A case study of the profit-maximizing multi-vehicle pickup and delivery selection problem for the road networks with the integratable nodes

Aolong Zha\textsuperscript{a,}\textsuperscript{*}, Qiong Chang\textsuperscript{b}, Naoto Imura\textsuperscript{a}, Katsuhiro Nishinari\textsuperscript{a}

\textsuperscript{a}Research Center for Advanced Science and Technology, The University of Tokyo
\textsuperscript{b}School of Computing, Tokyo Institute of Technology

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 required in the generated instances, and the accuracy of the optimized solution obtained within a given time.

Keywords: Integer programming, Optimization, Transportation

1. Introduction
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 Liu et al. (2018); Riedler & Raidl (2018); Asghari & Mirzapour Al-e-hashem (2020); Huang et al. (2020). This problem was first proposed in Qiu et al. (2017), 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.

To the best of our knowledge, in the graph-theoretic models constructed in the existing studies Ting et al. (2017); Gansterer et al. (2017); Al-Chami et al. (2018); Ahmadi-Javid et al.\textsuperscript{*}Corresponding author

Email addresses: a-zha@g.ecc.u-tokyo.ac.jp (Aolong Zha), q.chang@c.titech.ac.jp (Qiong Chang), nimura@g.ecc.u-tokyo.ac.jp (Naoto Imura), tknishi@mail.ecc.u-tokyo.ac.jp (Katsuhiro Nishinari)
Figure 1: A simple example for explaining the relationship between the request-based model (i.e., Figures 1a and 1c) and the location-based model (i.e., Figure 1b), where \{Req. 1 pickup coord.\} and \{Req. 2 pickup coord.\}, \{Req. 3 pickup coord.\} and \{Req. 2 dropoff coord.\}, and \{Req. 1 dropoff coord.\} and \{Req. 3 dropoff coord.\} can be integrated as \{location 1\}, \{location 2\} and \{location 3\}, respectively. The route shown in Figure 1c can be regarded as a feasible solution of the request-based model, but cannot be corresponded to in the location-based model.

(2018), 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 propose a new concise model in which the definition of nodes is based on the locations rather than the requests, and give a novel 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 Figure 1, we take the delivery of single truck as an example, and assume that the truck is transported according to the route given in Figure 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 Figure 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 Figure 1a, but not to some of the feasible solutions corresponded to the request-based model, such as the solution given in Figure 1c. Whereas in real 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.

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 $\bar{od}$, 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 $\bar{od}$ and 0 otherwise. We denote the load capacity of truck $t$ as $c^t$, and the cost of the truck $t$ traversing the arc $\bar{od}$ 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 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 $y^t_r$ 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 $u^t_v$.

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

**Definition 2 (Route of Truck $t$).** Route of truck $t$ is denoted by $S^t = \bigcup_{o \in V} \bigcup_{d \in V} \{\bar{od} \mid x^t_{od} = 1\}$, 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 (e.g., even if $\bar{vv} \mid x^t_{vv} = 1$, 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 $S^t$.\(^1\)

\(^1\)Existing studies usually include the constraints on time windows, which can eliminate subtour. In this study, in order to compare the request-based model with the location-based model, common parts of both models are omitted (e.g., the time windows constraints). Instead, we use the most basic MTZ-formulation to eliminate subtour.
Loading Before Unloading \(\forall i \in D_t, u^t_{f(i)} < u^t_{g(i)}\).

**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 \(S_t\):

\[
\forall i, j (i \neq j), D_i \cap D_j = \emptyset, \quad \bigcup_{D_t \in DS} D_t \subseteq R, \quad \bigcup_{S_t \in DS} S_t \subseteq E.
\]

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 \(l(S_t) = \sum_{od \in S_t} l_{od}\). 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.

