A Case Study of the Profit-Maximizing Multi-Vehicle Pickup and Delivery Selection Problem for the Road Networks with the Integratable Nodes

Aolong Zha[1][0000−0003−2480−6597], Qiong Chang[2], Naoto Imura[1], and Katsuhiro Nishinari[1]

[1] The University of Tokyo, Tokyo, Japan {a-zha,nimura}@g.ecc.u-tokyo.ac.jp tknishi@mail.ecc.u-tokyo.ac.jp

[2] Tokyo Institute of Technology, Tokyo, Japan q.chang@c.titech.ac.jp

Abstract. This paper is a study of an application-based model in profit-maximizing multi-vehicle pickup and delivery selection problem (PPDSP). The graph-theoretic model proposed by existing studies of PPDSP is based on transport requests to define the corresponding nodes (i.e., each request corresponds to a pickup node and a delivery node). In practice, however, there are probably multiple requests coming from or going to an identical location. Considering the road networks with the integratable nodes as above, we define a new model based on the integrated nodes for the corresponding PPDSP and propose a novel mixed-integer formulation. In comparative experiments with the existing formulation, as the number of integratable nodes increases, our method has a clear advantage in terms of the number of variables as well as the number of constraints in the generated instances, and the accuracy of the optimized solution obtained within a given time.

Keywords: Combinatorial optimization · Mixed-integer programming · Logistics.

1 Introduction

With the increasing expansion of the supply chain, logistics has become the backbone of the modern economy. Adequate transportation capacity is a necessary condition for commerce. However, during the peak period of logistics order trading, the shortage of transportation resources still occurs from time to time. As an operator (i.e., transportation company), it is a challenge to provide limited transportation resources to some of the requests in a competitive market in order to maximize profit.

Recently, the profit-maximizing multi-vehicle pickup and delivery selection problem (PPDSP) has gained a lot of attention in the field of practical transportation and logistics, such as *sensor transportation* [8], *dial-a-ride servise* [11], *green pickup-delivery* [3], and *customized bus network design* [7]. This problem was first proposed in [10], which involves three classical problem models: *routing optimization*, *pickup and delivery*, and *selective pickup* (a.k.a. *knapsack*). Solving this problem quickly and optimally can both help improve the operational efficiency of the carriers and contributes to more eco-friendly transportation.

For solving PPDSP, previous studies have presented a *branch-and-price algorithm* [1], an exact method of graph search [4], and several metaheuristic methods including *tabu search*, *genetic algorithm*, *scatter search* [12], *simulated annealing algorithm* [2], and *variable neighborhood search* [6]. To the best of our knowledge, in the graph-theoretic models constructed in the existing studies, the definitions of nodes are based on the pickup and delivery location from the requests. This means that one request needs to correspond to two nodes. However, in application scenarios, there are often plural requests coming from or to arrive at the same location. The number of variables to be required in the existing mixed-integer formulation heavily depends on the number of nodes in the model, and since PPDSP is an $\mathcal{NP}$-hard problem, its computational complexity is exponential with respect to the number of variables.

The motivation of this study is to provide a more reasonable and effective mathematical model for practical logistics and transportation problems. Considering the road networks with the integratable nodes as above, we proposes a new concise model in which the definition of nodes is based on the locations rather than the requests, and give a tailored mixed-integer programming formulation that reduces the number of required variables. It is necessary to claim that the solution set of our proposed location-based method is a subset of the solution set of the request-based method. This is because the integrated node can only be passed at most once under the *Hamiltonian cycle* constraint, where "at most" is due to the objective function of profit-maximizing (i.e., it is possible that any node will not be traversed).

In Fig. 1, we take the delivery of single truck as an example, and assume that the truck is transported according to the route given in Fig. 1a as a feasible solution in this scenario, where the pickup coordinates of request 1 (abbr. req. 1 pickup coord.) and req. 2 pickup coord. are the identical locations. We can integrate them as location 1 as shown in Fig. 1b, and after integrating all the same coordinates into one location respectively, our model becomes more concise. This integration changes the nodes of the model from repeatable coordinates based on the request to the unique locations, which results in each location can only be visited once. Therefore, the location-based model can also correspond to the feasible solutions in Fig. 1a, but not to some of the feasible solutions corresponded to the request-based model, such as the solution given in Fig. 1c. Whereas in real long-distance logistics application scenarios, making multiple retraces to the same location is rare (e.g., insufficient vehicle capacity, goods cannot be mixed, etc.), and also non-efficient. For such locations with a large number of requests or high loading volume, it is a more common strategy to assign multiple vehicles to them. Therefore, we argue that the location-based model can improve the optimization efficiency although it reduces the range of feasible solutions.

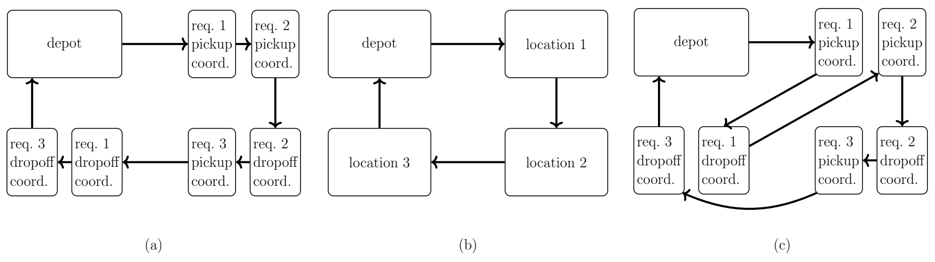

Fig. 1: A simple example for explaining the relationship between the request-based model (i.e., Fig. 1a and 1c) and the location-based model (i.e., Fig. 1b), where {req. 1 pickup coord.} and {req. 2 pickup coord.}, {req. 3 pickup coord.} and {req. 2 dropoff coord.}, and {req. 3 dropoff coord.} and {req. 1 dropoff coord.} can be integrated as {location 1}, {location 2} and {location 3}, respectively. The route shown in Fig. 1c can be regarded as a feasible solution of the request-based model, but cannot be corresponded to in the location-based model.

