar{c}_r = \sum_{(i,j) \in A} \alpha_{ijr} d_{ij} + \sum_{b \in B} y_{br} \chi_b - \mu_0 \tag{7}$$

Due to the NP-hardness of the pricing problem, relaxing the elementary requirements is widely used to improve its performance for various routing problems (Ropke and Cordeau, 2009; Luo et al., 2014; Yang et al., 2021). However, our pricing problem can be viewed as a special case of ESPPRC or SPPRC with precedence and pairing constraints. Both of them are also strongly NP-hard, which has been proven by Ropke and Pisinger (2007). We have tested cycle relaxation but the result is not promising (cf. Section 5.3.4). Therefore, we introduce the elementary version hereafter and propose a labeling algorithm to exactly solve it.

The labeling algorithm includes three key aspects: (i) label definition for the traveling and bid assignment; (ii) functions that extend the label from the starting depot to the returning one; and (iii) dominance rules of eliminating unpromising labels that cannot obtain a minimum-reduced-cost route for acceleration. Moreover, we integrate the algorithm with a bi-directional search where labels extend from both the forward and backward directions. This technique has been widely applied to speed up algorithm convergence (Righini and Salani, 2006; Tilk et al., 2017). To prevent generating labels twice as many as mono-directional labeling, both forward and backward extensions are limited by a specific threshold such as half of the planning horizon.

In the remainder of this subsection, we introduce the main body of the algorithm from the forward and backward directions. Subsequently, we present the merging procedure where complete routes can be formed by combining the labels from both directions. Finally, two acceleration techniques are introduced for the algorithm.

4.1.1. Forward labeling

In the forward labeling, a forward partial route from the starting depot to a physical vertex (customer or facility) i is represented by a label $L_i^f = \{\eta_i^f, c_i^f, \zeta_i^f, a_i^f, t_i^f, O_i^f, S_i^f, U_i^f\}$, where

- η_i^f: the last visited vertex, namely $\eta_i^f = i$;
- c_i^f: the reduced cost of the partial route;
- ζ_i^f: the current load of the vehicle;
- a_i^f: the earliest arrival time at the vertex i;
- t_i^f: the earliest feasible service start time at the vertex i;
- O_i^f: the set of open requests. A request is *open* when it has been started either from a pickup vertex or by a pickup bid but not completed to deliver to its receiver;
- S_i^f: the set of open requests from a facility, such that $S_i^f \subseteq O_i^f$;
- U_i^f: the set of completed requests. A request is *completed* when its sender and receiver have been served.

The elementary case of the pricing problem requires $U_i^f \cap O_i^f = \emptyset$. We also need to record the predecessor label of each label to allow backtracking for completing the route once the algorithm is done. The initial label is defined as $L_0^f = \{0, -\mu_0, 0, e_0, e_0, \emptyset, \emptyset, \emptyset\}$ at the starting depot. Given a forward label L_i^f, a new label L_j^f can be generated along a feasible arc (η_i^f, j). We discuss the feasibility conditions and derive the extension relations for L_j^f by distinguishing whether j is a pickup/delivery vertex or a facility.

Forward Case 1. If j is a pickup/delivery vertex, we can extend L_i^f to its adjacent vertex j if the following conditions are respected:

$$0 \leq \zeta_i^f + q_j \leq Q \tag{8a}$$

$$t_i^f + t_{ij} \leq l_j \tag{8b}$$

$$j \in P \ \wedge \ j \notin O_i^f \ \wedge \ j \notin U_i^f \tag{8c}$$

$$j \in D \ \wedge \ j - n \in O_i^f \tag{8d}$$

where conditions (8a) and (8b) state that the resulting extension satisfies the capacity and time window constraints, respectively. Condition (8c) implies that a sender is reachable if the corresponding request has not been open and is not completed. Condition (8d) indicates that a receiver is reachable if the corresponding request has been open.

Then the components of L_j^f can be updated by the following extension functions:

$$\begin{aligned}
\eta_j^f &= j \\
c_j^f &= c_i^f + d_{ij} \\
\zeta_j^f &= \zeta_i^f + q_j \\
a_j^f &= t_i^f + t_{ij} \\
t_j^f &= \max\{a_j^f, e_j\}
\end{aligned}$$

If $j \in P$, the components O_j^f, S_j^f and U_j^f are:

$$\begin{aligned}
O_j^f &= O_i^f \cup \{j\} \\
S_j^f &= S_i^f \\
U_j^f &= U_i^f
\end{aligned}$$

While if $j \in D$, they are performed as:

$$\begin{aligned}
O_j^f &= O_i^f \setminus \{j - n\} \\
S_j^f &= S_i^f \setminus \{j - n\} \\
U_j^f &= U_i^f \cup \{j - n\}
\end{aligned}$$

Algorithm 1 generateForwardLabels

Create an initial partial route $L_0^f = \{0, -\mu_0, 0, e_0, e_0, \emptyset, \emptyset, \emptyset\}$;
for all $v \in V$ **do**
    $UL_v^f \leftarrow \emptyset$, $EL_v^f \leftarrow \emptyset$;
end for
$UL_0^f \leftarrow \{L_0^f\}$;
while $\bigcup_{v \in V} UL_v^f \neq \emptyset$ **do**
    Select a label $L^f \in \bigcup_{v \in V} UL_v^f$ with the minimum reduced cost;
    Let u be the ending vertex of L^f, i.e., $u = \eta_L^f$;
    Move L^f from UL_u^f to EL_u^f;
    for all $i \in P \cup D$ **do**
        if extending L^f to i is feasible based on extension conditions (8a)-(8d) **then**
            Create a new label $\hat{L}_i^f$ based on extension equations in Forward Case 1;
            DOMINANCETEST($\hat{L}_i^f, UL_i^f, EL_i^f$);
        end if
    end for
    for all $s \in S$ **do**
        if $s = u$ **then**
            Continue;
        end if
        Create a dummy label $\hat{L}_s^f$;
        ASSIGNMENTEXTENSION($\hat{L}_s^f, s, UL_s^f, EL_s^f$);
    end for
end while

Forward Case 2. If j is a facility, a set of new labels which select a different number of bids would be created. In this case, we should enumerate each feasible group of bids that can be assigned to j relying on the current load and arrival time of label L_i^f. From this point of view, the pricing problem turns into a knapsack problem where we determine whether a pickup/delivery bid is selected. It can also be easily involved in the same labeling algorithm framework.

Let $B^p(j)/B^d(j)$ be the set of pickup/delivery bids associated with facility j. Denote $R(b)$ the set of served requests related to bid b. For example, suppose that delivery bid b serves receivers i_1 and i_2, then we have $R(b) = \{i_1 - n, i_2 - n\}$. Note that label L_i^f can extend to facility j if and only if there is at least one feasible bid for L_i^f. First, we create a label L_j^f as an initial dummy label

by updating the reduced cost and the times at j as

$$c_j^f = c_i^f + d_{ij}$$
$$a_j^f = t_i^f + t_{ij}$$
$$t_j^f = a_j^f$$

The other components of L_j^f remain the same as L_i^f. Then an assignment for a bid b is feasible if

$$0 \leq \zeta_j^f + q_b \leq Q \tag{9a}$$
$$b \in B^p(j) \ \wedge \ R(b) \notin O_j^f \ \wedge \ R(b) \notin U_j^f \tag{9b}$$
$$b \in B^d(j) \ \wedge \ R(b) \in O_j^f \setminus S_j^f \ \wedge \ a_j^f \leq \tau_b \tag{9c}$$

where condition (9b) states that a pickup bid is feasible if the corresponding requests have not been either started or completed. Without violating the due time of the crowdshipper, condition (9c) ensures that the delivery bid is feasible if the corresponding requests have been started but not covered by any pickup bid in the partial route.

We perform the bid selection in the sequence of the delivery before the pickup bids to break the symmetry. Let b be the bid with the minimum index among the bids in facility j, and two new labels L_1^f and L_2^f are generated corresponding to bid b is selected or not, respectively. Label L_1^f is

$$\eta_1^f = b$$
$$c_1^f = c_j^f + \chi_b$$
$$\zeta_1^f = \zeta_j^f + q_b$$
$$a_1^f = a_j^f$$

If $b \in B^p(j)$, the remaining components are updated as:

$$t_1^f = \max\{t_j^f, \tau_b\}$$
$$O_1^f = O_j^f \cup \{R(b)\}$$
$$S_1^f = S_j^f \cup \{R(b)\}$$
$$U_1^f = U_j^f$$

If $b \in B^d(j)$, they are set as:

$$t_1^f = t_j^f$$
$$O_1^f = O_j^f \setminus \{R(b)\}$$
$$S_1^f = S_j^f$$
$$U_1^f = U_j^f \cup \{R(b)\}$$

Algorithm 2 dominanceTest($\hat{L}^f, UL_i^f, EL_i^f$)

1: **if** $\hat{L}^f$ is dominated by any label in $UL_i^f \cup EL_i^f$ **then**
2: **return** ;
3: **else**
4: Add label $\hat{L}^f$ to UL_i^f;
5: Discard any label in $UL_i^f \cup EL_i^f$ dominated by $\hat{L}^f$;
6: **end if**

While label L_2^f where $\eta_2^f = b$ is generated by inheriting the other components of label L_j^f.

Whenever a new label is created and stored, we apply the following *forward dominance rules* to reduce the number of unpromising labels: A forward label L_1^f dominates another label L_2^f associated with the same end vertex if

$$t_1^f \leq t_2^f \tag{10a}$$
$$O_1^f = O_2^f \tag{10b}$$
$$S_1^f \subseteq S_2^f \tag{10c}$$
$$U_1^f \subseteq U_2^f \tag{10d}$$
$$c_1^f \leq c_2^f \tag{10e}$$

Proposition 1. Conditions (10a)-(10e) are valid domination that guarantee the optimality of the pricing problem.

Proof. The proof follows that every feasible extension of L_2^f is also feasible for L_1^f, and the reduced cost of the complete route generated by extending L_1^f is not worse than that generated by extending L_2^f. Denote a feasible partial route p that can extend the partial route of L_2^f to the returning depot. If there is no such route, L_2^f can be discarded. Let $r_1 = (L_1^f, p)$ and $r_2 = (L_2^f, p)$ be the complete routes generated by extending L_1^f and L_2^f with p, respectively. Since r_2 is feasible, conditions (10a)-(10d) ensure that r_1 is also feasible for the time window, capacity, pairing, and elementary constraints. Note that it is redundant to include the current load ζ in the dominance rules due to condition (10b). Moreover, the relationship between the reduced costs is confirmed by condition (10e). Therefore, L_1^f can dominate L_2^f without losing the optimality of the pricing problem. □.

