Optimizing a Multi-Commodity Home-Delivery and Pickup Service in Depopulated Rural Areas with Constraint Programming

Ryo Kuroiwa
Principles of Informatics Division, National Institute of Informatics, Tokyo, Japan
The Graduate University for Advanced Studies, SOKENDAI, Kanagawa, Japan

Tomoki Hasegawa
Future Technology Laboratories INC., Nagoya, Japan

Eiji Ueda
Future Technology Laboratories INC., Nagoya, Japan

Naoki Akiyama
Toyota Motor Corporation, Tokyo, Japan

Akira Yoshioka
Toyota Motor Corporation, Tokyo, Japan

Abstract

We study a routing problem for delivering and picking up multiple commodities with different priorities, motivated by the need to provide basic services to people in depopulated rural areas with a driver shortage. We define our problem as a generalization of the team orienteering problem with time windows, with additional constraints motivated by real-world applications. We develop constraint programming (CP) and mixed-integer programming (MIP) models to solve the formulated problem. In addition, we propose an incremental warm-starting strategy, which obtains an initial solution by solving a problem considering only a subset of commodities. In our experiment, CP outperforms MIP, and incremental warm-starting improves the performance of both approaches.

2012 ACM Subject Classification Applied computing → Transportation; Computing methodologies → Modeling methodologies; Mathematics of computing → Combinatorial optimization

Keywords and phrases Application, Operations Research & Mathematical Optimisation

Digital Object Identifier 10.4230/LIPIcs.CP.2026.36

Supplementary Material *Software*: `https://github.com/Kurorororo/cp2026-multicommodity-pd-service-code` [16], archived at `swh:1:dir:7bd2b51c53965a601023f76248acd43928ca9adb`

1 Introduction

In depopulated rural areas where people lack access to public transport, providing mobility is necessary for those unable to drive their own cars, e.g., elderly people, to access basic services [14]. Previous work has studied applications of vehicle routing problems to provide last-mile delivery services in rural areas [26, 7, 6]. With a similar motivation, we consider a home-delivery and pickup service in rural areas. In particular, considering the driver shortage in such areas, we define a multi-commodity delivery and pickup problem in which one driver simultaneously provides multiple services with different priorities, e.g., delivering food, delivering packages, and collecting packages. Our objective is to maximize the weighted number of served requests considering priorities, so our problem can be viewed as a variant of the orienteering problem [25]. In addition, motivated by our prospective real-world application, we incorporate multiple features: time-window constraints, heterogeneous vehicle fleets, multiple visits to distribution centers, a maximum time between pickup and delivery for perishable goods such as food, and drivers' shifts.

© Ryo Kuroiwa, Tomoki Hasegawa, Eiji Ueda, Naoki Akiyama, and Akira Yoshioka;
licensed under Creative Commons License CC-BY 4.0

To solve the formulated problem, we develop a constraint programming (CP) model and a mixed-integer programming (MIP) model. Furthermore, we propose an incremental warm-starting strategy for solving, where we solve a problem considering only a subset of commodities at each step and incrementally add a new commodity while using the current solution for warm-starting. Our experimental evaluation demonstrates that CP outperforms MIP, and incremental warm-starting improves the performance of both approaches.

2 Related Work

As far as we know, our problem setting has not been studied before, while it can be viewed as a hybridization and generalization of existing routing problems. Maximizing the (weighted) number of visited locations under time constraints is known as the orienteering problem [25]. As we have multiple vehicles and a time window at each location, our problem is particularly related to the team orienteering problem with time windows (TOPTW), to which CP was successfully applied [8]. While Baklagis et al. [1] studied a combination of TOPTW and pickup and delivery, they focused on pickup and delivery between customers with a single commodity. In contrast, we have multiple commodities, and each commodity is picked up from or delivered to a distribution center and is delivered to or picked up from each customer.

In the vehicle routing problems (VRP) literature, dealing with multiple commodities is actively studied [11]. As we allow each vehicle to visit a distribution center multiple times, our problem can be viewed as a VRP with multiple trips [5]. For a commodity such as food, we may specify a time limit for transportation after it is picked up, and problems with such constraints are called VRPs for perishable goods [24]. We have multiple vehicles with different capacities and available time windows. With this feature, our problem can be viewed as a heterogeneous vehicle routing problem [27]. We also need to assign a driver to each used vehicle while respecting drivers' shifts. Such problems were studied as VRPs with drivers' working hours [9]. Developing CP models for various VRP variants is an active area of research [15, 8, 22, 2, 18, 13, 17, 20], but the problems studied in such prior work do not involve multiple trips or perishable goods.