It should be emphasized that although the location-based model has been widely used on the vehicle routing problem, there is no existing study that employs the location-based model on the pickup and delivery probem because it is relatively difficult to formulate the location-based capacity constraint, especially for PPDSP. We propose a novel formulation for the location-based model on PPDSP and conduct comparative experiments with the existing formulation through simulations, which is the main contribution of our study.

2 Preliminaries

Given a directed graph $G = (V, E)$, where $V = \{0, 1, 2, \ldots, |V|\}$ is the set of nodes representing the location points (0 is depot) and E is the set of arc, denoted as $\overset{\frown}{od}$, and given a set of trucks $T = \{1, 2, \ldots, |T|\}$, we define a type of Boolean variables x^t_{od} that is equal to 1 if the truck t passes through the arc $\overset{\frown}{od}$ and 0 otherwise. We denote the load capacity of truck t as c^t, and the cost of the truck t traversing the arc $\overset{\frown}{od}$ as l^t_{od}, where $t \in T$ and $o, d \in V$.

Let $R = \{1, 2, \ldots, |R|\}$ be a set of requests. Each request r ($r \in R$) is considered as a tuple $r = \langle w_r, q_r, f(r), g(r)\rangle$, where

- w_r is the the payment that can be received for completing the shipping of request r;
- q_r is the volume of request r;
- $f(r)$ is the loading point of request r;
- $g(r)$ is the unloading point of request r,

and $f : R \to V \setminus \{0\}$ (resp. $g : R \to V \setminus \{0\}$) is a function for mapping the loading (resp. unloading) point of requests r. We also define another type of

Boolean variables y_r^t that is equal to 1 if request r is allocated to truck t and 0 otherwise, where $r \in R$ and $t \in T$. Besides, we denote the number of location points visited by truck t before it reaches v ($v \in V \setminus \{0\}$) as u_v^t.

Definition 1 (Delivery of Truck t). *Delivery of truck t is denoted by $D_t = \bigcup_{r \in R}\{r \mid y_r^t = 1\}$, where $t \in T$.*

Definition 2 (Route of Truck t). *Route of truck t is denoted by $S_t = \bigcup_{o \in V}\bigcup_{d \in V}\{\widehat{od} \mid x_{od}^t = 1\}$, where $t \in T$. S_t satisfies the following conditions if $D_t \neq \emptyset$:*

Hamiltonian Cycle – *Denote $P_t = \{0\} \cup \bigcup_{r \in D_t}\{f(r), g(r)\}$ as the set of location points visited and departed exactly once by truck t, where the predecessor and successor are the same node is not counted (i.e., even if $\widehat{vv} \mid x_{vv}^t = 1$, neither this time can be included in the number of visits or departures of truck t to/from location v);*
– *Ensure that no subtour exists in S_t.*[3]

Loading Before Unloading $\forall i \in D_t, u_{f(i)}^t < u_{g(i)}^t$.

Capacity Limitation *At any time, the total volume of cargo carried by truck t cannot exceed its capacity c^t.*

Definition 3 (Delivery Routing Solution). *Delivery routing solution $DS = \bigcup_{t \in T}\{(D_t, S_t)\}$ that is a partition of R into disjoint and contained D_t with the corresponding S_t:*

$$\forall i, j\ (i \neq j),\ D_i \cap D_j = \emptyset, \quad \bigcup_{D_i \in DS} D_i \subseteq R, \quad \bigcup_{S_i \in DS} S_i \subseteq E.$$

Denote the set of all possible delivery routing solutions as $\Pi(R, T)$.

Definition 4 (Profit-Cost Function). *A profit-cost function assigns a real-valued profit to every D_t: $w : D_t \to \mathbb{R}$ and a cost to every S_t: $l : S_t \to \mathbb{R}$, where $w(D_t) = \sum_{r \in D_t} w_r$ and $l(S_t) = \sum_{\widehat{od} \in S_t} l_{od}^t$. For any delivery routing solution $DS \in \Pi(R, T)$, the value of DS is calculated by*

$$\xi(DS) = \sum_{D_t \in DS} w(D_t) - \sum_{S_t \in DS} l(S_t).$$

In general, a delivery routing solution DS that considers only maximizing profits or minimizing costs is not necessarily an optimal DS. Therefore, we have to find the optimal delivery routing solution that maximizes the sum of the values of profit-cost functions. We define a delivery routing problem in profit-cost function as follows.

[3] The time window constraint included in the existing studies can be regarded as a special MTZ-based formulation [9]. In this study, we omit the time window constraint and instead use the most basic MTZ-formulation to eliminate subtour.

Definition 5 (PPDSP). *For a set of requests and trucks* (R, T)*, a profit-maximizing multi-vehicle pickup and delivery selection problem (PPDSP) is to find the optimal delivery routing solution* DS^* *such that*

$$DS^* \in \arg \max_{DS \in \Pi(R,T)} \xi(DS).$$

Here we show an example of PPDSP.

Example 1. Assume that two trucks $T = \{t_1, t_2\}$ are responsible for three requests $R = \{r_1, r_2, r_3\}$ with the following conditions.

– The information about the requests:

Request	w_r	q_r	$f(r)$	$g(r)$
r_1	13	4	a	c
r_2	7	2	a	b
r_3	4	1	b	c

– The information about the trucks:
  - The capacities of the trucks $c^{t_1} = 6$ and $c^{t_2} = 3$.
  - The cost matrices of each truck through each arc (δ is depot).