The forward labeling procedure is described in Algorithms 1-3, where UL_v^f (respectively, EL_v^f) represents the set of labels that ends at vertex v and have not (respectively, have) been extended. Algorithm 2 uses the criteria (10a)-(10e) to test the dominance relationship between labels.

Algorithm 3 assignmentExtension($\hat{L}_s^f, s, UL_s^f, EL_s^f$)

1: Initialize sets $TL_b \leftarrow \emptyset, b = 0, 1, \ldots, |s|$, where $\{0\}$ is a dummy index and $\{1, \ldots, |s|\}$ is the indices of bids in s;
2: $TL_0 \leftarrow \{\hat{L}_s^f\}$;
3: **for all** $b = 0, 1, \ldots, |s|$ **do**
4: Delete dominated labels in TL_b according to the forward dominance rules;
5: **for all** $L \in TL_b$ **do**
6: $i \leftarrow$ the minimum index among the feasible bids;
7: **if** $i = \emptyset$ and L has selected at least one bid **then**
8: DOMINANCETEST(L, UL_s^f, EL_s^f);
9: **else if** $i \neq \emptyset$ **then**
10: Create two new labels L_1^f and L_2^f based on extension equations in Forward Case 2;
11: **end if**
12: **end for**
13: **end for**

We make a detailed comment on Algorithm 3, which employs a novel labeling algorithm to deal with the knapsack problem. At the beginning of the algorithm, we first initialize temporary sets $TL_b, b = 0, 1, \ldots, |s|$ to store possible bid selection at each bid state b and add the dummy label to the set TL_0. Note that when the label finishes the complete assignment at Step 8, Algorithm 2 is invoked to test the generated labels. For this reason, we observe that at Step 4, it may be time-consuming to check the dominance relationship among all labels in each set TL_b. Therefore, we use a greedy strategy where only the pair of consecutive labels invokes the dominance test. According to our preliminary experiments, it is computationally convenient, although not all dominated labels are discarded.

Table 3 gives an illustrative example of the extension procedure in Algorithm 3 without considering the time window and capacity constraints. Given an unextended label L_1^f associated with a partial route $(0 - p_4 - p_1)$ and assume that it extends to facility s that has four bids $\{b_3^d, b_4^d, b_1^p, b_3^p\}$. As shown in the table, the algorithm first creates a dummy label L_s^f and adds it to set TL_0 for initialization. Then the bid selection occurs in order. Obviously, only bids b_4^d and b_3^p can be selected. Hence, sets $TL_{b_3^d}$ and $TL_{b_1^p}$ remain empty. While sets $TL_{b_4^d}$ and $TL_{b_3^p}$ have respectively two and four labels due to Step 10. After the dominance test, set UL_s^f has at most three labels since one partial route makes no bid selection during the extension.

Table 3: An illustrative example of the extension procedure in Algorithm 3.

Set	TL_0	$TL_{b_3^d}$	$TL_{b_4^d}$	$TL_{b_1^p}$	$TL_{b_3^p}$	UL_s^f
Routes	$0-p_4-p_1$	$\emptyset$	$0-p_4-p_1-(b_4^d)$	$\emptyset$	$0-p_4-p_1-(b_4^d,b_3^p)$	$0-p_4-p_1-(b_4^d,b_3^p)$
			$0-p_4-p_1$		$0-p_4-p_1-(b_4^d)$	$0-p_4-p_1-(b_4^d)$
					$0-p_4-p_1-(b_3^p)$	$0-p_4-p_1-(b_3^p)$
					$0-p_4-p_1$	

4.1.2. *Backward labeling*

Opposite to the forward extension, a backward label $L_i^b = \{\eta_i^b, c_i^b, \zeta_i^b, a_i^b, t_i^b, O_i^b, S_i^b, U_i^b\}$ is associated with a partial route that extends reversely from the returning depot and ends at a vertex i. The meanings of components of the label L_i^b are analogous to the forward label, except that:

- a_i^b: the latest departure time at the vertex i;
- t_i^b: the latest feasible service start time at the vertex i;
- O_i^b: the set of finished requests. A request is *finished* when it has been delivered either to a delivery vertex or by a delivery bid but not started from its sender;
- S_i^b: the set of finished requests from a facility, such that $S_i^b \subseteq O_i^b$;

The initial backward label is created at the returning depot as $L_0^b = \{0, 0, 0, l_0, l_0, \emptyset, \emptyset, \emptyset\}$. We can extend a label L_i^b along the arc (j, η_i^b) to obtain a new backward label L_j^b according to the following two cases:

Backward Case 1. If j is a pickup/delivery vertex, it is feasible if

$$-Q \leq \zeta_i^b + q_j \leq 0 \tag{11a}$$

$$t_i^b - t_{ji} \geq e_j \tag{11b}$$

$$j \in P \ \wedge \ j \in O_i^b \tag{11c}$$

$$j \in D \ \wedge \ j - n \notin O_i^b \ \wedge \ j - n \notin U_i^b \tag{11d}$$

where conditions (11a) and (11b) are compatible with the capacity and time windows, respectively. Condition (11c) indicates that a sender cannot be served unless its corresponding request has been finished. Finally, condition (11d) ensures that a receiver is reachable if the corresponding request has not been finished or completed. Then the components of L_j^b can be updated by the following extension functions:

$$\eta_j^b = j$$
$$c_j^b = c_i^b + d_{ji}$$

$$\zeta_j^b = \zeta_i^b + q_j$$
$$a_j^b = t_i^b - t_{ji}$$
$$t_j^b = \min\{a_j^b, l_j\}$$

If $j \in P$, the components O_j^b, S_j^b and U_j^b are:

$$O_j^b = O_i^b \setminus \{j\}$$
$$S_j^b = S_i^b \setminus \{j\}$$
$$U_j^b = U_i^b \cup \{j\}$$

While if $j \in D$, they are performed as:

$$O_j^b = O_i^b \cup \{j - n\}$$
$$S_j^b = S_i^b$$
$$U_j^b = U_i^b$$

Backward Case 2. If j is a facility, the reduced cost and times of the dummy label L_j^b are updated as:

$$c_j^b = c_i^b + d_{ji}$$
$$a_j^b = t_i^b - t_{ji}$$
$$t_j^b = a_j^b$$

We ensure that a bid b is feasible if the following conditions are satisfied:

$$-Q \leq \zeta_j^b + q_b \leq 0 \tag{12a}$$
$$b \in B^p(j) \ \wedge \ R(b) \in O_j^b \setminus S_j^b \ \wedge \ t_j^b \geq \tau_b \tag{12b}$$
$$b \in B^d(j) \ \wedge \ R(b) \notin O_j^b \ \wedge \ R(b) \notin U_j^b \tag{12c}$$

where condition (12b) states that a feasible pickup bid should regard its release time, and the corresponding requests have been delivered to their receivers but not covered by any delivery bid in the partial route. Condition (12c) ensures that a delivery bid is feasible if the corresponding requests have not been either finished or completed.

In contrast to the forward direction, we first assign pickup bids before delivery bids. Let b be the bid with the maximum index among the bids in this facility. Two new labels L_1^b and L_2^b are generated corresponding to whether bid b is selected or not, respectively. Label L_1^b is

$$\eta_1^b = b$$

$$
\begin{aligned}
c_1^b &= c_j^b + \chi_b \\
\zeta_1^b &= \zeta_j^b + q_b \\
a_1^b &= a_j^b
\end{aligned}
$$

If $b \in B^p(j)$, the remaining components are updated as:

$$
\begin{aligned}
t_1^b &= t_j^b \\
O_1^b &= O_j^b \setminus \{R(b)\} \\
S_1^b &= S_j^b \\
U_1^b &= U_j^b \cup \{R(b)\}
\end{aligned}
$$

If $b \in B^d(j)$, they are set as:

$$
\begin{aligned}
t_1^b &= \min\{t_j^b, \tau_b\} \\
O_1^b &= O_j^b \cup \{R(b)\} \\
S_1^b &= S_j^b \cup \{R(b)\} \\
U_1^b &= U_j^b
\end{aligned}
$$

While label L_2^b where $\eta_2^b = b$ is generated by inheriting the other components of label L_j^b.

The dominated backward labels can be discarded by applying the following *backward dominance rules*: A label L_1^b dominates another label L_2^b associated with the same end vertex if (10b)-(10e) are satisfied, together with condition

$$
t_1^b \geq t_2^b \tag{13}
$$

We omit the proof of validity for the backward domination since its similarity to that of the forward one.

4.1.3. *Merging forward and backward labels*

When labels from both directions are created, feasible complete routes with negative reduced costs should be identified by merging forward and backward labels. For a forward label L_i^f and a backward label L_i^b that propagate to the same vertex i, they can be merged if

$$
t_i^f \leq t_i^b \tag{14a}
$$

$$
\begin{cases}
O_i^f \setminus \{i\} = O_i^b & i \in P \\
O_i^f = O_i^b \setminus \{i - n\} & i \in D \\
O_i^f \setminus R(\bar{B}^p(i)) = O_i^b \setminus R(\bar{B}^d(i)) & i \in S \ \wedge \ \bar{B}^f(i) = \bar{B}^b(i)
\end{cases} \tag{14b}
$$

$$S_i^f \cap S_i^b = \emptyset \tag{14c}$$

$$U_i^f \cap U_i^b = \emptyset \tag{14d}$$

where $R(\bar{B}^p(i))/R(\bar{B}^d(i))$ denotes the set of served requests related to the selected pickup/delivery bids in this visit to facility i. In this case, two opposite labels can be merged only if they choose the same group of bids, i.e., $\bar{B}^f(i) = \bar{B}^b(i)$. Condition (14a) ensures the time feasibility of the resulting route. Condition (14b) respects that all the requests are not completed yet, while condition (14c) indicates that a request cannot be served by two bids. Condition (14d) guarantees that each request is visited no more than once.

Relying on the merge point i, the reduced cost $\bar{c}_r$ of the complete route r in the concatenation is calculated by

$$\bar{c}_r = \begin{cases} c_i^f + c_i^b & i \in P \cup D \\ c_i^f + c_i^b - \sum\limits_{b \in \bar{B}^f(i)} \chi_b & i \in S \end{cases} \tag{15}$$

where the additional expression in the case $i \in S$ derives from the fact that the assignment cost of all the selected bids in this visit has been accumulated twice from both directions. Hence, we correct the calculation by subtracting them once, which is then consistent with Equation (7).