Our work can be placed in a research line that has investigated realistic VRPs by incorporating different types of constraints, some of which are mentioned in the previous paragraph. Such problems are called general VRPs [10] or rich VRPs [3]. In particular, our work shares a similar motivation with previous work on VRPs for rural and less populated areas. For example, existing work studied healthcare applications [4], cooperative pickup and delivery [26, 7], and the coordination of trucks and drones [6] in rural areas, while their problem settings are different from ours and do not use CP to solve the problems.

3 Problem Definition

In our service, we deliver commodities from distribution centers to customers or pick up commodities from customers and deliver them to distribution centers. We have multiple groups of commodities with different priorities, and each group is associated with a single distribution center. We consider service requests from customers consisting of the following parts:

- Service type, either delivery or pickup.
- Commodity group.
- Priority score to serve the request.
- Demand for delivery or pickup.

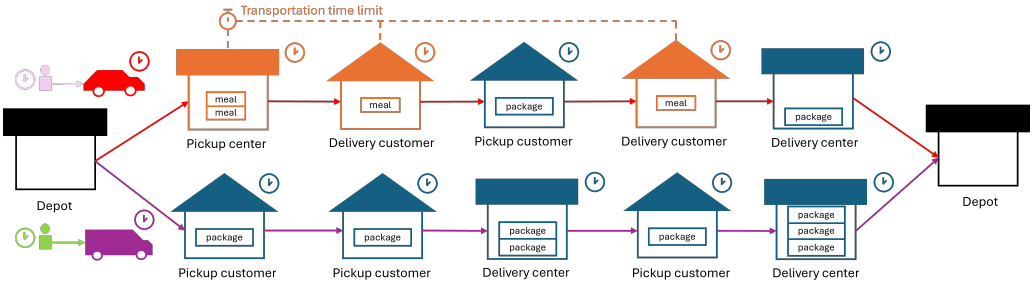

Figure 1 Illustrative example with two commodities, one for the delivery of perishable goods (meals) and another for pickup (packages). A solution with two vehicles and drivers is shown.

- Time window to arrive at the service location.
- Service time spent at the service location.
- Maximum transportation time that can be specified for the delivery of perishable goods.

We want to fulfill service requests as many as possible using a heterogeneous fleet of vehicles and drivers, where vehicles can have different depots and capacities, and drivers can have different shifts. In addition, drivers may not be able to use some of the vehicles. Figure 1 presents an illustrative example.

We summarize our notation in Table 1. We have a set of commodity groups $\mathcal{G}^{\text{delivery}}$ for delivery and another set $\mathcal{G}^{\text{pickup}}$ for pickup, and we use $\mathcal{G} = \mathcal{G}^{\text{delivery}} \cup \mathcal{G}^{\text{pickup}}$. For each group $g \in \mathcal{G}$, a set of requests R_g and a distribution center is associated, and we use $\mathcal{R}^{\text{delivery}} = \bigcup_{g \in \mathcal{G}^{\text{delivery}}} \mathcal{R}_g$, $\mathcal{R}^{\text{pickup}} = \bigcup_{g \in \mathcal{G}^{\text{pickup}}} \mathcal{R}_g$, and $\mathcal{R} = \mathcal{R}^{\text{delivery}} \cup \mathcal{R}^{\text{pickup}}$. We have a set of vehicles $\mathcal{V}$, and for each vehicle, we may construct a tour that starts and ends at its depot, sequentially serves requests by visiting the corresponding locations, and visits distribution centers. Each service request $r \in \mathcal{R}$ has the priority score p_r, and the objective is to construct a set of tours maximizing the sum of the priority scores of the served requests. When a vehicle serves a delivery request $r \in \mathcal{R}_g$, the units of the commodity in group g loaded in the vehicle is decreased by the demand q_r. When a vehicle visits the distribution center of group g, it can load arbitrary units of the commodity in group g. When a vehicle serves a pickup request $r \in \mathcal{R}_g$, its load is increased by the demand q_r, and the vehicle must visit the distribution center of g afterward, where the load for group g becomes 0. The total load of a vehicle $v \in \mathcal{V}$ must fall in $[0, Q_v]$ at any point, where Q_v is the capacity of v.