**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 $c^1 = 6$ and $c^2 = 3$.
- The cost matrices of each truck through each arc ($\delta$ is depot).

| $\delta$ | a | b | c | $\delta$ | a | b | c |
|------|----|----|----|------|----|----|----|
| 0    | 2  | 2  | 2  | 0    | 1  | 1  | 1  |
| 2    | 0  | 4  | 7  | 2    | 1  | 3  | 0  |
| b    | 2  | 4  | 0  | 2    | 0  | 0  | 0  |
| c    | 2  | 7  | 2  | 0    | 1  | 5  | 1  |

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

\[
\begin{align*}
\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} = \{\hat{a}, \hat{ab}, \hat{b}\})\}) = 2, \\
\xi & (\{(D_{t_1} = \emptyset, S_{t_1} = \emptyset), (D_{t_2} = \{r_3\}, S_{t_2} = \{\hat{b}, \hat{bc}, \hat{c}\})\}) = 1, \\
\xi & (\{(D_{t_1} = \emptyset, S_{t_1} = \emptyset), (D_{t_2} = \{r_2, r_3\}, S_{t_2} = \{\hat{a}, \hat{ab}, \hat{bc}, \hat{c}\})\}) = 5, \\
\xi & (\{(D_{t_1} = \{r_1\}, S_{t_1} = \{\hat{a}, \hat{ac}, \hat{c}\}, (D_{t_2} = \emptyset, S_{t_2} = \emptyset)\}) = 2, \\
\xi & (\{(D_{t_1} = \{r_1\}, S_{t_1} = \{\hat{a}, \hat{ac}, \hat{c}\}, (D_{t_2} = \{r_2\}, S_{t_2} = \{\hat{a}, \hat{ab}, \hat{b}\})\}) = 4, \\
\xi & (\{(D_{t_1} = \{r_1\}, S_{t_1} = \{\hat{a}, \hat{ac}, \hat{c}\}, (D_{t_2} = \{r_3\}, S_{t_2} = \{\hat{b}, \hat{bc}, \hat{c}\})\}) = 3, \\
\xi & (\{(D_{t_1} = \{r_1\}, S_{t_1} = \{\hat{a}, \hat{ac}, \hat{c}\}, (D_{t_2} = \{r_2, r_3\}, S_{t_2} = \{\hat{a}, \hat{ab}, \hat{bc}, \hat{c}\})\}) = 7, \\
\xi & (\{(D_{t_1} = \{r_2\}, S_{t_1} = \{\hat{a}, \hat{ab}, \hat{b}\}, (D_{t_2} = \emptyset, S_{t_2} = \emptyset)\}) = -1, \\
\xi & (\{(D_{t_1} = \{r_2\}, S_{t_1} = \{\hat{a}, \hat{ab}, \hat{b}\}, (D_{t_2} = \{r_3\}, S_{t_2} = \{\hat{b}, \hat{bc}, \hat{c}\})\}) = 0, \\
\xi & (\{(D_{t_1} = \{r_3\}, S_{t_1} = \{\hat{b}, \hat{bc}, \hat{c}\}, (D_{t_2} = \emptyset, S_{t_2} = \emptyset)\}) = -2, \\
\xi & (\{(D_{t_1} = \{r_3\}, S_{t_1} = \{\hat{b}, \hat{bc}, \hat{c}\}, (D_{t_2} = \{r_2\}, S_{t_2} = \{\hat{a}, \hat{ab}, \hat{b}\})\}) = 0, \\
\xi & (\{(D_{t_1} = \{r_1, r_2\}, S_{t_1} = \{\hat{a}, \hat{ab}, \hat{bc}, \hat{c}\}, (D_{t_2} = \emptyset, S_{t_2} = \emptyset)\}) = 10, \\
\xi & (\{(D_{t_1} = \{r_1, r_2\}, S_{t_1} = \{\hat{a}, \hat{ac}, \hat{c}, \hat{b}\}, (D_{t_2} = \emptyset, S_{t_2} = \emptyset)\}) = 7, \\
\xi & (\{(D_{t_1} = \{r_1, r_2\}, S_{t_1} = \{\hat{a}, \hat{ac}, \hat{c}, \hat{b}\}, (D_{t_2} = \{r_3\}, S_{t_2} = \{\hat{b}, \hat{bc}, \hat{c}\})\}) = 11, \\
\xi & (\{(D_{t_1} = \{r_1, r_2\}, S_{t_1} = \{\hat{a}, \hat{ac}, \hat{c}, \hat{b}\}, (D_{t_2} = \{r_1, r_3\}, S_{t_2} = \{\hat{b}, \hat{bc}, \hat{c}\})\}) = 8, \\
\xi & (\{(D_{t_1} = \{r_1, r_3\}, S_{t_1} = \{\hat{a}, \hat{ab}, \hat{bc}, \hat{c}\}, (D_{t_2} = \emptyset, S_{t_2} = \emptyset)\}) = 7, \\
\xi & (\{(D_{t_1} = \{r_1, r_3\}, S_{t_1} = \{\hat{b}, \hat{ba}, \hat{ac}, \hat{c}\}, (D_{t_2} = \emptyset, S_{t_2} = \emptyset)\}) = 2, \\
\xi & (\{(D_{t_1} = \{r_1, r_3\}, S_{t_1} = \{\hat{a}, \hat{ab}, \hat{bc}, \hat{c}\}, (D_{t_2} = \{r_2\}, S_{t_2} = \{\hat{a}, \hat{ab}, \hat{b}\})\}) = 9, \\
\xi & (\{(D_{t_1} = \{r_1, r_3\}, S_{t_1} = \{\hat{b}, \hat{ba}, \hat{ac}, \hat{c}\}, (D_{t_2} = \{r_2\}, S_{t_2} = \{\hat{a}, \hat{ab}, \hat{b}\})\}) = 4, \\
\xi & (\{(D_{t_1} = \{r_2, r_3\}, S_{t_1} = \{\hat{a}, \hat{ab}, \hat{bc}, \hat{c}\}, (D_{t_2} = \emptyset, S_{t_2} = \emptyset)\}) = 1.
\end{align*}
\]