4.1.4. Label elimination

The bi-directional labeling algorithm can be accelerated by eliminating more useless labels that cannot be extended to the returning depot. We observe that all open requests of a forward label L^f should be delivered to their receivers or assigned to corresponding delivery bids by respecting the time window requirements. Hence, L^f can be safely eliminated if there exists at least one open request that cannot be delivered on time. To determine whether such a situation happens, we must consider all open requests and solve a variant of the TSPTW. In practice, we can only test for subsets of open requests with a cardinality of no more than two. In other words, a forward label L^f can be eliminated if it cannot be merged with any of the backward labels that only visit one or two corresponding finished requests. Note that all these necessary feasible backward labels can be generated in advance because they never change during checking. Similarly, the forward labels that cover only one or two senders can also be generated beforehand to eliminate unpromising labels at each iteration of the backward extension.

4.1.5. Heuristic pricing

Even with the aforementioned enhancements, the exact method can be quite time-consuming. To avoid calling it as much as possible, Desaulniers et al. (2008) claimed that it is not necessary to solve the pricing problem optimally in the early iterations of the column generation. To speed up the labeling algorithm, we therefore apply a heuristic version of exact labeling based on a

reduced network. Specifically, for each vertex $i \in P \cup D$, we only consider a limited number of the "shortest" outgoing arcs that at most five arcs are connected to the pickup vertices and another five to the delivery. We use the criterion of minimizing the arc cost (5) to define the "shortest". Also, we retain all arcs $(0,i),(i,n+i),(n+i,0),\forall i \in P$ to keep the network feasible. We do not truncate the arcs connected with the facilities. When dominance checking at vertex i in both directions, we further relax condition (10b) of comparing the open requests in sets O_i, which can accelerate the pricing by eliminating more labels but may lose the optimal label. Consequently, the exact pricing must be performed to derive a valid solution when the heuristic column generator fails to find any negative reduced cost column.

4.2. Cut generation

The subset row cuts (SRCs) can be employed as the valid inequalities to tighten the lower bound of each branch-and-bound node (Jepsen et al., 2008). Given a subset $P' \subseteq P$, the SRCs are defined as:

$$\sum_{r \in R} \gamma_{P'}^{r} \theta_r \leq 1, \forall P' \subseteq P, |P'| = 3 \tag{16}$$

where $\gamma_{P'}^{r}$ takes the value 1 if route r visits at least two senders in P' and 0 otherwise. Each SRC implies that the solution only can involve at most one of the routes that visit more than one sender in the subset P'.

Note that the SRCs may impede the efficiency of the labeling algorithm due to the complexity of handling dual values of these cuts. Please refer to the work of Li et al. (2020) for details. To alleviate this negative impact, we implement a heuristic enumeration proposed by Desaulniers et al. (2008) to seek violated inequalities.

4.3. Branching strategies

We explore the branch-and-bound tree in a best-first fashion in which the node with the lowest lower bound is examined first. By considering the efficiency and tractability, we use the following four hierarchical branching strategies to derive integrality if the optimal solution $(\bar{\theta}_r, \bar{\omega}_b)(r \in R, b \in B^r)$ of the current RLMP is fractional.

The first branching strategy is applied to the number of vehicles. Supposed that the number of vehicles $\bar{m} = \sum_{r \in R} \bar{\theta}_r$ is fractional in the optimal solution, then we generate two children nodes by respectively forcing $\sum_{r \in R} \theta_r \leq \lfloor \bar{m} \rfloor$ and $\sum_{r \in R} \theta_r \geq \lceil \bar{m} \rceil$. These two constraints can be directly incorporated into the RLMP by modifying constraint (3c).

The second strategy is branching on request bid b whose value $\bar{\omega}_b$ is fractional. One children node is created by forcing $\bar{\omega}_b = 0$, namely, request bid b cannot be selected in the solution. The other node enforces $\bar{\omega}_b = 1$, which can be done by removing all pickup and delivery bids related

to the corresponding request and all arcs connected to the corresponding sender and receiver from the pricing problem.

The third branching strategy bases on a vertex set T whose size is limited to two. Denote $x(T)$ the inflow of the vertex set T. Due to the bid assignment in a vehicle route, we generalize the calculation of the inflow as

$$x(T) = \sum_{r \in R} \left(\sum_{i \notin T} \sum_{j \in T} \beta_{ij}^r + \sum_{b \in B(T)} y_{br} \right) \bar{\theta}_r \tag{17}$$

where $B(T)$ includes the bids that serve the customers in set T. Note that the inflow of a facility must be integral if all assignment values of bids in this facility are integer. Therefore, we can only impose $T \subseteq P \cup D$. Then it creates two branches $x(T) \leq 1$ and $x(T) \geq 2$, respectively. The dual values of these two constraints can be easily involved in the arc cost (5), after which the children nodes can be solved by applying the same labeling algorithm.

Finally, if all inflows are integer, there may exist a pair of bids (b_1, b_2) such that $\delta_{b_1,b_2} = \sum_{r \in R} y_{b_1}^r y_{b_2}^r \bar{\theta}_r$ takes a fractional value, where δ_{b_1,b_2} is a binary variable indicates whether route r selects both bids b_1 and b_2. Then we generate two branches by forcing $\delta_{b_1,b_2} = 0$ and $\delta_{b_1,b_2} = 1$. The former branch can be implemented by deleting all routes that select both of them in the current RLMP and enforcing the newly generated routes to select at most one of them. While for the latter, we can remove all routes that cover only one of them from the current RLMP and impose b_1 and b_2 to be assigned to the same route. Meanwhile, we do not extend to the related customers of bids b_1 and b_2, and other bids associated with these two requests.

The performance of the branching strategies can be improved by using strong branching. That is, for each branching decision, we first choose ten candidates whose fractional part is closest to 0.5. Then the algorithm investigates the lower bounds for both two children nodes of the candidate by solving the associated relaxations of the RLMP with the heuristic pricing described in Section 4.1.5. Finally, we branch the candidate that maximizes the lower bound in the weakest of the children nodes.

5. Computational study

5.1. Test instances

The algorithm is evaluated based on the set of PDPTW request instances introduced by Ropke and Cordeau (2009). It contains four problem classes, namely AA, BB, CC, and DD, each of which has 10 instances with different numbers of requests between 30 and 75 in increments of 5. Each customer i has a length W of the time window and the demand q_i is randomly chosen in $[5, Q]$, where Q is the vehicle capacity. The values of W and Q of these groups are given in Table 4. All customers are randomly generated within a square of $[0, 50] \times [0, 50]$ according to a uniform

distribution for each problem class. The depot is located at the center of this square, namely (25.0, 25.0). We also consider four transshipment facilities in a square topology whose coordinates are respectively (12.5,12.5), (12.5, 37.5), (37.5, 37.5), and (37.5, 12.5) in all instances.

Table 4: Values of W and Q.

Class	W	Q
AA	60	15
BB	60	20
CC	120	15
DD	120	20

Table 5: Combinations of PDB and RB.

PDB \ RB	0.1n (H)	0.2n (F)
0.6n (H)	HH	HF
1.2n (F)	FH	FF

The test instances are completed by generating the bid information into the request instances. We first specify the total number of pickup and delivery bids (PDB) and request bids (RB) for each instance. As stated by Kafle et al. (2017), a bid with pickup and delivery operations could bring more instability to the crowdsourced system. Therefore, we limit the number of submitted RBs. Moreover, the complexity of the algorithm heavily depends on the size of the PDB rather than the RB since there is no need to determine the latter in the pricing procedure. For the practicality and tractability of the problem, the number of PDB and RB is set to be 1.2n and 0.2n, respectively, where n is the number of requests to be served. For each instance, we further consider different combinations of the full (F) and half (H) sizes of PDB and RB, as shown in Table 5.

To simulate the bidding process, the information of each crowdshipper and attributes of each bid are randomly defined. A location is randomly chosen from the same area as the origin of a crowdshipper and the maximum capacity equals the vehicle capacity, ensuring that all customers could be crowdsourced in the instance. To meet the assumption, the considered available time expresses the earliest and the latest, respectively, to distinguish pickup and delivery bids. Each bid contains: (1) at most two random customers and a facility within a radius of five kilometers from their origins; (2) a maximum load to be carried; (3) a cost-minimum route by solving a predefined TSPTW; (4) a bid price determined by the travel cost of the route and a compensation coefficient ρ that is randomly chosen from a uniform distribution $[0.5, 1.0]$; (5) the release/due time obtaining from the consequent route.

The number of available vehicles is the same as that in the best solution to the request instances found by Baldacci et al. (2011). In total, there are 4 (problem classes) $\times$ 10 (instances) $\times$ 4 (bid sizes) = 160 instances. The instance name is subject to the format of *Class-n-Size-K*, where *Class* represents the name of a problem class, n implies the number of requests that should be satisfied, *Size* indicates the number of involved bids, and K corresponds to the number of vehicles. For example, BB50FH4 is an instance with 50 requests in problem class BB, containing 60 PDBs (1.2n) and 5 RBs (0.1n), and 4 vehicles for transportation.

5.2. Experimental setup

We conducted the experiments on an Intel Core i7-4720 with a 2.60 GHz CPU and 8G RAM running the Windows 10 operating system. The BPC algorithm was coded in Java with CPLEX 12.6 to solve the RLMP. We limit the maximum running time to 7,200 seconds on each instance. The route-based formulation and the pricing problem ESPPPDKP are solved by the BPC algorithm based on the bi-directional labeling algorithm, equipped with label elimination, heuristic pricing, subset row cuts, and strong branching.

To limit the number of tables, we only report the aggregated results of four problem classes. The detailed computational results are provided in Appendix A. However, the average values may not reflect the consequences accurately since there are 40 instances in each class. Therefore, we separate each group of instances equally into two sets where the first 20 instances with $n \leq 50$ are denoted as *Class-1*, and the last 20 ones with $n > 50$ are denoted as *Class-2*.

5.3. Overall performance results

This section concerns the following four aspects of the experiments to assess the efficiency of the formulation and algorithm. First, we evaluate the lower bound of the root node and measure the overall performance under the impact of the SRCs. Second, we examine the impact of the proposed acceleration techniques on the labeling algorithm at the root node. Third, we investigate the impact of the strong branching. Finally, we compare the results between the arc-based and route-based formulations.

5.3.1. The impact of the SRCs

Table 6: Impact of the SRCs on the relaxation of the route-based formulation.