$l^{t_1}_{od}$	δ	a	b	c
δ	0	2	2	2
a	2	0	4	7
b	2	4	0	2
c	2	7	2	0

$l^{t_2}_{od}$	δ	a	b	c
δ	0	1	1	1
a	1	0	3	5
b	1	3	0	1
c	1	5	1	0

For all $DS \in \Pi(R, T)$, we have their respective profit-cost function as follows.

$$\begin{aligned}
&\xi(\{(D_{t_1} = \emptyset, S_{t_1} = \emptyset), (D_{t_2} = \emptyset, S_{t_2} = \emptyset)\}) = 0,\\
&\xi(\{(D_{t_1} = \emptyset, S_{t_1} = \emptyset), (D_{t_2} = \{r_2\}, S_{t_2} = \{\overset{\frown}{\delta a}, \overset{\frown}{ab}, \overset{\frown}{b\delta}\})\}) = 2,\\
&\xi(\{(D_{t_1} = \emptyset, S_{t_1} = \emptyset), (D_{t_2} = \{r_3\}, S_{t_2} = \{\overset{\frown}{\delta b}, \overset{\frown}{bc}, \overset{\frown}{c\delta}\})\}) = 1,\\
&\xi(\{(D_{t_1} = \emptyset, S_{t_1} = \emptyset), (D_{t_2} = \{r_2, r_3\}, S_{t_2} = \{\overset{\frown}{\delta a}, \overset{\frown}{ab}, \overset{\frown}{bc}, \overset{\frown}{c\delta}\})\}) = 5,\\
&\xi(\{(D_{t_1} = \{r_1\}, S_{t_1} = \{\overset{\frown}{\delta a}, \overset{\frown}{ac}, \overset{\frown}{c\delta}\}), (D_{t_2} = \emptyset, S_{t_2} = \emptyset)\}) = 2,\\
&\xi(\{(D_{t_1} = \{r_1\}, S_{t_1} = \{\overset{\frown}{\delta a}, \overset{\frown}{ac}, \overset{\frown}{c\delta}\}), (D_{t_2} = \{r_2\}, S_{t_2} = \{\overset{\frown}{\delta a}, \overset{\frown}{ab}, \overset{\frown}{b\delta}\})\}) = 4,\\
&\xi(\{(D_{t_1} = \{r_1\}, S_{t_1} = \{\overset{\frown}{\delta a}, \overset{\frown}{ac}, \overset{\frown}{c\delta}\}), (D_{t_2} = \{r_3\}, S_{t_2} = \{\overset{\frown}{\delta b}, \overset{\frown}{bc}, \overset{\frown}{c\delta}\})\}) = 3,\\
&\xi(\{(D_{t_1} = \{r_1\}, S_{t_1} = \{\overset{\frown}{\delta a}, \overset{\frown}{ac}, \overset{\frown}{c\delta}\}), (D_{t_2} = \{r_2, r_3\}, S_{t_2} = \{\overset{\frown}{\delta a}, \overset{\frown}{ab}, \overset{\frown}{bc}, \overset{\frown}{c\delta}\})\}) = 7,\\
&\xi(\{(D_{t_1} = \{r_2\}, S_{t_1} = \{\overset{\frown}{\delta a}, \overset{\frown}{ab}, \overset{\frown}{b\delta}\}), (D_{t_2} = \emptyset, S_{t_2} = \emptyset)\}) = -1,\\
&\xi(\{(D_{t_1} = \{r_2\}, S_{t_1} = \{\overset{\frown}{\delta a}, \overset{\frown}{ab}, \overset{\frown}{b\delta}\}), (D_{t_2} = \{r_3\}, S_{t_2} = \{\overset{\frown}{\delta b}, \overset{\frown}{bc}, \overset{\frown}{c\delta}\})\}) = 0,\\
&\xi(\{(D_{t_1} = \{r_3\}, S_{t_1} = \{\overset{\frown}{\delta b}, \overset{\frown}{bc}, \overset{\frown}{c\delta}\}), (D_{t_2} = \emptyset, S_{t_2} = \emptyset)\}) = -2,\\
&\xi(\{(D_{t_1} = \{r_3\}, S_{t_1} = \{\overset{\frown}{\delta b}, \overset{\frown}{bc}, \overset{\frown}{c\delta}\}), (D_{t_2} = \{r_2\}, S_{t_2} = \{\overset{\frown}{\delta a}, \overset{\frown}{ab}, \overset{\frown}{b\delta}\})\}) = 0,\\
&\xi(\{(D_{t_1} = \{r_1, r_2\}, S_{t_1} = \{\overset{\frown}{\delta a}, \overset{\frown}{ab}, \overset{\frown}{bc}, \overset{\frown}{c\delta}\}), (D_{t_2} = \emptyset, S_{t_2} = \emptyset)\}) = 10,\\
&\xi(\{(D_{t_1} = \{r_1, r_2\}, S_{t_1} = \{\overset{\frown}{\delta a}, \overset{\frown}{ac}, \overset{\frown}{cb}, \overset{\frown}{b\delta}\}), (D_{t_2} = \emptyset, S_{t_2} = \emptyset)\}) = 7,\\
&\xi(\{(D_{t_1} = \{r_1, r_2\}, S_{t_1} = \{\overset{\frown}{\delta a}, \overset{\frown}{ab}, \overset{\frown}{bc}, \overset{\frown}{c\delta}\}), (D_{t_2} = \{r_3\}, S_{t_2} = \{\overset{\frown}{\delta b}, \overset{\frown}{bc}, \overset{\frown}{c\delta}\})\}) = 11,\\
&\xi(\{(D_{t_1} = \{r_1, r_2\}, S_{t_1} = \{\overset{\frown}{\delta a}, \overset{\frown}{ac}, \overset{\frown}{cb}, \overset{\frown}{b\delta}\}), (D_{t_2} = \{r_3\}, S_{t_2} = \{\overset{\frown}{\delta b}, \overset{\frown}{bc}, \overset{\frown}{c\delta}\})\}) = 8,\\
&\xi(\{(D_{t_1} = \{r_1, r_3\}, S_{t_1} = \{\overset{\frown}{\delta a}, \overset{\frown}{ab}, \overset{\frown}{bc}, \overset{\frown}{c\delta}\}), (D_{t_2} = \emptyset, S_{t_2} = \emptyset)\}) = 7,
\end{aligned}$$

$\xi(\{(D_{t_1} = \{r_1, r_3\}, S_{t_1} = \{\widehat{\delta b}, \widehat{ba}, \widehat{ac}, \widehat{c\delta}\}), (D_{t_2} = \emptyset, S_{t_2} = \emptyset)\}) = 2,$
$\xi(\{(D_{t_1} = \{r_1, r_3\}, S_{t_1} = \{\widehat{\delta a}, \widehat{ab}, \widehat{bc}, \widehat{c\delta}\}), (D_{t_2} = \{r_2\}, S_{t_2} = \{\widehat{\delta a}, \widehat{ab}, \widehat{b\delta}\})\}) = 9,$
$\xi(\{(D_{t_1} = \{r_1, r_3\}, S_{t_1} = \{\widehat{\delta b}, \widehat{ba}, \widehat{ac}, \widehat{c\delta}\}), (D_{t_2} = \{r_2\}, S_{t_2} = \{\widehat{\delta a}, \widehat{ab}, \widehat{b\delta}\})\}) = 4,$
$\xi(\{(D_{t_1} = \{r_2, r_3\}, S_{t_1} = \{\widehat{\delta a}, \widehat{ab}, \widehat{bc}, \widehat{c\delta}\}), (D_{t_2} = \emptyset, S_{t_2} = \emptyset)\}) = 1.$