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

3. Problem Formulation

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

\[
\begin{align*}
\text{max} & \quad \sum_{r \in R} \sum_{t \in T} (w_r \cdot y^t_r) - \sum_{t \in T} \sum_{o \in V} \sum_{d \in V} (l^t_{od} \cdot x^t_{od}), \\
\text{s.t.} & \quad x^t_{od}, y^t_r \in \{0, 1\}, \\
& \sum_{t \in T} y^t_r \leq 1, \\
& \forall t \in T, o, d \in V, r \in R, \\
& \forall r : r \in R, \\
& \forall t \in T, r \in R, \\
& \forall t \in T, r \in R, \\
& \forall t \in T, o \in V, \\
& \forall t \in T, o \in V, \\
& \forall t \in T, o \in V, \\
& \forall t \in T, o \in V, \\
& \forall t \in T, o \in V, \\
& \forall t \in T, o \in V, \\
& \forall t \in T, o \in V, \\
& \forall t \in T, o \in V, \\
& \forall t \in T, o \in V, \\
& \forall t \in T, o \in V, \\
& \forall t \in T, o \in V. \\
\end{align*}
\]

\[\begin{aligned}
& \sum_{d \in V} x^t_{od} - \sum_{d \in V} x^t_{do} = 0, \\
& \sum_{d \in V} x^t_{od} \leq 1, \\
& \forall t \in T, o \in V, \\
& \forall t \in T, o \in V, \\
& \forall t \in T, o \in V, \\
& \forall t \in T, o \in V, \\
& \forall t \in T, o \in V, \\
& \forall t \in T, o \in V, \\
& \forall t \in T, o \in V, \\
& \forall t \in T, o \in V, \\
& \forall t \in T, o \in V. \\
\end{aligned}\]

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 \(x^t_{od}\) and \(y^t_r\). Eq. (3) guarantees that, each request can be assigned to at most one truck.

**Theorem 1.** For PPDS, 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 Miller et al. (1960), restricts the integer variables \( u^t_v \), 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 \( u^t_v \). Furthermore, Eq. (8) associated with Eqs. (4)–(7) also enforces that depot 0 must be visited and departed by truck \( t \) exactly once if \( D_t \neq \emptyset \).