	Without SRCs				With SRCs				
Set	Opt	Gap (%)	Nodes	Time	Opt	Gap (%)	Cuts	Nodes	Time
AA-1	20	0.36	8	72.0	20	0.02	61	2	31.1
AA-2	20	0.27	20	971.5	20	0.07	112	4	251.9
BB-1	20	0.17	7	56.7	20	0.05	71	2	31.5
BB-2	20	0.24	28	579.9	20	0.10	90	6	314.6
CC-1	20	0.33	20	207.0	20	0.09	59	3	119.0
CC-2	15	0.41	51	2827.8	17	0.23	84	13	2377.1
DD-1	16	0.62	70	1546.9	17	0.08	81	3	1508.4
DD-2	4	0.88	85	6676.1	4	0.38	119	9	6581.4
Average	16.9	0.41	36.1	1617.24	17.3	0.13	84.5	5.3	1401.88

We first report the impact of the SRCs on the relaxation of the route-based formulation. The results are summarized in Table 6, in which the column "Set" indicates the name of an instance set. Note that the comparative results are distinguished under the blocks "Without SRCs" and

"With SRCs", respectively. Columns "Opt" report the number of instances that can be solved to optimality. Columns "Gap(%)" present the integrity gap between the upper bound obtained by our BPC algorithm (UB) and the corresponding optimal value of the linear relaxation (LB) in percentage, which is thus computed as $100 \times (UB - LB)/UB$. The column "SRCs" records the number of SRCs added to the root node in total. Columns "Nodes" and "Time" give the number of branch-and-bound nodes explored by our BPC algorithm and the total computational time in seconds, respectively. In the last row, we also report the average values of the corresponding columns.

According to Table 6, we observe that the relaxation of the route-based formulation can provide a tight lower bound for the PDPCBT since the integrality gap without SRCs is quite minor, only 0.41% on average. After introducing valid inequalities, the integrality gap is then significantly improved up to 0.13%, which ensures that the SRCs can lift the lower bound and considerably close the gap. Furthermore, the SRCs also have a great impact on overall performance. On the one hand, when they are applied to the algorithm, there are three more instances that can solve to optimality within the time limit by comparing without SRCs. In total, the BPC algorithm exactly solved 138 out of 160 instances. On the other hand, noticeably fewer nodes and less computational time show the fast convergence brought by SRCs. Finally, as revealed in columns "Opt" and "Time", sets *Class-2* are harder to solve than *Class-1*. This observation makes sense because the former has more requests and available bids to be considered, which can lead to more difficult problem structures.

A detailed look into the instance-level results (cf. Appendix A) shows that instances with full RB size tend to be no more complex and consume less time to solve than those with half RB size. This is mostly because more request bids for selection means fewer requests for the pricing problem to be determined, which accelerates route generation. Instances with full PDB size generally show to be harder to solve than those with half PDB size as there are more candidates during the assignment in the labeling algorithm. Finally, it should be emphasized that although there are two instances in Class CC and five instances in Class DD that cannot be solved optimally, the algorithm can still output a feasible upper bound for them.

5.3.2. *The impact of the acceleration techniques*

As aforementioned, we have embedded the bi-directional search, label elimination, and heuristic pricing to accelerate the labeling algorithm. Four types of column generation procedures are executed at the root node of each instance without any valid inequalities to evaluate their performance. They are distinguished as follows: AnB – only without bi-directional search, AnL – only without label elimination, AnH – only without heuristic pricing, and All – with all

Table 7: Impact of the acceleration techniques on the root node.

	AnB		*AnL*		*AnH*		*All*	
Set	T_{LB}	Labels	T_{LB}	Labels	T_{LB}	Labels	T_{LB}	Labels
AA-1	10.5	529	10.2	1063	9.2	568	10.1	579
AA-2	85.6	1152	96.0	3105	79.0	1288	79.9	1301
BB-1	17.1	636	11.2	2515	6.1	664	7.1	661
BB-2	56.8	1455	93.6	6158	53.1	1680	50.0	1668
CC-1	14.4	1424	14.0	1966	13.4	1515	10.9	1507
CC-2	173.4	6411	174.4	9055	239.7	6642	143.0	6704
DD-1	48.6	3062	48.3	7204	44.4	3227	30.1	3216
DD-2	432.1	10918	588.7	23543	564.5	10945	229.5	10977
Average	104.82	3198.3	129.55	6826.1	126.17	3316.1	70.08	3326.6

acceleration techniques. Table 7 summarizes the results, in which columns "T_{LB}" give the time in seconds to obtain the optimal value at the root node, and columns "Labels" report the total number of labels generated in the last column generation iteration.

As can be seen from Table 7, the average times in columns T_{LB} demonstrate the importance of applying the acceleration techniques in the labeling algorithm. Specifically, compared with type *All*, the computational time of type *AnB* is approximately 1.5 times longer when the bi-directional search is not applied. Besides, when comparing values in columns "Labels" that reflect the tractability of instance sets, type *AnL* shows that the label elimination can decrease the computational effort due to the significant reduction of generated labels. Finally, types *AnH* and *All* differ little in *Class-1*, which indicates the effectiveness of the pure exact labeling algorithm. However, the computational time of the latter certainly takes less in *Class-2*, and thus shows the superiority of the heuristic pricing. In conclusion, label elimination and heuristic pricing perform better than the bi-directional search.

5.3.3. The impact of the strong branching

Table 8: Impact of the strong branching on particular instances.

		Without Strong Branching				With Strong Branching			
Class	Num	Opt	TB	Nodes	Time	Opt	TB	Nodes	Time
AA	17	17	0.7	7	358.7	17	17.7	5	294.2
BB	19	19	0.7	16	481.7	19	18.0	8	340.3
CC	17	14	3.1	28	3169.6	15	88.5	17	2446.9
DD	11	3	2.5	14	5887.3	6	75.9	13	5435.1
Average	16.0	13.3	1.74	16.1	2474.32	14.3	50.01	10.8	2129.13

This section measures the performance of the algorithm with or without the strong branching. Note that we do not consider instances that cannot obtain a feasible upper bound since they could

not be solved either without the strong branching. Moreover, we restrict the comparison only to the particular instances in which the algorithm cannot derive integrality at the root node. The results are provided in Table 8. Two new columns "Num" and "T_B" report the number of the particular instances and the average running time spent in the branching procedure, respectively.

As shown in Table 8, incorporating the strong branching helps the algorithm achieve better performance. Despite that it spent more time on exploring branch candidates, strong branching can solve four more instances to optimality. Instead, the algorithm without strong branching explored 1.5 times more branch-and-bound nodes on average than that with strong branching, which accordingly consumed more time to solve.

5.3.4. Arc-based formulation vs. route-based formulation

Table 9: Comparison between arc-based formulation and route-based formulation

	ABF			RBF1			RBF2		
Set	Opt	Gap(%)	Time	Opt	Gap(%)	Time	Opt	Gap(%)	Time
AA-1	1	21.96	7160.3	20	0.02	31.1	19	0.02	377.0
AA-2	0	33.90	7200.0	20	0.07	251.9	15	0.08	2129.2
BB-1	3	19.68	6715.5	20	0.05	31.5	20	0.05	60.1
BB-2	0	39.89	7200.0	20	0.10	314.6	20	0.10	629.1
CC-1	0	27.56	7200.0	20	0.09	119.0	18	0.09	854.0
CC-2	0	41.13	7200.0	17	0.23	2377.1	13	0.25	3275.6
DD-1	0	33.96	7200.0	17	0.08	1508.4	15	0.16	2815.0
DD-2	0	45.83	7200.0	4	0.38	6581.4	2	0.57	6755.4
Average	0.5	32.99	7134.48	17.3	0.13	1401.88	15.3	0.17	2111.93

Finally, we present the comparison between the arc-based formulation (ABF) and the route-based formulation (RBF). Moreover, we consider both the elementary (RBF1) and non-elementary (RBF2) versions of RBF. In the non-elementary route-based formulation, the set of feasible routes R expands to allow routes that have cycles. However, we impose two conditions that are inspired by Ropke and Cordeau (2009) on each route: (i) it can visit the same sender or pickup bid again only after the corresponding request has been completed, and (ii) it cannot visit a receiver or delivery bid until the corresponding request has been open. We utilize CPLEX to solve the linear relaxation of the ABF and the same BPC algorithm to solve the RBF2. Notably, the ABF has been strengthened with cuts automatically introduced by CPLEX and the RBF2 has also been enhanced with SRCs.

Table 9 presents the number of instances that are solved to optimality, together with the integrality gap obtained by solving linear relaxations at the root node and the overall computational time in seconds. As observed from the columns "Gap(%)", we can conclude that the two linear

relaxations of the route-based formulation considerably outperform that of the arc-based formulation. The lower bound of the ABF is relatively weak, and only four instances could be solved in a limited time of two hours. This is mainly because the increasing problem size makes the arc-based formulation more complex and less tractable.

With SRCs on the relaxations, the difference between the RBF1 and RBF2 is rather small. However, in terms of both the efficiency and the effectiveness, the non-elementary route-based formulation is inferior to the elementary one. The reason behind this fact is that the non-elementary routes would visit more customers and select more bids, especially in the problem classes CC and DD that have a longer length of time windows. In other words, this can lead to higher complexity for dominating labels and identifying more active SRCs for the RBF2. Therefore, the RBF2 optimally solves fewer instances and takes a longer time than the RBF1.

5.4. Sensitivity analysis and managerial findings

Based on the results presented in Section 5.3, we shed light on some interesting insights and conduct sensitivity analysis on the proposed problem in this section.

5.4.1. Analysis on the time windows

First, we analyze the performance on the different lengths of the time window. Typically, the wider the time windows, the more computational time is needed for handling the instances. This conclusion can be proved from the column "Time" in Table 6 and the column "T_{LB}" in Table 7. This is mainly because the pricing problem becomes more complex since the labeling algorithm requires processing more labels, which thereby increases the computational effort in each branch-and-bound node. Meanwhile, although there is not much difference in the integrality gap (i.e., column "Gap(%)"), the tighter the time windows are, the stronger the lower bound can be yielded. Therefore, AA and BB result in a shorter average computational time to close the gap.

5.4.2. Analysis on the vehicle capacity

Next, we examine the impact of the vehicle capacity on the algorithm. As mentioned earlier, vehicles in classes AA and CC have smaller capacities than those in BB and DD. However, the capacity shows a trivial effect on the problem. By comparing the obtained results in Table 6, we observe that instances with smaller capacities do not mean more tractable. Also, their integrality gaps do not result in tighter than the larger ones, which may consume longer computational times.