In this example, the optimal delivery routing solution DS^* for the given PPDSP is $\{(D_{t_1} = \{r_1, r_2\}, S_{t_1} = \{\widehat{\delta a}, \widehat{ab}, \widehat{bc}, \widehat{c\delta}\}), (D_{t_2} = \{r_3\}, S_{t_2} = \{\widehat{\delta b}, \widehat{bc}, \widehat{c\delta}\})\}$, and its value is 11.

3 Problem Formulation

In this section, we present the following MIP formulation of the location-based model for PPDSP and prove its correctness as well as the space complexity of the generated problem.

$$\max \sum_{r\in R}\sum_{t\in T}(w_r \cdot y_r^t) - \sum_{t\in T}\sum_{o\in V}\sum_{d\in V}(l_{od}^t \cdot x_{od}^t), \tag{1}$$

$$\text{s.t. } x_{od}^t, y_r^t \in \{0,1\}, \qquad \forall(t,o,d,r) : t\in T, o,d\in V, r\in R, \tag{2}$$

$$\sum_{t\in T} y_r^t \le 1, \qquad \forall r : r\in R, \tag{3}$$

$$y_r^t \le \sum_{\substack{o\in V\\ o\neq f(r)}} x_{of(r)}^t, \qquad \forall(t,r) : t\in T, r\in R, \tag{4}$$

$$y_r^t \le \sum_{\substack{o\in V\\ o\neq g(r)}} x_{og(r)}^t, \qquad \forall(t,r) : t\in T, r\in R, \tag{5}$$

$$\sum_{d\in V} x_{od}^t - \sum_{d\in V} x_{do}^t = 0, \qquad \forall(t,o) : t\in T, o\in V, \tag{6}$$

$$\sum_{\substack{d\in V\\ d\neq o}} x_{od}^t \le 1, \qquad \forall(t,o) : t\in T, o\in V, \tag{7}$$

$$u_d^t - u_o^t \ge 1 - |V|(1 - x_{od}^t), \qquad \forall(t,o,d) : t\in T, o,d\in V\setminus\{0\}, o\neq d, \tag{8}$$

$$u_{f(r)}^t - u_{g(r)}^t < |V|(1 - y_r^t), \qquad \forall(t,r) : t\in T, r\in R, \tag{9}$$

$$\begin{cases} \mathit{\Gamma} - M\cdot(1 - x_{od}^t) \le h_d^t - h_o^t \le \mathit{\Gamma} + M\cdot(1 - x_{od}^t), \\ \mathit{\Gamma} \triangleq \sum_{r\in f^{-1}(d)}(q_r\cdot y_r^t) - \sum_{r\in g^{-1}(d)}(q_r\cdot y_r^t), \\ M \triangleq c^t + \sum_{r\in R} q_r, \end{cases}, \qquad \forall(t,o,d) : t\in T, o,d\in V\setminus\{0\}, o\neq d, \tag{10}$$

$$0 \le h_v^t \le c^t, \qquad \forall(t,v) : t\in T, v\in V\setminus\{0\}, \tag{11}$$

$$0 \le u_v^t \le |V| - 2, \qquad \forall(t,v) : t\in T, v\in V\setminus\{0\}. \tag{12}$$

3.1 Correctness

The objective function in Eq. (1) maximizes the profit-cost function for all possible delivery routing solutions. Eq. (2) introduces two types of Boolean variables x_{od}^t and y_r^t. Eq. (3) guarantees that, each request can be assigned to at most one truck.

Theorem 1. *For PPDSP, the Hamiltonian cycle constraints can be guaranteed by the simultaneous Eqs.* (4)–(8) *and Eq.* (12)*.*

Proof. According to Eq. (4) (resp. Eq. (5)), if request r is assigned to truck t, then truck t reaches the pickup (resp. dropoff) point of r at least once via a point that is not the pickup (resp. dropoff) point of r. Eq. (6) ensures that the number of visits and the number of departures of a truck at any location must be equal. Eq. (7) restricts any truck departing from a location to at most one other location. It is clear from the above that, all nodes of $\bigcup_{r \in D_t}\{f(r), g(r)\}$ are ensured to be visited and departed by truck t exactly once.

Eq. (8), a canonical MTZ subtour elimination constraint [9], resrticts the integer variables u_v^t, where $v \in V \setminus \{0\}$, such that they form an ascending series to represent the order that truck t arrives at each location v. Eq. (12) gives the domain of such variables u_v^t. Furthermore, Eq. (8) associated with Eqs. (4)–(7) also enforces that depot 0 must be visited and departed by truck t exactly once if $D_t \neq \emptyset$. $\square$

Eq. (9) ensures that, for any request r, the arrival of its pickup point must precede the arrival of its dropoff point by truck t if $y_r^t = 1$.

Theorem 2. *For PPDSP, the capacity constraint can be guaranteed by the Eqs.* (10) *and* (11)*.*

Proof. We introduce another integer variables h_v^t, where $v \in V \setminus \{0\}$, in Eq. (10), whose domains are between 0 and c^t (i.e., the capacity of truck t) as given in Eq. (11). Such a variable h_v^t can be considered as the loading volume of truck t when it departs at location v. We define the amount of change in the loading of truck t at location v as Γ, which equals the loaded amount at v (i.e., $\sum_{r \in f^{-1}(v)}(q_r \cdot y_r^t)$) minus the unloaded amount at v (i.e., $\sum_{r \in g^{-1}(v)}(q_r \cdot y_r^t)$). The main part of Eq. (10) guarantees that $h_d^t - h_o^t$ is exactly equal to Γ when $x_{od}^t = 1$, which is written as a big-M linear inequality, where we specify M as $c^t + \sum_{r \in R} q_r$. $\square$

3.2 Space Complexity

Consider that the number of trucks is m (i.e., $|T| = m$) and the number of requests is n (i.e., $|R| = n$). Assume that the average repetition rate of the same locations is k, where $k \geq 1$, then we can estimate the number of unique locations as $2nk^{-1}$, which is also the number of integrated nodes (viz., $|V \setminus \{0\}| = 2nk^{-1}$).