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 \( y^t_r = 1 \).

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

**Proof.** We introduce another integer variables \( h^t_v \), 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 \( h^t_v \) 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 \( \Gamma \), which equals the loaded amount at \( v \) (i.e., \( \sum_{r \in f^{-1}(v)} (q_r \cdot y^t_r) \)) minus the unloaded amount at \( v \) (i.e., \( \sum_{r \in g^{-1}(v)} (q_r \cdot y^t_r) \)). The main part of Eq. (10) guarantees that \( h^t_d - h^t_o \) is exactly equal to \( \Gamma \) when \( x^t_odd = 1 \), which is written as a big-\( M \) linear inequality, where we specify \( M \) as \( c_t + \sum_{r \in R} q_r \).

### 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 \( \Theta(mn^2k^{-2}) \).

**Proof.** The number of Boolean variables \( x^t_odd \) (resp. \( y^t_r \)) required in proposed formulation is \( m(1 + 2nk^{-1})^2 \) (resp. \( mn \)); while both the number of required integer variables \( u^t_d \) and \( h^t_d \) 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. Please refer to Appendix A for the specific description of the existing formulation used for the comparative experiments.
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)  | $m\binom{2nk-1}{2}$ |

4.1. 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. 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 $n = \text{round}\left(\frac{k(|V|-1)}{2}\right)$.

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 $2n$.

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

**Input:** $G = (V, E)$, $k$

**Init.**: $n \leftarrow \text{round}\left(\frac{k(|V|-1)}{2}\right)$, `repeaTimeList` $\leftarrow [|V| - 1$ of 1]

1. **while** $\sum_i \text{repeaTimeList}[i] < 2n$ **do**
2. 
3. 
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.

---

2[http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/](http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/)
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: $G = (V, E)$, $n = \text{round}\left(\frac{k(V-1)}{2}\right)$, repeaTimeList

Init. : shuffList $\leftarrow \emptyset$, pairList $\leftarrow \emptyset$, reshuffle $\leftarrow \text{true}$

1 for $i \leftarrow 0$ to $|V| - 2$ do
2     shuffList.EXTEND([repeaTimeList[i] of i])
3 while reshuffle do
4     RANDOMSHUFFLE(shuffList)
5     pairList $\leftarrow \emptyset$
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.APPEND([shuffList[2i], shuffList[1 + 2i]])
11     if $i = n - 1$ then reshuffle $\leftarrow \text{false}$
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 $L$. We merge head and tail to obtain a sorted list of node pairs sortedPairs at the end of Algorithm 3. 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, $l_{od}^t$, i.e., the
Algorithm 3: Sort the node pairs by the sum of the repeat times of each node in the node pair

**Input:** repeaTimeList, pairList

**Init.:** \( \mathcal{L} \leftarrow \text{repeaTimeList}, \text{head} \leftarrow [], \text{tail} \leftarrow [], \text{max} \leftarrow 0, \text{maxIndex} \leftarrow -1, \text{sortedPairs} \leftarrow [] \)