5.4.3. Comparison with truck-only transportation

Since we utilize the request instances introduced by Ropke and Cordeau (2009) in which the benchmark results have been provided and later Baldacci et al. (2011) closed the unsolved in-

Table 10: Solution analysis by comparing between PDPCBT and PDPTW benchmarks.

Set	Num	PDPTW	PDPCBT	Saving(%)	B_{avail}	B_{sel}	Rate(%)
AA-1	20	1493.9	1423.8	4.71	41.9	7.0	17.59
AA-2	20	2242.5	2171.4	3.19	68.1	10.0	15.12
BB-1	20	1423.4	1390.4	2.30	41.9	5.1	12.09
BB-2	20	2604.7	2512.8	3.37	68.1	10.2	14.81
CC-1	20	1374.2	1345.7	2.16	41.9	5.1	12.91
CC-2	17	2114.3	2074.3	1.90	67.9	7.4	11.28
DD-1	17	1329.1	1299.5	2.22	39.1	5.2	13.26
DD-2	4	1834.3	1799.0	1.90	51.3	7.3	14.93
Average	17.3	1802.06	1752.10	2.72	52.52	7.13	14.00

stances. Hence, the previous results that equate to truck-only transportation can serve as a reference to analyze the potential benefit of crowdshipping. We drop off our instances that cannot obtain an optimal solution by the proposed BPC algorithm and give the comparisons in Table 10. Column "Num" indicates the number of instances in each set. Columns "PDPTW" and "PDPCBT" show the average optimal value found from the literature and our BPC algorithm, respectively. Column "Saving(%)" denotes the percentage ratio of cost savings by bid selection. Columns "B_{avail}" and "B_{sel}" present the average numbers of available and selected bids, respectively. Finally, the column "Rate(%)" denotes the percentage ratio of bid usage in decision-making.

As expected, incorporating crowdshipping can lead to an average drop of 2.72% in the total shipping cost. However, this reduction and the available bids are not positively related. For example, the average available bids in sets AA-1 and CC-1 take the same value. But AA-1 generated the most savings while CC-1 contributed only 2.16%. It illustrates that no matter how many bids are available in the crowdsourced system, only by making proper operational decisions can the company maximize the benefits of the whole transportation.

Recall that customers in CC and DD have wider time windows than those in AA and BB. From column "PDPCBT", instances with tighter time windows tend to have higher total costs compared with those with wider time windows in most cases. Nevertheless, the bid usages in Classes CC and DD are smaller than those in AA and BB. Because the delivery of requests in CC and DD is not urgent, it can be cheaper to visit one more request with extra mileage in the same area for private vehicles instead of selecting bids with compensation. This observation also leads to the following sensitivity analysis on how the compensation coefficient ρ affects the decision.

5.4.4. Analysis on the compensation coefficient

We test on specific instances with the combination of $n \in \{35, 40, 45\}$ and PDB $\times$ RB = FF in each problem class. Meanwhile, we fix the compensation coefficient $\rho = \{0.25, 0.5, 0.75, 1.0, 1.25\}$

in these instances. The analysis consists of 60 instances, and each of them could be solved to optimality by our BPC algorithm within the time limit. The results are summarized in Table 11 and the columns have the same meaning as those in Table 10.

Table 11: The sensitivity analysis on the compensation coefficient ρ.

	$\rho = 0.25$		$\rho = 0.50$		$\rho = 0.75$		$\rho = 1.00$		$\rho = 1.25$	
Name	Saving(%)	Rate(%)	Saving(%)	Rate(%)	Saving(%)	Rate(%)	Saving(%)	Rate(%)	Saving(%)	Rate(%)
AA35FF3	8.74	28.57	6.46	20.41	5.53	14.29	5.23	10.20	5.23	10.20
AA40FF3	10.74	33.93	7.75	21.43	6.72	12.50	6.41	10.71	6.36	10.71
AA45FF3	10.47	20.63	8.75	19.05	7.46	15.87	6.94	12.70	6.71	12.70
BB35FF3	7.41	34.69	4.07	20.41	2.54	14.29	2.10	8.16	2.10	8.16
BB40FF3	10.96	33.93	7.31	21.43	5.27	17.86	4.30	14.29	4.09	8.93
BB45FF4	9.53	38.10	5.91	22.22	4.14	14.29	3.48	12.70	3.23	9.52
CC35FF3	6.87	34.69	4.22	16.33	3.71	12.24	3.71	12.24	3.71	12.24
CC40FF3	6.82	23.21	4.13	19.64	3.11	14.29	2.75	10.71	2.75	10.71
CC45FF3	5.03	25.40	3.08	12.70	2.58	9.52	2.33	9.52	2.18	11.11
DD35FF3	7.97	26.53	4.54	24.49	2.71	12.24	2.67	4.08	2.67	4.08
DD40FF3	6.21	28.57	3.82	19.64	3.02	12.50	2.95	10.71	2.95	10.71
DD45FF3	7.01	33.33	3.74	19.05	3.02	12.70	2.60	11.11	2.28	11.11
Average	8.15	30.13	5.32	19.73	4.15	13.55	3.79	10.60	3.69	10.02

Still, there is not much relationship between the considered available bids and bid usages. However, with a higher ρ of the crowdshippers, columns "Rate(%)" reflect the same observation that it is not worthwhile selecting more bids to make the delivery. Despite $\rho = 1.25$ means that the travel cost of the crowdshipper is higher than the vehicle, utilizing crowdshipping is profitable for decision-making. When ρ is at a lower rate instead, delivery with crowdshipping can achieve a higher cost reduction. For example, with $\rho = 0.25$ in BB40FF3, the rate of bid usage is up to 33.93%, leading to a significant fall of over 10% in the total cost. We also see that for some instances, the bid usage is consistent at different values of ρ. By exploiting the obtained optimal solution, we found that the same bid selection occurs in these situations, which represents that selecting these bids is quite superior to making extra delivery for vehicles for cost savings. In this case, the crowdshippers of these bids have an incentive to inflate their bid prices for more utility. However, this would subsequently result in higher shipping costs for the e-commerce company. According to Zou and Kafle (2022), designing mechanisms is a good choice to avoid misreporting information and incentivize truthful bidding, which can serve as a future research direction.

5.4.5. Analysis on the failed package delivery

Finally, we analyze a common issue where failed deliveries happen in crowdshipping. After optimizing the bid selection and route scheduling, some customers may be unserved because of the no-shows of crowdshippers in the distribution process. We assume a fleet of backup vehicles is used to handle the failed package delivery. Starting and ending at the depot, each backup

vehicle performs the following door-to-door delivery for each unserved customer: (i) For each sender in a failed pickup bid, it delivers the packages from this sender to its receiver. Note that the related planned route does not require a detour to its receiver, nor the bid-related facility if no other bids are assigned; (ii) For each receiver in a failed delivery bid, it visits the corresponding transshipment facility and takes the packages to its receiver. Note that the related planned route still requires a detour to the facility for package transshipment; and (iii) For each failed request bid, it directly collects the package from the sender to its receiver. Accordingly, the re-plan cost comprises the travel costs of backup vehicles, the new travel costs of planned routes, and the compensation of crowdshippers that successfully execute the delivery.

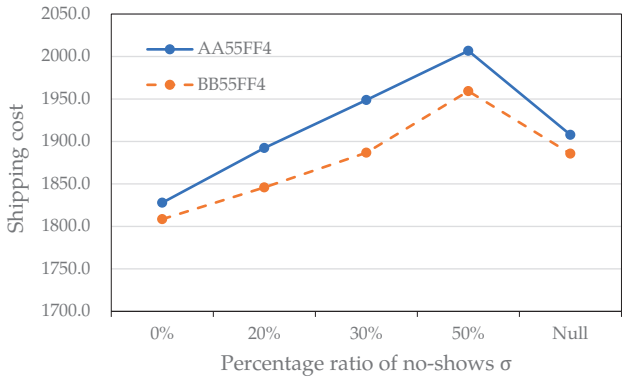


Figure 6: Analysis on the failed package delivery with different σ.

We study AA55FF4 and BB55FF4 with different $\sigma = 20, 30$, and 50% where the first σ of selected bids are assumed to be failed. The consequent results are visualized in Figure 6, in which 0% represents that all selected bids are executed without no-shows, and Null indicates the shipping cost of truck-only transportation. It is clear that the re-plan cost increases with the failure ratio σ. Compared to the case with truck-only, incorporating crowdshipping can still benefit cost savings when σ is low. While the company is not recommended to blindly outsource the delivery when crowdshippers are at a high ratio.

6. Conclusions

Motivated by the last-mile delivery of e-commerce companies, this paper addresses a variant of pickup and delivery problems considering both crowdsourced bids and transshipment (PDPCBT). To attract more crowdshippers to submit bids for delivery tasks, transshipment facilities are used for parcel relay. Consequently, the private vehicle fleet has to work in tandem with crowdshippers, which makes the problem more complicated and challenging. To minimize the total cost consisting of the travel cost of vehicles and the compensation of crowdshippers, we aim at determining the optimal bid selection and route scheduling. We first formulate the PDPCBT

into an arc-based formulation on a logical network and observe that problem is hard to tackle due to its size. Then we reformulate it into a route-based model on a physical network, which is solved exactly by a tailored branch-and-price-and-cut (BPC) algorithm. Notably, we decompose the two possible ways of serving requests into a shortest path problem and a knapsack problem in the same algorithm framework.

To evaluate the performance of the proposed BPC algorithm, we conduct computational experiments based on the request instances introduced by Ropke and Cordeau (2009) and the randomly generated bid instances. Extensive results indicate that the bi-directional algorithm integrated with acceleration techniques achieves outstanding performance. It optimally solved 138 out of 160 instances, together with feasible upper bounds for 7 instances. Furthermore, the proposed valid inequalities and strong branching can effectively speed up the convergence of the BPC algorithm. Finally, we conduct a sensitivity analysis of the obtained solutions and provide managerial findings for the e-commerce company. It shows that incorporating crowdshipping results in an average drop of 2.72% in the total shipping cost, compared with truck-only transportation, even when the expected compensation is high or the failed deliveries are at a low ratio. We can conclude that crowdshipping offers an excellent chance for cost reduction.

This paper has some limitations and we suggest the following possible directions for future research:

1. A finite capacity can be limited to the transshipment facility for parcel relay temporarily. The problem would then require a synchronization constraint in which routes and bids can affect each other. That is, the change in the arrival time of one vehicle or crowdshipper at a transshipment facility may make the facility overload.