Theorem 3. *The number of linear (in)equations corresponding to Eqs.* (3)–(10) *is always of a pseudo-polynomial size, and both this number and the number of required variables are bounded by* $\Theta(mn^2k^{-2})$*.*

Table 1: The bounded numbers of linear (in)equations corresponding to the constraints that are formulated in Eqs. (3)–(10).

Constraint	#(In)equations	Constraint	#(In)equations
Eq. (3)	n	Eq. (7)	$m(1+2nk^{-1})$
Eq. (4)	mn	Eq. (8)	$m\binom{2nk^{-1}}{2}$
Eq. (5)	mn	Eq. (9)	mn
Eq. (6)	$m(1+2nk^{-1})$	Eq. (10)	$2m\binom{2nk^{-1}}{2}$

Proof. The number of Boolean variables x_{od}^t (resp. y_r^t) required in proposed formulation is $m(1+2nk^{-1})^2$ (resp. mn); while both the number of required integer variables u_d^t and h_d^t are $m(1+2nk^{-1})$. In Table 1, we list the bounded numbers of linear (in)equations that correspond to Eqs. (3)–(10) involved in the proposed PPDSP formulation. Therefore, the number of required variable as well as the number of corresponded linear (in)equations are of $\Theta(mn^2k^{-2})$. □

4 Experiment

We compare the performance of a MIP optimizer in solving randomly generated PPDSP instances based on the proposed formulation (i.e., location-based model), and based on the existing formulation (i.e., request-based model), respectively. Due to page limitations of the submission, please refer to the appendix section in the preprint version of this manuscript [13] for the specific description of the existing formulation used for the comparative experiments.

4.1 Simulation Instances Generation

The directed graph informations for generating instances are set based on the samples of TSPLIB benchmark with displaying data as the coordinates of nodes.[4] We denote the number of nodes contained in the selected sample as $|V|$, and set the first of these nodes to be the depot node. The pickup and dropoff points of each request are chosen from the non-depot nodes. Therefore, given an average repetition rate k for the same locations, we need to repeatably select n pairs of non-depot nodes (i.e., a total of $2n$ repeatable non-depot nodes) to correspond to n requests, where $n = \text{round}(\frac{k(|V|-1)}{2})$.

In Algorithm 1, we construct the *repeaTimeList* of length $|V|-1$ to mark the number of times of each non-depot node being selected. In order to ensure that each non-depot node is selected at least once, each value on such list is initialized to 1. We continuously generate a random index of *repeaTimeList* and add one to the value corresponding to the generated index (i.e., the number of times of the non-depot node being selected increases by one) until the sum of these numbers reaches $2n$.

[4] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/

Algorithm 1: Randomly specify the number of repetitions of each non-depot node

```
   Input: $G = (V, E)$, $k$
   Init. : $n \leftarrow \mathrm{round}(\frac{k(|V|-1)}{2})$, $repeaTimeList \leftarrow [|V|-1 \text{ of } 1]$
1  while $\sum_i repeaTimeList[i] < 2n$ do
2      $i \leftarrow \mathrm{round}(\textsc{RandomUniform}(0, |V|-2))$
3      $repeaTimeList[i] \leftarrow repeaTimeList[i] + 1$
4  return $repeaTimeList$
```

The pickup and dropoff nodes for each request are randomly paired up in Algorithm 2. We first rewrite *repeaTimeList* as a list of length $2n$, *shuffList*, which consists of all selected non-depot nodes, where the number of repetitions indicates the number of times they are selected. For example, we have $shuffList = [0, 0, 0, 1, 2, 2]$ for $repeaTimeList = [3, 1, 2]$. Next, we shuffle *shuffList* and clear *pairList*, which is used to store pairs of nodes indicating the pickup and dropoff points of all requests. We then divide *shuffList* into n pairs in order. If two nodes of any pair are identical (i.e., the pickup and dropoff are the same point), or the pair with considering order is already contained in *pairList*, then we are back to Line 4. Otherwise, we append the eligible pair of nodes into *pairList* until the last pair is added.

Algorithm 2: Randomly pair up the pickup and dropoff nodes for each request

```
    Input: $G = (V, E)$, $n = \mathrm{round}(\frac{k(|V|-1)}{2})$, $repeaTimeList$
    Init. : $shuffList \leftarrow [\,]$, $pairList \leftarrow [\,]$, $reshuffle \leftarrow$ true
1   for $i \leftarrow 0$ to $|V| - 2$ do
2       $shuffList.\textsc{Extend}([repeaTimeList[i] \text{ of } i])$
3   while $reshuffle$ do
4       $\textsc{RandomShuffle}(shuffList)$
5       $pairList \leftarrow [\,]$
6       for $i \leftarrow 0$ to $n - 1$ do
7           if $shuffList[2i] = shuffList[1 + 2i]$ or
             $[shuffList[2i], shuffList[1 + 2i]]$ in $pairList$ then
8               break
9           else
10              $pairList.\textsc{Append}([shuffList[2i], shuffList[1 + 2i]])$
11              if $i = n - 1$ then $reshuffle \leftarrow$ false
12  return $pairList$
```

Since we intend to generate instances corresponding to different k for each selected sample of TSPLIB benchmark, and the number of requests n gets smaller as k decreases, we need to produce a decrementable list of node pairs such that we can remove some of the node pairs to correspond to smaller k, but the remaining part of the node pairs still contains all locations in the sample other than that as depot (i.e., they still need to be selected at least once). Therefore, in Algorithm 3, we assume that a sorted list of node pairs pops the end elements out of it one by one, yet it is always guaranteed that every non-depot node is selected at least once. We insert the node pair in which both nodes are currently selected only once into the frontmost of *head* in Line 7, and then append the node pair in which one of them is selected only once into *head* in Line 11. Then we keep inserting the node pair in which the sum of the repetitions of the two nodes is currently maximum into the frontmost of *tail* in Line 22. Note that such insertions and appends require popping the corresponding node pair out of *pairList* and updating $\mathcal{L}$. We merge *head* and *tail* to obtain a sorted list of node pairs *sortedPairs* at the end of Algorithm 3.