We use the term location to represent either a depot, a distribution center, or a service request. If a location j is visited immediately after location i, the travel time c_{ij} is incurred. Here, we use c_{vj}/c_{iv} to represent the travel time from/to the depot for vehicle v and c_{gj}/c_{ig} to represent the travel time from/to the distribution center of group $g \in \mathcal{G}$. We assume that the travel time satisfies the triangle inequality, i.e., $c_{ij} + c_{jk} \geq c_{ik}$ for any i, j, and k. A request $r \in \mathcal{R}$ has the time window $[a_r, b_r]$: if a vehicle serves a request $r \in \mathcal{R}$, it needs to arrive at the location by the deadline b_r and waits until the ready time a_r if arriving earlier. Similarly, the distribution center of group g has the time window $[a_g, b_g]$. Serving request r requires the service time s_r, and visiting the distribution center of group g requires service time s_g. We assume that the service time is positive.

For a delivery request r, the maximum transportation time T_r can be specified. In such a case, if the commodity for r is picked up at time t, it must be delivered by $t + T_r$. By $\mathcal{G}^{\text{perishable}} \subseteq \mathcal{G}^{\text{delivery}}$, we denote the set of such commodity groups.

Table 1 Summary of notation used in the problem definition.

Notation	Description
$\mathcal{G}^{\text{delivery}}$	the set of commodity groups for delivery
$\mathcal{G}^{\text{pickup}}$	the set of commodity groups for pickup
$\mathcal{G}^{\text{perishable}}$	the set of commodity groups for delivery with the maximum transportation time
$\mathcal{G}$	the set of all commodity groups ($\mathcal{G} = \mathcal{G}^{\text{delivery}} \cup \mathcal{G}^{\text{pickup}}$)
$[a_g, b_g]$	the time window for the distribution center of a commodity group $g \in \mathcal{G}$
s_g	the service time for the distribution center of a commodity group $g \in \mathcal{G}$
$\mathcal{R}_g$	the set of service requests for group $g \in \mathcal{G}$
$\mathcal{R}^{\text{delivery}}$	the set of all delivery requests ($\mathcal{R}^{\text{delivery}} = \bigcup_{g \in \mathcal{G}^{\text{delivery}}} \mathcal{R}_g$)
$\mathcal{R}^{\text{pickup}}$	the set of all pickup requests ($\mathcal{R}^{\text{pickup}} = \bigcup_{g \in \mathcal{G}^{\text{pickup}}} \mathcal{R}_g$)
$\mathcal{R}$	the set of all service requests ($\mathcal{R} = \mathcal{R}^{\text{delivery}} \cup \mathcal{R}^{\text{pickup}}$)
p_r	the priority score of service request $r \in \mathcal{R}$
q_r	the demand for service request $r \in \mathcal{R}$
$[a_r, b_r]$	the time window for service request $r \in \mathcal{R}$
s_r	the service time for service request $r \in \mathcal{R}$
T_r	the maximum transportation time for a delivery request $r \in \mathcal{R}$
c_{ij}	the travel time from location i to location j
$\mathcal{V}$	the set of vehicles
$\mathcal{V}_d$	the set of vehicles that can be assigned to driver $d \in \mathcal{D}$
Q_v	the capacity of vehicle $v \in \mathcal{V}$
$[a_v, b_v]$	the time window for vehicle $v \in \mathcal{V}$
$\mathcal{D}$	the set of drivers
$\mathcal{D}_v$	the set of drivers that can be assigned to vehicle $v \in \mathcal{V}$
$[a_d, b_d]$	the time window for driver $d \in \mathcal{D}$
T_d	the maximum working time for driver $d \in \mathcal{D}$

We have a set of drivers $\mathcal{D}$, where each driver $d \in \mathcal{D}$ has the time window $[a_d, b_d]$ and the maximum working time T_d. For each vehicle used for a tour, a driver d must be assigned, the tour's starting and ending times must fall within the time window $[a_d, b_d]$, and the tour duration must be less than or equal to T_d. A driver cannot be assigned to multiple vehicles. We use $\mathcal{D}_v \subseteq \mathcal{D}$ to denote the set of drivers that can be assigned to vehicle v, and $\mathcal{V}_d \subseteq \mathcal{V}$ to denote the set of vehicles that can be assigned to driver d.