1. while \(|\text{pairList}| > 0\) do
2. 
3. \( i \leftarrow 0 \)
4. while \( i < |\text{pairList}| \) do
5. 
6. if \( \mathcal{L}[\text{pairList}[i][0]] = 1 \) and \( \mathcal{L}[\text{pairList}[i][1]] = 1 \) then
7. 
8. \( \mathcal{L}[\text{pairList}[i][0]] \leftarrow \mathcal{L}[\text{pairList}[i][0]] - 1 \)
9. 
10. \( \mathcal{L}[\text{pairList}[i][1]] \leftarrow \mathcal{L}[\text{pairList}[i][1]] - 1 \)
11. 
12. \( \text{head}.\text{INSERT}(0, \text{pairList}.\text{POP}(i)) \)
13. 
14. else if \( \mathcal{L}[\text{pairList}[i][0]] = 1 \) or \( \mathcal{L}[\text{pairList}[i][1]] = 1 \) then
15. 
16. \( \mathcal{L}[\text{pairList}[i][0]] \leftarrow \mathcal{L}[\text{pairList}[i][0]] - 1 \)
17. 
18. \( \mathcal{L}[\text{pairList}[i][1]] \leftarrow \mathcal{L}[\text{pairList}[i][1]] - 1 \)
19. 
20. \( \text{head}.\text{APPEND}(\text{pairList}.\text{POP}(i)) \)
21. 
22. else \( i \leftarrow i + 1 \)
23. 
24. \( \text{max} \leftarrow 0 \)
25. 
26. \( \text{maxIndex} \leftarrow -1 \)
27. 
28. for \( j \leftarrow 0 \) to \(|\text{pairList}| - 1\) do
29. 
30. if \( \mathcal{L}[\text{pairList}[j][0]] + \mathcal{L}[\text{pairList}[j][1]] > \text{max} \) then
31. 
32. \( \text{max} \leftarrow \mathcal{L}[\text{pairList}[j][0]] + \mathcal{L}[\text{pairList}[j][1]] \)
33. 
34. \( \text{maxIndex} \leftarrow j \)
35. 
36. if \( \text{maxIndex} \neq -1 \) then
37. 
38. \( \mathcal{L}[\text{pairList}[\text{maxIndex}][0]] \leftarrow \mathcal{L}[\text{pairList}[\text{maxIndex}][0]] - 1 \)
39. 
40. \( \mathcal{L}[\text{pairList}[\text{maxIndex}][1]] \leftarrow \mathcal{L}[\text{pairList}[\text{maxIndex}][1]] - 1 \)
41. 
42. \( \text{tail}.\text{INSERT}(0, \text{pairList}.\text{POP}(\text{maxIndex})) \)
43. 
44. \( \text{sortedPairs} \leftarrow \text{head}.\text{EXTEND}(\text{tail}) \)
45. 
46. return sortedPairs

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.

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\};

\(^3\)Distance(od) is the Euclidean distance between the coordinates of node o and the coordinates of node d.
Algorithm 4: Randomly generate the list of requests

Input: $G = (V, E)$, $n = \text{round}(\frac{k|V|-1}{2})$, sortedPairs

Init.: $\text{avgDistance} \leftarrow \frac{1}{|V||V|-1} \sum_{o \in V} \sum_{d \in V \setminus o} \text{DISTANCE}(od)$, $\text{avgVolume} \leftarrow 5$,
requestList $\leftarrow []$

1 for $r \leftarrow 0$ to $n - 1$ do
2 \hspace{1cm} $q_r \leftarrow \text{round}(\text{RANDOMUNIFORM}(1, 2 \times \text{avgVolume} - 1))$
3 \hspace{1cm} $w_r \leftarrow \text{round}(2 \times \text{avgDistance} \times q_r \div \text{avgVolume})$
4 \hspace{1cm} $f(r) \leftarrow \text{sortedPairs}[r][0]$
5 \hspace{1cm} $g(r) \leftarrow \text{sortedPairs}[r][1]$
6 \hspace{1cm} requestList.$\text{Append}((w_r, q_r, f(r), g(r)))$
7 \hspace{1cm} return requestList

- 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 Settings

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 within 3,600 CPU seconds time limit. Source code for our experiments is available at https://github.com/ReprodSuplem/PPDSP.

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 ($\#\text{Var.}$), the number of generated constraints ($\#\text{Con.}$), 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, ulysse16 and ulysse22, respectively. Each cell recording the left and right values corresponds to a comparison item, where the left value refers to the performance of the item based on the existing formulation-based method, while the right value corresponds to the performance of the item based on 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
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 generated variables, the number of generated constraints, and the optimal values.

| 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 |

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 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.
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 generated variables, the number of generated constraints, and the optimal values.