Algorithm 3: Sort the node pairs by the sum of the repeat times of each node in the node pair

```
   Input: $repeaTimeList$, $pairList$
   Init. : $\mathcal{L} \leftarrow repeaTimeList$, $head \leftarrow [\,]$, $tail \leftarrow [\,]$, $max \leftarrow 0$, $maxIndex \leftarrow -1$,
           $sortedPairs \leftarrow [\,]$
1  while $|pairList| > 0$ do
2      $i \leftarrow 0$
3      while $i < |pairList|$ do
4          if $\mathcal{L}[pairList[i][0]] = 1$ and $\mathcal{L}[pairList[i][1]] = 1$ then
5              $\mathcal{L}[pairList[i][0]] \leftarrow \mathcal{L}[pairList[i][0]] - 1$
6              $\mathcal{L}[pairList[i][1]] \leftarrow \mathcal{L}[pairList[i][1]] - 1$
7              $head.\textsc{Insert}(0, pairList.\textsc{Pop}(i))$
8          else if $\mathcal{L}[pairList[i][0]] = 1$ or $\mathcal{L}[pairList[i][1]] = 1$ then
9              $\mathcal{L}[pairList[i][0]] \leftarrow \mathcal{L}[pairList[i][0]] - 1$
10             $\mathcal{L}[pairList[i][1]] \leftarrow \mathcal{L}[pairList[i][1]] - 1$
11             $head.\textsc{Append}(pairList.\textsc{Pop}(i))$
12         else $i \leftarrow i + 1$
13     $max \leftarrow 0$
14     $maxIndex \leftarrow -1$
15     for $j \leftarrow 0$ to $|pairList| - 1$ do
16         if $\mathcal{L}[pairList[j][0]] + \mathcal{L}[pairList[j][1]] > max$ then
17             $max \leftarrow \mathcal{L}[pairList[j][0]] + \mathcal{L}[pairList[j][1]]$
18             $maxIndex \leftarrow j$
19     if $maxIndex \neq -1$ then
20         $\mathcal{L}[pairList[maxIndex][0]] \leftarrow \mathcal{L}[pairList[maxIndex][0]] - 1$
21         $\mathcal{L}[pairList[maxIndex][1]] \leftarrow \mathcal{L}[pairList[maxIndex][0]] - 1$
22         $tail.\textsc{Insert}(0, pairList.\textsc{Pop}(maxIndex))$
23 $sortedPairs \leftarrow head.\textsc{Extend}(tail)$
24 return $sortedPairs$
```

Parameter settings Finally, we can generate a list containing n requests for each selected sample of TSPLIB benchmark, as shown in Algorithm 4. In addition, we generate data for the three types of trucks recursively in the order of maximum load capacity of 25, 20, and 15 until the number of generated trucks reaches m. And these three types of trucks correspond to their respective cost coefficients for each traversing arc of 1.2, 1, and 0.8. For example, for a truck t of load capacity 15, l^t_{od}, i.e., the cost of it passing through the arc $\overset{\frown}{od}$ is $0.8 \times \textsc{Distance}(\overset{\frown}{od})$.[5] According to the average volume of 5 for each request set in Algorithm 4, it is expected that each truck can accommodate four requests at the same time.

Algorithm 4: Randomly generate the list of requests

```
   Input: $G = (V, E)$, $n = \text{round}(\frac{k(|V|-1)}{2})$, $sortedPairs$
   Init. : $avgDistance \leftarrow \frac{1}{|V|\cdot|V-1|} \sum_{o\in V} \sum_{\substack{d\in V\\ d\neq o}} \textsc{Distance}(\overset{\frown}{od})$, $avgVolume \leftarrow 5$,
           $requestList \leftarrow [\,]$
1  for $r \leftarrow 0$ to $n-1$ do
2      $q_r \leftarrow \text{round}(\textsc{RandomUniform}(1, 2 \times avgVolume - 1))$
3      $w_r \leftarrow \text{round}(2 \times avgDistance \times q_r \div avgVolume)$
4      $f(r) \leftarrow sortedPairs[r][0]$
5      $g(r) \leftarrow sortedPairs[r][1]$
6      $requestList.\textsc{Append}(\langle w_r, q_r, f(r), g(r)\rangle)$
7  return $requestList$
```

We generate a total of $3 \times 5 \times 5 = 75$ PPDSP instances for our comparative experiments based on the proposed formulation and on the existing formulation, respectively, according to the following parameter settings.

- The selected TSPLIB samples are {*burma14*, *ulysses16*, *ulysses22*};
- The average repetition rate of the same locations, $k \in \{1, 1.5, 2, 2.5, 3\}$;
- The number of trucks $m \in \{2, 4, 6, 8, 10\}$.

4.2 Experimental Setup

All experiments are performed on an Apple M1 Pro chip, using the Ubuntu 18.04.6 LTS operating system via the Podman virtual machine with 19 GB of allocated memory. A single CPU core is used for each experiment. We implemented the instance generators for both the proposed formulation and the existing formulation by using Python 3. Each generated problem instance is solved by the MIP optimizer–Cplex of version 20.1.0.0 [5] within 3,600 CPU seconds time limit. Source code for our experiments is available at https://github.com/ReprodSuplem/PPDSP.

[5] $\textsc{Distance}(\overset{\frown}{od})$ is the *Euclidean distance* between the coordinates of node o and the coordinates of node d.