4 Constraint Programming Model

We present our CP model. We first describe the decision variables used in the model. Then, we show the full model with the objective function and constraints.

4.1 Decision Variables

In our CP model, we use three types of decision variables:

- Boolean variable whose domain is $\{\bot, \top\}$.
- Integer variable whose domain is a finite subset of integers.
- Optional interval variable, consisting of three variables: an integer variable representing the length of the interval, an integer variable representing the start of the interval, and a Boolean variable representing the existence of the interval.

We consider vehicle tours as network flows in a graph where nodes are locations. We use a Boolean variable x_{ij} to represent that edge (i, j) is used in a tour, i.e., location j is visited immediately after another location i. In a tour, a vehicle may visit a distribution center multiple times. Therefore, we duplicate the distribution center of group $g \in \mathcal{G}$ for each vehicle $v \in \mathcal{V}$, introducing a set of locations $\mathcal{O}_{vg} = \{o^1_{vg}, ..., o^{k_{vg}}_{vg}\}$, where k_{vg} is the maximum number of times the distribution center of group g can be visited by vehicle v in a tour. We compute k_{vg} in the following procedure, given that the minimum time for vehicle v to visit the distribution center for g, serve request r for g, and return to the distribution center is $s_g + c_{gr} + s_r + c_{rg}$.

1. $k_{vg} \leftarrow 1$ and $t \leftarrow \max\{a_v + c_{vg}, a_g\}$.
2. Select $r \in \mathcal{R}_g$ minimizing $s_g + c_{gr} + s_r + c_{rg}$ and not selected before.
3. $t \leftarrow t + s_g + c_{gr} + s_r + c_{rg}$.
4. If $t \leq \min\{b_v - c_{gv}, b_g\}$, then $k_{vg} \leftarrow k_{vg} + 1$ and go to 2.

We use $\mathcal{O}_v = \bigcup_{g\in\mathcal{G}} \mathcal{O}_{vg}$, $\mathcal{O}_g = \bigcup_{v\in\mathcal{V}} \mathcal{O}_{vg}$, and $\mathcal{O} = \bigcup_{v\in\mathcal{V},g\in\mathcal{G}} \mathcal{O}_{vg}$. For each $o \in \mathcal{O}_{vg}$, we use $a_o = a_g$, $b_o = b_g$, $s_o = s_g$, $c_{io} = c_{ig}$, and $c_{oj} = c_{gj}$. Given this extension in notation, we define the set of edges $\mathcal{A}$ as follows:

$$\mathcal{A}_v = \begin{array}{l} \{(v, j) \mid j \in \mathcal{R}^{\text{pickup}} \cup \{o^1_{vg} \mid g \in \mathcal{G}^{\text{delivery}}\}, a_v + c_{vj} \leq b_j\} \cup \\ \{(i, v) \mid i \in \mathcal{R}^{\text{delivery}} \cup \{o^1_{vg} \mid g \in \mathcal{G}^{\text{pickup}}\}, a_i + s_i + c_{iv} \leq b_v\} \end{array} \quad \forall v \in \mathcal{V}$$

$$\mathcal{A} = \begin{array}{l} \big(\{(i, j) \mid i \in \mathcal{R} \cup \mathcal{O}, j \in (\mathcal{R} \cup \mathcal{O}) \setminus \{i\}, a_i + s_i + c_{ij} \leq b_j\} \cup \bigcup_{v\in\mathcal{V}} \mathcal{A}_v\big) \setminus \\ \big(\{(r, o^1_{vg}) \mid g \in \mathcal{G}^{\text{delivery}}, r \in \mathcal{R}_g, v \in \mathcal{V}\} \cup \{(o^1_{vg}, r) \mid v \in \mathcal{V}, g \in \mathcal{G}^{\text{pickup}}, r \in \mathcal{R}_g\}\big). \end{array}$$

We do not include infeasible or useless edges considering time windows. In addition, serving a delivery request before visiting the corresponding distribution center is impossible, and visiting the distribution center before serving any associated pickup request does not have any benefit. Without loss of generality, for a delivery group g, we assume that only one copy of the distribution center o^1_{vg} can be selected as the first location, and we exclude each edge (r, o^1_{vg}) for $r \in \mathcal{R}_g$. We also exclude edges related to pickup requests in a similar way, assuming that only o^1_{vg} can be selected as the last location for a pickup group g.