| m  | item | $k = 1$ | $k = 1.5$ | $k = 2$ | $k = 2.5$ | $k = 3$ |
|----|------|---------|-----------|---------|-----------|---------|
|    |      | $(n = 8)$ | $(n = 11)$ | $(n = 15)$ | $(n = 19)$ | $(n = 23)$ |
| 2  | #Var. | 720 588 | 1248 594 | 2176 602 | 3360 610 | 4800 618 |
|    | #Con. | 94 94 | 96 94 | 109 112 | 157 187 | 168 230 |
|    | Opt.  | 720 94 | 1248 94 | 2176 112 | 3360 187 | 4800 230 |
| 4  | #Var. | 1440 1176 | 2496 1188 | 4352 1204 | 6720 1220 | 9600 1236 |
|    | #Con. | 94 94 | 96 94 | 109 112 | 157 187 | 168 230 |
|    | Opt.  | 1440 99 | 2496 104 | 4352 104 | 6720 104 | 9600 104 |
| 6  | #Var. | 4080 1764 | 3744 1782 | 6528 1806 | 10080 1830 | 14400 1854 |
|    | #Con. | 94 94 | 96 94 | 109 112 | 157 187 | 168 230 |
|    | Opt.  | 4080 99 | 3744 104 | 6528 104 | 10080 104 | 14400 104 |
| 8  | #Var. | 2880 2352 | 4992 2376 | 8704 2408 | 13440 2440 | 19200 2472 |
|    | #Con. | 94 94 | 96 94 | 109 112 | 157 187 | 168 230 |
|    | Opt.  | 2880 99 | 4992 104 | 8704 104 | 13440 104 | 19200 104 |
| 10 | #Var. | 3600 2940 | 6240 2970 | 10880 3010 | 16800 3050 | 24000 3090 |
|    | #Con. | 94 94 | 96 94 | 109 112 | 157 187 | 168 230 |
|    | Opt.  | 3600 99 | 6240 104 | 10880 104 | 16800 104 | 24000 104 |

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.

Acknowledgments

The authors are grateful for the support of the Progressive Logistic Science Corporate Sponsored Research Programs of the Research Center for Advanced Science and Technology, The University of Tokyo.
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 generated variables, the number of generated constraints, and the optimal values.

| m   | item | $k = 1$ $(n = 11)$ | $k = 1.5$ $(n = 16)$ | $k = 2$ $(n = 21)$ | $k = 2.5$ $(n = 26)$ | $k = 3$ $(n = 32)$ |
|-----|------|-------------------|----------------------|-------------------|-------------------|-------------------|
| 2   | #Var.| 1248              | 2448                 | 4048              | 6048              | 8976              |
|     | #Con.| 2311              | 4636                 | 7761              | 11686             | 17452             |
|     | Opt. | 116               | 161                  | 182               | 85                | 79                |
| 4   | #Var.| 2496              | 4896                 | 8096              | 12096             | 17952             |
|     | #Con.| 4611              | 9256                 | 15501             | 23346             | 34872             |
|     | Opt. | 134               | 141                  | 53                | 34                | 15                |
| 6   | #Var.| 3744              | 7344                 | 12144             | 18144             | 26928             |
|     | #Con.| 6911              | 13876                | 23241             | 35006             | 52292             |
|     | Opt. | 115               | 136                  | 0                 | 45                | 0                 |
| 8   | #Var.| 4992              | 9792                 | 16192             | 24192             | 35904             |
|     | #Con.| 9211              | 18496                | 30981             | 46666             | 69712             |
|     | Opt. | 124               | 102                  | 25                | 58                | 41                |
| 10  | #Var.| 6240              | 12240                | 20240             | 30240             | 44880             |
|     | #Con.| 11511             | 23116                | 38721             | 58326             | 87132             |
|     | Opt. | 124               | 31                   | 44                | 42                | 44                |

"m" represents the instance number, "#Var." is the number of generated variables, "#Con." is the number of generated constraints, and "Opt." is the optimal value.
References

Ahmadi-Javid, A., Amiri, E., & Meskar, M. (2018). A profit-maximization location-routing-pricing problem: A branch-and-price algorithm. *Eur. J. Oper. Res.*, 271(3), 866–881. https://doi.org/10.1016/j.ejor.2018.02.020