Table 2: Comparison of the existing formulation-based method with the proposed formulation-based method for TSPLIB sample *burma14* in terms of the number of variables and constraints generated, as well as the optimal values achieved.

m	item	$k = 1$ $(n = 7)$		$k = 1.5$ $(n = 10)$		$k = 2$ $(n = 13)$		$k = 2.5$ $(n = 16)$		$k = 3$ $(n = 20)$	
2	#Var.	576	**458**	1056	**464**	1680	**470**	2448	**476**	3696	**484**
	#Con.	**1027**	1041	1942	**1062**	3145	**1083**	4636	**1104**	7072	**1132**
	Opt.	30	30	**42**	40	**56**	53	65	**75**	100	**107**
4	#Var.	1152	**916**	2112	**928**	3360	**940**	4896	**952**	7392	**968**
	#Con.	**2047**	2075	3874	**2114**	6277	**2153**	9256	**2192**	14124	**2244**
	Opt.	**33**	30	**49**	44	58	**61**	53	**77**	68	**103**
6	#Var.	1728	**1374**	3168	**1392**	5040	**1410**	7344	**1428**	11088	**1452**
	#Con.	**3067**	3109	5806	**3166**	9409	**3223**	13876	**3280**	21176	**3356**
	Opt.	**33**	31	**46**	42	57	**58**	43	**81**	46	**103**
8	#Var.	2304	**1832**	4224	**1856**	6720	**1880**	9792	**1904**	14784	**1936**
	#Con.	**4087**	4143	7738	**4218**	12541	**4293**	18496	**4368**	28228	**4468**
	Opt.	**33**	31	**46**	44	56	56	4	**79**	81	**104**
10	#Var.	2880	**2290**	5280	**2320**	8400	**2350**	12240	**2380**	18480	**2420**
	#Con.	**5107**	5177	9670	**5270**	15673	**5363**	23116	**5456**	35280	**5580**
	Opt.	**33**	31	44	44	57	**58**	24	**80**	49	**96**

4.3 Results

Tables 2–4 show the performance of the existing formulation-based method and our proposed formulation-based method in terms of the number of generated variables (#Var.), the number of generated constraints (#Con.), and the optimal values (Opt.) for the various average repetition rates of the same locations (k) and the different numbers of trucks (m), corresponding to TSPLIB samples *burma14*, *ulysses16* and *ulysses22*, respectively. Each cell recording the left and right values corresponds to a comparison item, where the left values refers to the performance of the existing formulation-based method, while the right values corresponds to the performance of the proposed formulation-based method. We compare the performance of these two methods in such cells and put the values of the dominant side in bold.

We can see that either the number of variables or the number of constraints generated by our proposed method is proportional to m, while neither the number of variables nor the number of constraints generated by our proposed method increases significantly as k becomes larger. Such a result is consistent with Theorem 3. In contrast, although the number of variables and the number of constraints produced by the existing method is also proportional to m, with k becoming larger, exponential increases in both of them are observed. Furthermore, it is interesting to note that even when $k = 1$, the proposed method generates fewer variables than the existing method.

As k increases, the optimal value obtained by the proposed method increases within the time limit; while there is no such trend in the optimal values obtained

Table 3: Comparison of the existing formulation-based method with the proposed formulation-based method for TSPLIB sample *ulysses16* in terms of the number of variables and constraints generated, as well as the optimal values achieved.

m	item	$k = 1$ ($n = 8$)		$k = 1.5$ ($n = 11$)		$k = 2$ ($n = 15$)		$k = 2.5$ ($n = 19$)		$k = 3$ ($n = 23$)	
2	#Var.	720	**588**	1248	**594**	2176	**602**	3360	**610**	4800	**618**
	#Con.	**1300**	1380	2311	**1401**	4107	**1429**	6415	**1457**	9235	**1485**
	Opt.	94	94	**96**	94	109	**112**	157	**187**	168	**230**
4	#Var.	1440	**1176**	2496	**1188**	4352	**1204**	6720	**1220**	9600	**1236**
	#Con.	**2592**	2752	4611	**2791**	8199	**2843**	12811	**2895**	18447	**2947**
	Opt.	**99**	98	**104**	101	104	**124**	140	**198**	29	**247**
6	#Var.	2160	**1764**	3744	**1782**	6528	**1806**	10080	**1830**	14400	**1854**
	#Con.	**3884**	4124	6911	**4181**	12291	**4257**	19207	**4333**	27659	**4409**
	Opt.	**99**	98	**105**	101	104	**139**	92	**205**	0	**226**
8	#Var.	2880	**2352**	4992	**2376**	8704	**2408**	13440	**2440**	19200	**2472**
	#Con.	**5176**	5496	9211	**5571**	16383	**5671**	25603	**5771**	36871	**5871**
	Opt.	**99**	98	91	**98**	89	**138**	75	**176**	96	**241**
10	#Var.	3600	**2940**	6240	**2970**	10880	**3010**	16800	**3050**	24000	**3090**
	#Con.	**6468**	6868	11511	**6961**	20475	**7085**	31999	**7209**	46083	**7333**
	Opt.	**99**	98	96	**101**	111	**130**	82	**186**	27	**221**

by the existing method. In addition, as m grows up, in theory, the upper bound of the optimal value cannot be smaller. However, the optimal values obtained by the proposed method and the existing method do not maintain the trend of increasing, instead, they both have inflection points. This is due to the fact that the search space of the problem becomes huge, which makes it inefficient for the solver to update the optimized solution. There are even cases on the existing method where the initial solution is not obtained until the end of time.

Last but not least, we can clearly see that for those problems with larger k (i.e., more nodes that can be integrated), the method based on our proposed formulation generates fewer variables and constraints, as well as achieves larger optimal values, than the method based on the existing formulation.

4.4 Discussion

In a real-world scenario, different logistics orders may be sent from the same place to various places, or from different places to the same place. With a certain range of shipping services, as the number of orders increases, the average repetition rate (k) of pickup locations as well as delivery locations becomes higher. From the experimental results, we can see that, although the proposed method is more efficient than the existing methods, PPDSP is still difficult to be solved exactly, even for problem instances generated based on small-scale road networks in the limited time of 3,600 CPU seconds to find the optimal solution. We do not consider the time window constraint in this paper because

Table 4: Comparison of the existing formulation-based method with the proposed formulation-based method for TSPLIB sample *ulysses22* in terms of the number of variables and constraints generated, as well as the optimal values achieved.