To handle the time window constraints, we represent visiting location $i \in \mathcal{R} \cup \mathcal{O}$ using an optional interval variable δ_i. The location is visited if the interval is present. The start of the interval corresponds to the arrival time at the location, and the end of the interval corresponds to the departure time. We use a Boolean variable $\mathsf{Pres}(\delta_i) \in \{\bot, \top\}$ such that $\mathsf{Pres}(\delta_i) = \top$ indicates the presence. The length of the interval is fixed to s_i, and the start $\mathsf{Start}(\delta_i)$ falls in $[a_i, b_i]$, denoted by $\delta_i : \mathsf{optIntervalVar}(s_i, [a_i, b_i])$. We also use $\mathsf{End}(\delta_i) = \mathsf{Start}(\delta_i) + s_i$. In addition, we define an optional interval variable $\lambda_v : \mathsf{optIntervalVar}([0, b_v - a_v], [a_v, b_v])$, representing the tour for vehicle $v \in \mathcal{V}$. The vehicle is used if the interval is present; it leaves the depot at the start of the interval and returns to the depot at the end of the interval. The length $\mathsf{Length}(\lambda_v)$ of the interval is an integer variable in $[0, b_v - a_v]$, and the start $\mathsf{Start}(\lambda_v)$ falls in $[a_v, b_v]$. We use $\mathsf{End}(\lambda_v) = \mathsf{Start}(\lambda_v) + \mathsf{Length}(\lambda_v)$. To link δ_r for request $r \in \mathcal{R}$ and λ_v for vehicle $v \in \mathcal{V}$, we also use a Boolean variable y_{rv} representing that r is served by v.

To handle other constraints, we use Boolean variable z_{dv} indicating that driver $d \in \mathcal{D}$ uses vehicle v, integer variable l_{gi} representing the load of the vehicle that arrives at location i, and an integer variable τ_{gi} representing the transportation time of the commodity of group g loaded in the vehicle that arrives at location i. We summarize the decision variables in Table 3. We present the CP model in Objective (1)–Formula (45). We partition the model into different parts and explain them one by one.

Table 2 Summary of additional notation used in the CP model.

Notation	Description
$\mathcal{O}_{vg}$	the set of distribution centers for vehicle $v \in \mathcal{V}$ and group $g \in \mathcal{G}$
$\mathcal{O}_v$	the set of distribution centers for vehicle $v \in \mathcal{V}$ ($\mathcal{O}_v = \bigcup_{g \in \mathcal{G}} O_{vg}$)
$\mathcal{O}_g$	the set of distribution centers for group $g \in \mathcal{G}$ ($\mathcal{O}_g = \bigcup_{v \in \mathcal{V}} O_{vg}$)
$\mathcal{O}$	the set of distribution centers ($O = \bigcup_{v \in \mathcal{V}, g \in \mathcal{G}} O_{vg}$)
k_{vg}	the number of times vehicle $v \in \mathcal{V}$ can visit the distribution center of group $g \in \mathcal{G}$
$\mathcal{A}$	the set of directed edges connecting two locations

Table 3 Summary of the decision variables in the CP model.

Variable	Description
x_{ij}	Boolean variable indicating location j is visited directly after i for $(i,j) \in \mathcal{A}$
δ_i	optional interval variable representing visiting location $i \in \mathcal{R} \cup \mathcal{O}$
λ_v	optional interval variable representing the tour of vehicle $v \in \mathcal{V}$
y_{rv}	Boolean variable indicating request $r \in \mathcal{R}$ is served by vehicle $v \in \mathcal{V}$
z_{dv}	Boolean variable indicating driver $d \in \mathcal{D}$ uses vehicle $v \in \mathcal{V}$
l_{gi}	load of group $g \in \mathcal{G}$ when arriving at location $i \in \mathcal{R} \cup \mathcal{O}$
τ_{gi}	transportation time of group $g \in \mathcal{G}^{\text{perishable}}$ when arriving at location $i \in \mathcal{R} \cup \mathcal{O}$

4.2 Objective and Tour Constraints

Because we use an optional interval variable δ_r to represent the visit to the location of request r, Objective (1) maximizes the total scores of the served requests, where a request $r \in \mathcal{R}$ is served if $\mathsf{Pres}(\delta_r) = \top$. We use the indicator function $\mathbb{1}$ such that $\mathbb{1}(\top) = 1$ and $\mathbb{1}(\bot) = 0$.