Al-Chami, Z., Flity, H. E., Manier, H., & Manier, M. (2018). A new metaheuristic to solve a selective pickup and delivery problem. 4th International Conference on Logistics Operations Management, GOL 2018, Le Havre, France, April 10-12, 2018, 1–5. https://doi.org/10.1109/GOL.2018.8378089

Asghari, M. & Mirzapour Al-e-hashem, S. M. J. (2020). 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. https://doi.org/https://doi.org/10.1016/j.tre.2019.11.009

CPLEX, I. I. (2020). *IBM ILOG CPLEX Optimizer*. https://www.ibm.com/analytics/cplex-optimizer.

Gansterer, M., Küçüktepe, M., & Hartl, R. F. (2017). The multi-vehicle profitable pickup and delivery problem. *OR Spectr.*, 39(1), 303–319. https://doi.org/10.1007/s00291-016-0454-y

Huang, D., Gu, Y., Wang, S., Liu, Z., & Zhang, W. (2020). A two-phase optimization model for the demand-responsive customized bus network design. *Transportation Research Part C: Emerging Technologies*, 111, 1–21. https://doi.org/https://doi.org/10.1016/j.trc.2019.12.004

Liu, C., Aleman, D. M., & Beck, J. C. (2018). Modelling and solving the senior transportation problem. *Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 15th International Conference, CPAIOR 2018, Delft, The Netherlands, June 26-29, 2018, Proceedings*, volume 10848 of Lecture Notes in Computer Science, 412–428. https://doi.org/10.1007/978-3-319-93031-2_30

Miller, C. E., Tucker, A. W., & Zemlin, R. A. (1960). Integer programming formulation of traveling salesman problems. *J. ACM*, 7(4), 326–329. https://doi.org/10.1145/321043.321046

Qiu, X., Feuerriegel, S., & Neumann, D. (2017). Making the most of fleets: A profit-maximizing multi-vehicle pickup and delivery selection problem. *Eur. J. Oper. Res.*, 259(1), 155–168. https://doi.org/10.1016/j.ejor.2016.10.010

Riedler, M. & Raidl, G. R. (2018). Solving a selective dial-a-ride problem with logic-based benders decomposition. *Comput. Oper. Res.*, 96, 30–54. https://doi.org/10.1016/j.cor.2018.03.008

Ting, C., Liao, X., Huang, Y., & Liaw, R. (2017). Multi-vehicle selective pickup and delivery using metaheuristic algorithms. *Inf. Sci.*, 406, 146–169. https://doi.org/10.1016/j.ins.2017.04.001
A. Existing Formulation

Here we provide a detailed description of the existing formulation proposed in Qiu et al. (2017) that were used to perform the comparative experiments in Section 4.

As we stated before, in order to compare the most essential differences between the request-based model and the location-based model, the constraints on time windows are not considered in this study. However, because in the existing formulation, the time windows constraints also act as the subtour elimination constraint. For this reason, we replace those constraints on time windows in the existing formulation with the same MTZ subtour elimination constraint as in our proposed formulation to ensure the fairness of the comparative experiments.

In the existing formulation, n requests are represented as a directed graph \( G = (V, E) \), whose node set \( V = \{0\} \cup P \cup D \cup \{2n + 1\} \), where \( P = \{1, 2, \ldots, n\} \) is the set of pickup nodes, \( D = \{n + 1, n + 2, \ldots, 2n\} \) is the set of dropoff nodes, and node 0 and 2n + 1 indicate two depots.\(^4\) A request \( r \) is formalized by using the combination \((r, r + n)\) to refer to the pickup and dropoff nodes, and each node \( v \in P \cup D \) corresponds to a change amount \( q_o \) of the truck’s load, where \( q_r \) and \( q_{r+n} \) respectively correspond to the loading and unloading of request \( r \), \( q_r > 0, q_{r+n} < 0 \), \( q_r = |q_{r+n}| \) and \( r \in \{1, 2, \ldots, n\} \). In addition, we particularly specify \( q_0 = q_{2n+1} = 0 \) for the two depot nodes. The definitions and notations of the other variables are consistent with that previously described in this paper. The existing formulation used for comparative experiments are as follows.