m	item	$k = 1$		$k = 1.5$		$k = 2$		$k = 2.5$		$k = 3$	
		$(n = 11)$		$(n = 16)$		$(n = 21)$		$(n = 26)$		$(n = 32)$	
2	#Var.	1248	**1074**	2448	**1084**	4048	**1094**	6048	**1104**	8976	**1116**
	#Con.	**2311**	2685	4636	**2720**	7761	**2755**	11686	**2790**	17452	**2832**
	Opt.	116	**120**	161	**206**	182	**225**	85	**297**	79	**336**
4	#Var.	2496	**2148**	4896	**2168**	8096	**2188**	12096	**2208**	17952	**2232**
	#Con.	**4611**	5359	9256	**5424**	15501	**5489**	23346	**5554**	34872	**5632**
	Opt.	**134**	120	141	**185**	53	**216**	34	**277**	15	**327**
6	#Var.	3744	**3222**	7344	**3252**	12144	**3282**	18144	**3312**	26928	**3348**
	#Con.	**6911**	8033	13876	**8128**	23241	**8223**	35006	**8318**	52292	**8432**
	Opt.	115	**125**	136	**192**	0	**227**	45	**242**	0	**343**
8	#Var.	4992	**4296**	9792	**4336**	16192	**4376**	24192	**4416**	35904	**4464**
	#Con.	**9211**	10707	18496	**10832**	30981	**10957**	46666	**11082**	69712	**11232**
	Opt.	**124**	117	102	**206**	25	**224**	58	**228**	41	**276**
10	#Var.	6240	**5370**	12240	**5420**	20240	**5470**	30240	**5520**	44880	**5580**
	#Con.	**11511**	13381	23116	**13536**	38721	**13691**	58326	**13846**	87132	**14032**
	Opt.	**124**	123	31	**173**	44	**204**	42	**243**	44	**319**

adding more constraints would make the solution of the problem more difficult. Nonetheless, for solving larger scale PPDSP within a reasonable computation time, we still need to tailor heuristic algorithms for the location-based model in our future work. Compared to the request-based model, we propose a surrogate optimization model for PPDSP, whose solution set is a subset of the former. This may result in a lower upper bound on the optimized value of exact solutions. However, our approach remains competitive in obtaining approximate solutions.

5 Conclusion

In this paper, we revisit PPDSP on road networks with the integratable nodes. For such application scenarios, we define a location-based graph-theoretic model and give the corresponding MIP formulation. We prove the correctness of this formulation as well as analyze its space complexity. We compare the proposed method with the existing formulation-based method. The experimental results show that, for the instances with more integratable nodes, our method has a significant advantage over the existing method in terms of the generated problem size, and the optimized values.

Acknowledgements The authors are grateful for the support sponsors from *the Progressive Logistic Science Laboratory*, the Research Center for Advanced Science and Technology, the University of Tokyo.

References

1. Ahmadi-Javid, A., Amiri, E., Meskar, M.: A profit-maximization location-routing-pricing problem: A branch-and-price algorithm. Eur. J. Oper. Res. **271**(3), 866–881 (2018). https://doi.org/10.1016/j.ejor.2018.02.020
2. Al-Chami, Z., Flity, H.E., Manier, H., Manier, M.: A new metaheuristic to solve a selective pickup and delivery problem. In: Boukachour, J., Sbihi, A., Alaoui, A.E.H., Benadada, Y. (eds.) 4th International Conference on Logistics Operations Management, GOL 2018, Le Havre, France, April 10-12, 2018. pp. 1–5. IEEE (2018). https://doi.org/10.1109/GOL.2018.8378089
3. Asghari, M., Mirzapour Al-e-hashem, S.M.J.: A green delivery-pickup problem for home hemodialysis machines; sharing economy in distributing scarce resources. Transportation Research Part E: Logistics and Transportation Review **134**, 101815 (2020). https://doi.org/https://doi.org/10.1016/j.tre.2019.11.009
4. Bruni, M.E., Toan, D.Q., Nam, L.H.: The multi-vehicle profitable pick up and delivery routing problem with uncertain travel times. Transportation research procedia **52**, 509–516 (2021). https://doi.org/10.1016/j.trpro.2021.01.060
5. CPLEX, I.I.: IBM ILOG CPLEX Optimizer. https://www.ibm.com/analytics/cplex-optimizer (2020)
6. Gansterer, M., Küçüktepe, M., Hartl, R.F.: The multi-vehicle profitable pickup and delivery problem. OR Spectr. **39**(1), 303–319 (2017). https://doi.org/10.1007/s00291-016-0454-y
7. Huang, D., Gu, Y., Wang, S., Liu, Z., Zhang, W.: A two-phase optimization model for the demand-responsive customized bus network design. Transportation Research Part C: Emerging Technologies **111**, 1–21 (2020). https://doi.org/https://doi.org/10.1016/j.trc.2019.12.004
8. Liu, C., Aleman, D.M., Beck, J.C.: Modelling and solving the senior transportation problem. In: van Hoeve, W.J. (ed.) Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 15th International Conference, CPAIOR 2018, Delft, The Netherlands, June 26-29, 2018, Proceedings. Lecture Notes in Computer Science, vol. 10848, pp. 412–428. Springer (2018). https://doi.org/10.1007/978-3-319-93031-2_30
9. Miller, C.E., Tucker, A.W., Zemlin, R.A.: Integer programming formulation of traveling salesman problems. J. ACM **7**(4), 326–329 (1960). https://doi.org/10.1145/321043.321046
10. Qiu, X., Feuerriegel, S., Neumann, D.: Making the most of fleets: A profit-maximizing multi-vehicle pickup and delivery selection problem. Eur. J. Oper. Res. **259**(1), 155–168 (2017). https://doi.org/10.1016/j.ejor.2016.10.010
11. Riedler, M., Raidl, G.R.: Solving a selective dial-a-ride problem with logic-based benders decomposition. Comput. Oper. Res. **96**, 30–54 (2018). https://doi.org/10.1016/j.cor.2018.03.008
12. Ting, C., Liao, X., Huang, Y., Liaw, R.: Multi-vehicle selective pickup and delivery using metaheuristic algorithms. Inf. Sci. **406**, 146–169 (2017). https://doi.org/10.1016/j.ins.2017.04.001
13. Zha, A., Chang, Q., Imura, N., Nishinari, K.: A case study of the profit-maximizing multi-vehicle pickup and delivery selection problem for the road networks with the integratable nodes (2022). https://doi.org/10.48550/arXiv.2208.14866