2. Due to information asymmetry, crowdshipper may strategically misreport one's willingness to increase the compensation, but also incur a higher shipping cost for the company in the reality. Therefore, it would be beneficial to design appropriate mechanisms for the company so that crowdshippers must bid truthfully.

3. Because of the unpredictable behaviors of crowdshippers such as no-shows, it is interesting to investigate how to extend this static problem to an uncertain problem in a dynamic crowdsourced system. At this point, decision-makers must re-plan routes when vehicles have already set off for delivery.

Acknowledgments

The authors wish to sincerely thank the associate editor and two anonymous reviewers for their valuable suggestions and insightful comments to improve the manuscript.

References

Aized, T., Srai, J.S., 2014. Hierarchical modelling of last mile logistic distribution system. The International Journal of Advanced Manufacturing Technology 70, 1053–1061.

Alcaraz, J.J., Caballero-Arnaldos, L., Vales-Alonso, J., 2019. Rich vehicle routing problem with last-mile outsourcing decisions. Transportation Research Part E: Logistics and Transportation Review , 263–286.

Alnaggar, A., Gzara, F., Bookbinder, J., 2019. Crowdsourced delivery: A review of platforms and academic literature. Omega-international Journal of Management Science 98, 102139.

Archetti, C., Savelsbergh, M., Speranza, M., 2016. The vehicle routing problem with occasional drivers. Eur. J. Oper. Res. 254, 472–480.

Arslan, A.M., Agatz, N.A.H., Kroon, L.G., Zuidwijk, R.A., 2019. Crowdsourced delivery - a dynamic pickup and delivery problem with ad hoc drivers. Transp. Sci. 53, 222–235.

Baldacci, R., Bartolini, E., Mingozzi, A., 2011. An exact algorithm for the pickup and delivery problem with time windows. Oper. Res. 59, 414–426.

Baldacci, R., Ngueveu, S.U., Calvo, R.W., 2017. The vehicle routing problem with transhipment facilities. Transp. Sci. 51, 592–606.

Behrend, M., Meisel, F., Fagerholt, K., Andersson, H., 2019. An exact solution method for the capacitated item-sharing and crowdshipping problem. Eur. J. Oper. Res. 279, 589–604.

Boysen, N., Emde, S., Schwerdfeger, S., 2022. Crowdshipping by employees of distribution centers: Optimization approaches for matching supply and demand. Eur. J. Oper. Res. 296, 539–556.

Boysen, N., Fedtke, S., Schwerdfeger, S., 2021. Last-mile delivery concepts: a survey from an operational research perspective. OR Spectrum 43, 1–58.

Cortés, C., Matamala, M., Contardo, C., 2010. The pickup and delivery problem with transfers: Formulation and a branch-and-cut solution method. Eur. J. Oper. Res. 200, 711–724.

Dahle, L., Andersson, H., Christiansen, M., Speranza, M.G., 2019. The pickup and delivery problem with time windows and occasional drivers. Comput. Oper. Res. 109, 122–133.

Desaulniers, G., Desrosiers, J., Solomon, M.M., 2005. Column Generation. Springer US.

Desaulniers, G., Lessard, F., Hadjar, A., 2008. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. Transp. Sci. 42, 387–404.

Feng, X., Chu, F., Chu, C., Huang, Y., 2021. Crowdsource-enabled integrated production and transportation scheduling for smart city logistics. International Journal of Production Research 59, 2157–2176.

Friedrich, C., Elbert, R., 2022. Adaptive large neighborhood search for vehicle routing problems with transshipment facilities arising in city logistics. Comput. Oper. Res. 137, 105491.

Ghilas, V., Cordeau, J., Demir, E., Woensel, T.V., 2018. Branch-and-price for the pickup and delivery problem with time windows and scheduled lines. Transp. Sci. 52, 1191–1210.

Huang, K., Ardiansyah, M., 2019. A decision model for last-mile delivery planning with crowdsourcing integration. Comput. Ind. Eng. 135, 898–912.

Jepsen, M., Petersen, B., Spoorendonk, S., Pisinger, D., 2008. Subset-row inequalities applied to the vehicle-routing problem with time windows. Oper. Res. 56, 497–511.

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

Laporte, G., 1992. The traveling salesman problem: An overview of exact and approximate algorithms. European Journal of Operational Research 59, 231–247.

Le, T.V., Stathopoulos, A., van Woensel, T., Ukkusuri, S.V., 2019. Supply, demand, operations, and management of

crowd-shipping services: A review and empirical evidence. Transportation Research Part C: Emerging Technologies 103, 83–103.

Li, J., Qin, H., Baldacci, R., Zhu, W., 2020. Branch-and-price-and-cut for the synchronized vehicle routing problem with split delivery, proportional service time and multiple time windows. Transportation Research Part E: logistics and Transportation Review 140, 101955.

Lu, N., 2022. Research on the crowd-sourcing platform mode and car-cargo matching mechanism of intra-city freight from the perspective of carbon emissions reduction. Master's thesis. Chongqing Jiaotong University. Chongqing, China.

Luo, Z., Qin, H., Lim, A., 2014. Branch-and-price-and-cut for the multiple traveling repairman problem with distance constraints. Eur. J. Oper. Res. 234, 49–60.

Macrina, G., Pugliese, L., Guerriero, F., Laganà, D., 2017. The vehicle routing problem with occasional drivers and time windows, in: Optimization and Decision Science: Methodologies and Applications, pp. 577–587.

Macrina, G., Pugliese, L., Guerriero, F., Laporte, G., 2020. Crowd-shipping with time windows and transshipment nodes. Comput. Oper. Res. 113, 104806.

Masson, R., Lehuédé, F., Péton, O., 2013. An adaptive large neighborhood search for the pickup and delivery problem with transfers. Transp. Sci. 47, 344–355.

Mourad, A., Puchinger, J., Chu, C., 2019. A survey of models and algorithms for optimizing shared mobility. Transportation Research Part B: Methodological 123, 323–346.

Pourrahmani, E., Jaller, M., 2021. Crowdshipping in last mile deliveries: Operational challenges and research opportunities. Socio-economic Planning Sciences 78, 101063.

Qin, H., Su, E., Wang, Y., Li, J., 2022. Branch-and-price-and-cut for the electric vehicle relocation problem in one-way carsharing systems. Omega 109, 102609.

Qin, H., Su, X., Ren, T., Luo, Z., 2021. A review on the electric vehicle routing problems: Variants and algorithms. Frontiers of Engineering Management 8, 370–389.

Righini, G., Salani, M., 2006. Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. Discret. Optim. 3, 255–273.

Ropke, S., Cordeau, J., 2009. Branch and cut and price for the pickup and delivery problem with time windows. Transp. Sci. 43, 267–286.

Ropke, S., Pisinger, D., 2007. Complexity of shortest path problems arising in column generation for vehicle routing problems with pairing and precedence constraints. Technical Report. Technical University of Denmark. Lyngby, Denmark.

Sampaio, A.H., Savelsbergh, M., Veelenturf, L.P., Woensel, T.V., 2020. Delivery systems with crowd-sourced drivers: A pickup and delivery problem with transfers. Networks 76, 232–255.

dos Santos, A.G., Viana, A., Pedroso, J.P., 2022. 2-echelon lastmile delivery with lockers and occasional couriers. Transportation Research Part E: Logistics and Transportation Review 162, 102714.

Sun, P., Veelenturf, L.P., Hewitt, M., van Woensel, T., 2018. The time-dependent pickup and delivery problem with time windows. Transportation Research Part B: Methodological 116, 1–24.

Tilk, C., Rothenbächer, A.K., Gschwind, T., Irnich, S., 2017. Asymmetry matters: Dynamic half-way points in bidirectional labeling for solving shortest path problems with resource constraints faster. Eur. J. Oper. Res. 261, 530–539.

Voigt, S., Kuhn, H., 2021. Crowdsourced logistics: the pickup and delivery problem with transshipments and occasional drivers. Networks 79, 403–426.

Wang, Y., 2018. Research on crowdsourcing logistics model: a case study of dada - jd daojia co. Innovation 5, 23–33.

Yang, W., Ke, L., Wang, D.Z.W., Lam, J.S.L., 2021. A branch-price-and-cut algorithm for the vehicle routing problem

with release and due dates. Transportation Research Part E: logistics and Transportation Review 145, 102167.

Yao, Y., van Woensel, T., Veelenturf, L.P., Mo, P., 2021. The consistent vehicle routing problem considering path consistency in a road network. Transportation Research Part B: Methodological 153, 21–44.

Yu, V.F., Jodiawan, P., Redi, A.A.N.P., 2022. Crowd-shipping problem with time windows, transshipment nodes, and delivery options. Transportation Research Part E: Logistics and Transportation Review 157, 102545.

Zhang, X., Li, Z., 2017. The challenge and operation strategy of collaborative economy platform: A case study of yun niao platform. Industrial Economy Review 4, 13–22.

Zhou, L., Baldacci, R., Vigo, D., Wang, X., 2018. A multi-depot two-echelon vehicle routing problem with delivery options arising in the last mile distribution. Eur. J. Oper. Res. 265, 765–778.

Zou, B., Kafle, N., 2022. Designing mechanisms for crowdsourced urban parcel delivery. Transportation Letters .

Appendix A. Detailed instance-level results

Tables A.1-A.4 report detailed instance-level results for four problem classes of the PDPCBT. For each table, columns "Name" and "UB" provide the instance name and the corresponding upper bound obtained by the branch-and-price-and-cut algorithm within the time limit, respectively. Columns "LB" and "LBC" give the lower bound obtained by solving the linear relaxation of the route-based formulation without and with subset row cuts, respectively. Columns "$Gap_1(\%)$" and "$Gap_2(\%)$" present the integrity gap between the upper bound and the corresponding optimal value of the linear relaxation in percentage. The column "SRCs" records the number of cuts added to the root node in total. Columns "Nodes" and "Time" give the number of branch-and-bound nodes explored by the algorithm and the total computational time in seconds, respectively. The following three columns report the performance at the root node. Column "T_{LB}" gives the computational time in seconds to solve the linear relaxation of the route-based formulation without any valid inequalities. Column "Labels" reports the number of labels generated in the last column generation iteration. The penultimate column "T_B" reports the running time spent in the branching procedure. Finally, column "B_{sel}" presents the numbers of selected bids.

We mark "B" as the footnote of the upper bound (column "UB") if an instance can reach a feasible upper bound by the algorithm but without proving optimal. While an instance cannot be solved within the time limit, we fill the cell with "N/A". Note that cells are filled with "–" if we cannot provide the corresponding optimal information.