\[
\begin{align*}
\text{max} & \quad \sum_{t \in T} \left( \sum_{o \in P} w_o \sum_{d \in V} x_{od}^t \right) - \sum_{t \in T} \sum_{o \in V} \sum_{d \in V} (l_{od}^t \cdot x_{od}^t), \\
\text{s.t.} & \quad x_{od}^t \in \{0, 1\}, \quad \forall (t, o, d) : t \in T, o, d \in V, \\
& \quad \sum_{d \in V} x_{0d}^t = \sum_{o \in V} x_{o(2n+1)}^t = 1, \quad \forall t : t \in T, \\
& \quad \sum_{t \in T} \sum_{o \in V} x_{od}^t \leq 1, \quad \forall (t, d) : t \in T, d \in P, \\
& \quad \sum_{d \in V} x_{od}^t - \sum_{d \in V} x_{(o+n)d}^t = 0, \quad \forall (t, o) : t \in T, o \in P, \\
& \quad \sum_{d \in V} x_{vd}^t - \sum_{o \in V} x_{ov}^t = 0, \quad \forall (t, v) : t \in T, v \in P \cup D, \\
& \quad u_d^t - u_o^t \geq 1 - |V|(1 - x_{od}^t), \quad \forall (t, o, d) : t \in T, o, d \in V, o \neq d, \\
& \quad u_d^t - u_o^t > 0, \quad \forall (t, o, d) : t \in T, o \in P, d \in D, \\
& \quad h_d^t - h_o^t \geq q_d - c(t - x_{od}^t), \quad \forall (t, o, d) : t \in T, o, d \in V, \\
& \quad \max\{0, q_o\} \leq h_v^t \leq \min\{c, c + q_o\}, \quad \forall (t, v) : t \in T, v \in V, \\
& \quad 0 \leq u_v^t \leq |V| - 1, \quad \forall (t, v) : t \in T, v \in V \setminus \{0\},
\end{align*}
\]

\(^4\)In the experiments of this study, such two depots are in the identical location.
\begin{align*}
    x_{vv}^t &= 0, & \forall (t,v) : t \in T, v \in V, & (A.12) \\
    x_{0d}^t &= 0, & \forall (t,d) : t \in T, d \in D, & (A.13) \\
    x_{o(2n+1)}^t &= 0, & \forall (t,o) : t \in T, o \in P, & (A.14) \\
    x_{oo}^t &= 0, & \forall (t,o) : t \in T, o \in P \cup D, & (A.15) \\
    x_{(2n+1)d}^t &= 0, & \forall (t,d) : t \in T, d \in P \cup D. & (A.16)
\end{align*}

Eq. (A.3) restricts that all routes start and end at the depots, whereas Eq. (A.4) constrains each node \( d \) cannot be visited more than once. Eq. (A.5) guarantees that pickup and dropoff from a request can only be assigned to the same truck. Eq. (A.6) ensures that, for each non-depot node, the number of visits and the number of departures of a truck must be equal. Eqs. (A.7) and (A.8), associated with the domain of \( u_i^t \) shown in Eq. (A.11), are the MTZ subtour elimination constraint and the \textit{loading before unloading} constraint. Eqs. (A.9) and (A.10) correspond to the capacity constraint. Eqs. (A.12)–(A.16) are the constraints in Qiu et al. (2017) described as helping to improve the solving speed. Eq. (A.12) blocks cycles at the nodes, while Eq. Eq. (A.13) forbids the direct visit of dropoff nodes from the start depot. Eq. (A.14) (resp. Eq. (A.15)) prevents an immediate access to the end depot after visiting a pickup node (resp. visiting the start depot again). Eq. (A.16) avoids trucks from starting at the end depot.

It should be noted that we found three incorrectnesses in the existing formulations in Qiu et al. (2017), which correspond to Eqs. (A.3), (A.4) and (A.12) in the above formulation, and we have rectified each of them.