Table A.1: Detailed instance-level results of Class AA

Name	UB	LB	Gap_1(%)	LBC	Gap_2(%)	SRCs	Nodes	Time	T_{LB}	Labels	T_B	B_{sel}
AA30FF3	1041.068	1040.057	0.10	1041.068	0.00	23	1	2.8	1.9	349	0.0	10
AA30FH3	1069.865	1069.865	0.00	1069.865	0.00	0	1	1.4	1.4	385	0.0	8
AA30HF3	1050.685	1048.273	0.23	1050.685	0.00	30	1	1.4	0.5	265	0.0	7
AA30HH3	1082.111	1082.111	0.00	1082.111	0.00	0	1	2.1	2.1	269	0.0	4
AA35FF3	1229.293	1220.904	0.68	1229.293	0.00	43	1	7.5	3.4	482	0.0	7
AA35FH3	1269.313	1260.573	0.69	1269.313	0.00	49	1	12.1	4.9	520	0.0	4
AA35HF3	1229.392	1222.239	0.58	1229.392	0.00	20	1	3.3	1.5	407	0.0	7
AA35HH3	1269.313	1261.376	0.63	1269.313	0.00	100	1	9.7	6.2	449	0.0	4
AA40FF3	1420.186	1417.713	0.17	1420.186	0.00	212	1	11.5	7.6	581	0.0	9
AA40FH3	1438.444	1436.507	0.13	1438.444	0.00	62	1	17.5	8.5	570	0.0	7
AA40HF3	1420.934	1419.570	0.10	1420.934	0.00	50	1	6.6	3.7	529	0.0	7
AA40HH3	1466.729	1465.676	0.07	1466.729	0.00	50	1	10.1	7.3	547	0.0	4
AA45FF3	1624.096	1623.698	0.02	1624.096	0.00	24	1	26.2	19.4	826	0.0	11
AA45FH3	1645.075	1645.075	0.00	1645.075	0.00	0	1	46.8	46.6	808	0.0	9
AA45HF3	1645.581	1639.237	0.39	1645.581	0.00	110	1	22.8	13.7	740	0.0	8
AA45HH3	1675.082	1668.736	0.38	1675.082	0.00	153	1	36.8	18.8	851	0.0	6
AA50FF4	1700.495	1693.380	0.42	1700.495	0.00	29	1	16.6	12.7	817	0.0	9
AA50FH4	1732.808	1714.264	1.07	1727.451	0.31	192	13	338.1	20.4	836	39.3	8
AA50HF4	1718.783	1707.290	0.67	1718.783	0.00	29	1	11.5	8.0	653	0.0	7
AA50HH4	1747.162	1730.575	0.95	1744.272	0.17	43	3	36.1	14.2	704	4.3	4
AA55FF4	1827.906	1820.169	0.42	1824.879	0.17	50	5	123.2	25.9	877	2.3	12
AA55FH4	1873.299	1871.024	0.12	1873.299	0.00	50	1	54.6	45.1	940	0.0	8
AA55HF4	1834.730	1831.583	0.17	1831.730	0.16	9	5	46.8	14.2	678	9.4	8
AA55HH4	1895.722	1893.795	0.10	1895.722	0.00	50	1	32.4	28.8	793	0.0	4
AA60FF4	2000.083	1999.106	0.05	2000.083	0.00	50	1	43.0	35.9	977	0.0	17
AA60FH4	2052.251	2046.594	0.28	2051.770	0.02	82	3	270.9	69.5	1136	12.8	14
AA60HF4	2012.996	2003.968	0.45	2010.145	0.14	63	5	104.2	24.8	882	15.2	13
AA60HH4	2073.678	2073.313	0.02	2073.678	0.00	36	1	49.7	54.7	907	0.0	8
AA65FF4	2155.594	2151.817	0.18	2154.037	0.07	200	3	168.3	74.2	1140	14.1	15
AA65FH4	2203.861	2195.143	0.40	2202.838	0.05	156	3	368.7	132.8	1378	15.2	8
AA65HF4	2167.429	2164.157	0.15	2165.677	0.08	289	7	212.6	44.1	906	20.4	12
AA65HH4	2219.576	2209.250	0.47	2216.652	0.13	200	3	217.6	116.9	1090	18.2	4
AA70FF4	2321.086	2309.855	0.48	2318.289	0.12	155	7	694.1	111.9	1824	24.4	15
AA70FH4	2374.602	2362.325	0.52	2373.421	0.05	195	3	601.4	184.5	1932	22.5	11
AA70HF4	2344.578	2334.524	0.43	2340.973	0.15	67	9	485.2	129.9	1436	1.1	13
AA70HH4	2404.767	2395.919	0.37	2404.767	0.00	171	1	231.9	177.2	1691	0.0	7
AA75FF5	2393.285	2389.698	0.15	2392.642	0.03	158	3	239.1	85.2	1958	6.6	11
AA75FH5	2424.046	2422.899	0.05	2423.047	0.04	50	3	246.7	103.5	2210	37.2	8
AA75HF5	2408.313	2401.642	0.28	2405.403	0.12	130	15	644.8	49.0	1496	50.5	8
AA75HH5	2439.216	2433.018	0.25	2438.769	0.02	73	3	203.2	90.5	1770	7.3	4

Table A.2: Detailed instance-level results of Class BB

Name	UB	LB	Gap_1(%)	LBC	Gap_2(%)	SRCs	Nodes	Time	T_{LB}	Labels	T_B	B_{sel}
BB30FF3	1054.528	1054.528	0.00	1054.528	0.00	0	1	1.3	1.1	287	0.0	5
BB30FH3	1064.508	1064.508	0.00	1064.508	0.00	0	1	2.0	1.8	283	0.0	3
BB30HF3	1059.246	1059.246	0.00	1059.246	0.00	0	1	1.0	0.8	225	0.0	4
BB30HH3	1069.226	1069.226	0.00	1069.226	0.00	0	1	3.2	3.1	230	0.0	2
BB35FF3	1249.872	1248.810	0.09	1249.872	0.00	50	1	5.7	3.3	507	0.0	5
BB35FH3	1277.234	1277.094	0.01	1277.234	0.00	100	1	11.2	5.4	529	0.0	2
BB35HF3	1249.872	1248.810	0.09	1249.872	0.00	50	1	5.1	2.9	365	0.0	5
BB35HH3	1277.234	1277.143	0.01	1277.234	0.00	50	1	10.0	6.9	445	0.0	2
BB40FF3	1420.212	1417.871	0.16	1420.212	0.00	107	1	10.8	6.4	743	0.0	9
BB40FH3	1459.769	1448.945	0.74	1451.361	0.58	50	7	221.3	11.0	915	27.3	5
BB40HF3	1429.946	1425.956	0.28	1429.946	0.00	133	1	10.2	4.5	558	0.0	5
BB40HH3	1460.535	1460.535	0.00	1460.535	0.00	0	1	10.4	10.1	678	0.0	3
BB45FF4	1486.311	1486.183	0.01	1486.311	0.00	50	1	9.0	6.4	888	0.0	9
BB45FH4	1523.817	1521.458	0.15	1523.817	0.00	71	1	14.0	10.8	796	0.0	6
BB45HF4	1499.478	1497.193	0.15	1499.411	0.00	108	3	14.3	4.1	745	1.3	7
BB45HH4	1539.503	1538.723	0.05	1539.503	0.00	50	1	10.9	8.3	660	0.0	2
BB50FF4	1657.525	1650.409	0.43	1657.448	0.00	174	3	60.9	11.4	1178	8.5	9
BB50FH4	1673.653	1667.135	0.39	1672.938	0.04	178	3	84.5	20.9	1127	9.5	7
BB50HF4	1668.520	1659.948	0.51	1665.852	0.16	132	9	69.2	7.0	1019	11.5	7
BB50HH4	1686.682	1680.224	0.38	1684.664	0.12	121	3	75.7	15.8	1034	16.9	4
BB55FF4	1808.502	1807.317	0.07	1808.502	0.00	100	1	36.2	30.4	1593	0.0	11
BB55FH4	1841.803	1836.723	0.28	1841.803	0.00	132	1	80.3	56.1	1842	0.0	6
BB55HF4	1808.628	1807.380	0.07	1808.628	0.00	50	1	32.2	23.7	1268	0.0	10
BB55HH4	1846.007	1838.677	0.40	1846.007	0.00	182	1	60.6	36.2	1422	0.0	6
BB60FF6	2373.040	2371.889	0.05	2371.941	0.05	9	7	145.8	27.5	1458	3.8	9
BB60FH6	2373.040	2372.130	0.04	2372.208	0.04	23	5	181.4	36.9	1357	5.4	9
BB60HF6	2381.717	2378.144	0.15	2378.797	0.12	106	11	210.7	16.1	1146	9.6	7
BB60HH6	2393.360	2392.411	0.04	2393.360	0.00	100	1	35.2	27.7	1099	0.0	3
BB65FF6	2568.385	2568.118	0.01	2568.385	0.00	92	1	45.4	41.8	1734	0.0	12
BB65FH6	2575.729	2571.954	0.15	2573.128	0.10	74	13	507.2	68.7	1737	18.9	12
BB65HF6	2591.552	2584.700	0.26	2588.473	0.12	118	11	177.2	27.2	1427	7.7	9
BB65HH6	2607.818	2602.097	0.22	2604.911	0.11	66	5	275.3	47.1	1400	8.6	7
BB70FF6	2814.305	2812.136	0.08	2813.666	0.02	48	3	154.2	49.4	1954	7.9	21
BB70FH6	2859.189	2845.078	0.49	2857.527	0.06	172	3	230.0	92.6	1921	11.4	19
BB70HF6	2829.573	2826.200	0.12	2829.573	0.00	136	1	61.9	42.1	1733	0.0	10
BB70HH6	2885.667	2871.507	0.49	2879.040	0.23	66	5	265.0	62.3	1695	7.6	4
BB75FF6	2883.578	2881.105	0.09	2881.155	0.08	83	3	237.8	68.8	2250	3.7	14
BB75FH6	2939.787	2928.628	0.38	2934.479	0.18	113	13	1248.0	118.6	2379	65.4	15
BB75HF6	2905.810	2894.168	0.40	2894.284	0.40	37	21	914.4	47.4	1905	40.1	12
BB75HH6	2968.276	2939.772	0.96	2954.512	0.46	100	21	1393.4	79.1	2035	76.7	7

Table A.3: Detailed instance-level results of Class CC

Name	UB	LB	Gap_1(%)	LBC	Gap_2(%)	SRCs	Nodes	Time	T_{LB}	Labels	T_B	B_{sel}
CC30FF3	1057.402	1053.602	0.36	1057.402	0.00	100	1	4.7	1.5	614	0.0	4
CC30FH3	1061.700	1061.700	0.00	1061.700	0.00	0	1	3.1	2.8	509	0.0	3
CC30HF3	1057.402	1053.602	0.36	1057.402	0.00	100	1	2.7	1.2	490	0.0	4
CC30HH3	1061.700	1061.700	0.00	1061.700	0.00	0	1	2.3	2.1	385	0.0	3
CC35FF3	1184.928	1184.928	0.00	1184.928	0.00	0	1	3.1	2.7	706	0.0	6
CC35FH3	1206.072	1204.673	0.12	1206.072	0.00	40	1	5.2	3.9	755	0.0	3
CC35HF3	1184.928	1184.928	0.00	1184.928	0.00	0	1	2.3	2.0	668	0.0	6
CC35HH3	1206.072	1204.673	0.12	1206.072	0.00	40	1	6.0	3.2	820	0.0	3
CC40FF3	1312.661	1306.536	0.47	1312.661	0.00	100	1	17.0	7.8	1136	0.0	9
CC40FH3	1333.116	1327.587	0.41	1333.116	0.00	100	1	26.6	12.3	1357	0.0	4
CC40HF3	1317.421	1312.206	0.40	1317.421	0.00	100	1	10.8	5.2	892	0.0	7
CC40HH3	1343.648	1335.191	0.63	1343.648	0.00	100	1	15.4	6.9	1098	0.0	4
CC45FF3	1469.500	1461.176	0.57	1469.500	0.00	100	1	55.4	18.5	2125	0.0	7
CC45FH3	1500.064	1487.911	0.81	1493.217	0.46	131	11	1383.4	38.5	3971	65.0	4
CC45HF3	1469.500	1461.176	0.57	1469.500	0.00	94	1	27.9	10.6	1659	0.0	7
CC45HH3	1500.064	1488.007	0.80	1492.829	0.48	127	7	443.9	20.5	3517	45.1	4
CC50FF4	1650.030	1650.030	0.00	1650.030	0.00	0	1	21.0	20.7	1941	0.0	7
CC50FH4	1673.504	1665.604	0.47	1666.170	0.44	12	13	211.8	27.7	2937	31.7	7
CC50HF4	1650.030	1650.030	0.00	1650.030	0.00	0	1	11.8	11.6	1862	0.0	7
CC50HH4	1673.597	1665.832	0.46	1666.518	0.42	30	11	125.6	17.7	2693	13.5	3
CC55FF4	1795.125	1792.777	0.13	1794.384	0.04	105	3	88.5	38.4	4365	5.9	7
CC55FH4	1824.410	1809.604	0.81	1815.136	0.51	152	31	3932.2	59.8	4544	205.3	5
CC55HF4	1795.211	1793.691	0.08	1795.211	0.00	77	1	32.5	30.8	3769	0.0	6
CC55HH4	1827.359	1812.835	0.79	1818.685	0.47	82	25	1810.6	55.8	3924	99.5	3
CC60FF4	1951.457	1941.447	0.51	1945.916	0.28	128	13	960.7	80.5	5183	54.7	8
CC60FH4	1976.793^B	1959.430	0.88	1964.793	0.61	82	30	7200.0	138.2	7375	272.2	–
CC60HF4	1951.457	1941.447	0.51	1946.304	0.26	131	5	310.6	60.0	5033	10.3	8
CC60HH4	1979.235	1959.957	0.97	1965.717	0.68	94	61	7061.1	105.0	6479	252.0	5
CC65FF4	2074.746	2072.057	0.13	2074.746	0.00	173	1	201.5	167.3	7504	0.0	11
CC65FH4	N/A	2110.540	–	2118.182	–	56	8	7200.0	390.3	12734	122.9	–
CC65HF4	2080.462	2076.103	0.21	2079.417	0.05	125	3	360.2	84.3	6829	14.4	10
CC65HH4	2137.451^B	2118.285	0.90	2125.232	0.57	55	16	7200.0	220.1	10421	89.2	–
CC70FF5	2148.369	2148.369	0.00	2148.369	0.00	0	1	104.0	103.9	5334	0.0	12
CC70FH5	2175.444	2170.881	0.21	2175.444	0.00	100	1	282.2	206.5	8083	0.0	8
CC70HF5	2156.959	2156.959	0.00	2156.959	0.00	0	1	79.6	79.6	4529	0.0	9
CC70HH5	2186.588	2179.246	0.34	2186.588	0.00	37	1	209.7	173.4	8075	0.0	5
CC75FF5	2311.797	2303.582	0.36	2304.948	0.30	102	25	4029.3	223.1	7102	119.9	10
CC75FH5	2342.912	2336.017	0.29	2339.724	0.14	67	15	4142.3	320.6	8787	170.0	5
CC75HF5	2313.711	2307.938	0.25	2309.312	0.19	54	9	710.6	117.2	6506	24.3	9
CC75HH5	2347.961	2340.227	0.33	2343.235	0.20	54	13	1626.5	204.7	7505	30.6	4

B: An upper bound that has not been proved optimally.

N/A: No solution is provided within the time limit.

–: No information is provided in the cell.

Table A.4: Detailed instance-level results of Class DD

Name	UB	LB	Gap_1(%)	LBC	Gap_2(%)	SRCs	Nodes	Time	T_{LB}	Labels	T_B	B_{sel}
DD30FF2	1107.482	1098.915	0.77	1107.482	0.00	50	1	127.9	11.7	1924	0.0	7
DD30FH2	1120.123	1102.429	1.58	1120.123	0.00	60	1	283.5	16.9	2163	0.0	2
DD30HF2	1121.557	1102.417	1.71	1121.557	0.00	100	1	197.6	9.0	1984	0.0	3
DD30HH2	1122.556	1105.198	1.55	1122.556	0.00	64	1	88.7	15.1	1878	0.0	1
DD35FF3	1170.470	1166.488	0.34	1170.470	0.00	100	1	11.4	9.1	1694	0.0	8
DD35FH3	1175.747	1175.747	0.00	1175.747	0.00	0	1	13.4	13.0	1954	0.0	6
DD35HF3	1177.963	1173.614	0.37	1177.963	0.00	50	1	16.2	7.3	1646	0.0	3
DD35HH3	1180.505	1180.412	0.01	1180.505	0.00	48	1	12.1	8.2	1912	0.0	5
DD40FF3	1306.058	1306.058	0.00	1306.058	0.00	0	1	22.3	21.9	2569	0.0	8
DD40FH3	1319.814	1318.040	0.13	1319.814	0.00	102	1	33.2	24.9	3050	0.0	5
DD40HF3	1310.119	1310.119	0.00	1310.119	0.00	0	1	13.2	13.0	2482	0.0	7
DD40HH3	1323.875	1321.376	0.19	1323.875	0.00	174	1	33.6	20.6	2765	0.0	4
DD45FF3	1434.775	1434.775	0.00	1434.775	0.00	0	1	41.7	41.3	5040	0.0	8
DD45FH3	1451.786	1451.510	0.02	1451.786	0.00	50	1	60.6	50.1	4487	0.0	6
DD45HF3	1450.156	1441.893	0.57	1448.403	0.12	137	5	364.5	31.2	3535	13.3	5
DD45HH3	1467.167	1456.570	0.72	1467.167	0.00	149	1	254.6	41.0	4039	0.0	3
DD50FF3	N/A	1535.742	–	1551.193	–	145	8	7200.0	68.9	5785	134.4	–
DD50FH3	N/A	1549.351	–	1562.226	–	141	3	7200.0	93.8	6619	59.5	–
DD50HF3	1571.435	1544.335	1.72	1559.663	0.75	133	21	6993.5	42.8	4366	108.4	7
DD50HH3	1580.077^B	1556.581	1.49	1571.304	0.56	115	15	7200.0	62.7	4426	146.6	–
DD55FF3	N/A	1686.449	–	1696.679	–	105	7	7200.0	148.5	10931	68.2	–
DD55FH3	1708.054	1695.895	0.71	1705.196	0.17	85	3	3560.1	166.1	10546	41.0	6
DD55HF3	1712.634	1695.076	1.03	1703.245	0.55	95	18	6884.8	100.0	9990	71.4	8
DD55HH3	1718.066	1707.780	0.60	1716.000	0.12	94	3	941.6	126.9	10041	33.8	6
DD60FF3	N/A	1843.448	–	1854.346	–	103	4	7200.0	203.0	13204	40.3	–
DD60FH3	N/A	1869.794	–	1880.622	–	101	4	7200.0	461.3	20262	35.2	–
DD60HF3	1880.972^B	1850.039	1.64	1862.816	0.97	106	16	7200.0	227.4	14514	87.7	–
DD60HH3	N/A	1877.858	–	1893.166	–	127	4	7200.0	372.7	18831	37.5	–
DD65FF4	2051.791^B	2036.629	0.74	2045.684	0.30	117	15	7200.0	109.8	5523	97.9	–
DD65FH4	N/A	2054.973	–	2067.332	–	135	16	7200.0	170.0	6918	64.5	–
DD65HF4	2057.140	2041.681	0.75	2049.843	0.35	152	15	5041.4	77.4	4889	73.1	9
DD65HH4	2079.293^B	2060.161	0.92	2072.310	0.34	124	14	7200.0	111.0	5745	45.5	–
DD70FF4	N/A	2130.958	–	2139.914	–	90	8	7200.0	165.8	8513	64.0	–
DD70FH4	N/A	2168.498	–	2178.499	–	134	3	7200.0	276.7	9253	28.2	–
DD70HF4	2145.314^B	2131.069	0.66	2140.022	0.25	131	17	7200.0	143.8	8125	116.3	–
DD70HH4	N/A	2168.703	–	2178.713	–	151	4	7200.0	194.1	8819	33.8	–
DD75FF4	N/A	2279.822	–	2283.265	–	175	8	7200.0	395.9	13455	155.1	–
DD75FH4	N/A	2304.753	–	2311.089	–	116	5	7200.0	577.8	15797	79.4	–
DD75HF4	N/A	2280.912	–	2285.795	–	134	7	7200.0	238.5	11639	98.4	–
DD75HH4	N/A	2308.660	–	2314.763	–	102	6	7200.0	324.0	12552	105.8	–

B: An upper bound that has not been proved optimally.

N/A: No solution is provided within the time limit.

–: No information is provided in the cell.