We use a Boolean variables x_{ij} for each $(i,j) \in \mathcal{A}$, representing that a vehicle traverses an edge (i,j). Each tour starts from the depot with a particular vehicle v using an edge (v,j) and must return to the depot with the same vehicle using an edge (i,v). Constraints (2) and (3) ensure that at most one tour starts and ends with each vehicle, where $\mathsf{AtMostOne}(\{x_{vj} \mid (v,j) \in \mathcal{A}\})$ is equivalent to $\sum_{(v,j)\in\mathcal{A}} \mathbb{1}(x_{vj}) \leq 1$. Constraints (4) and (5) ensure that at most one incoming edge and at most one outgoing edge are traversed for each location.

We use an optional interval variable λ_v to represent the tour duration for vehicle v. Constraint (6) ensures that if an edge (v,j) is used, vehicle v is used ($\mathsf{Pres}(\lambda_v) = \top$), and location j is visited ($\mathsf{Pres}(\delta_i) = \top$). Constraint (7) is analogous to Constraint (6) for the end of the tour. Constraint (8) ensures that if edge (i,j) is used, the locations i and j are visited.

Constraints (6) and (9) ensure that $\mathsf{Pres}(\lambda_v) = \top$ iff there exists an outgoing edge (v,j) such that $x_{vj} = \top$, where $\mathsf{AtLeastOne}(\{x_{vj} \mid (v,j) \in \mathcal{A}\})$ is equivalent to $\bigvee_{(v,j)\in\mathcal{A}} x_{vj} = \top$. Similarly, Constraints (7) and (10) ensure that $\mathsf{Pres}(\lambda_v) = \top$ if and only if there exists an incoming edge (i,v) such that $x_{iv} = \top$. As a result, there exists an outgoing edge (v,j) with $x_{vj} = \top$ if and only if there exists an incoming edge (i,v) with $x_{iv} = \top$, ensuring that a vehicle must depart from and return to the depot. Similarly, Constraints (8) and (11) ensure that $\mathsf{Pres}(\delta_j) = \top$ if and only if an incoming edge is used, Constraints (8) and (12) ensure that $\mathsf{Pres}(\delta_i) = \top$ if and only if an outgoing edge is used, resulting in the flow conservation.

Constraint (13) ensures that the arrival time at the first location i is greater than or equal to the start time of the tour plus the distance from the depot, and Constraint (14) does similar for the end of the tour. Constraint (15) ensures that the arrival time at j is

distant from the departure time at i at least by the travel time. Because the service time is positive, the arrival time at j is larger than that of i, and a tour cannot be a cycle that starts and ends at the same location in $\mathcal{R} \cup \mathcal{O}$.

$$\max \sum_{r \in \mathcal{R}} p_r \mathbb{1}(\mathsf{Pres}(\delta_r)). \tag{1}$$

$$\mathsf{AtMostOne}(\{x_{vj} \mid (v,j) \in \mathcal{A}\}) \qquad \forall v \in \mathcal{V} \tag{2}$$

$$\mathsf{AtMostOne}(\{x_{iv} \mid (i,v) \in \mathcal{A}\}) \qquad \forall v \in \mathcal{V} \tag{3}$$

$$\mathsf{AtMostOne}(\{x_{ij} \mid (i,j) \in \mathcal{A}\}) \qquad \forall j \in \mathcal{R} \cup \mathcal{O} \tag{4}$$

$$\mathsf{AtMostOne}(\{x_{ij} \mid (i,j) \in \mathcal{A}\}) \qquad \forall i \in \mathcal{R} \cup \mathcal{O} \tag{5}$$

$$x_{vj} \implies \mathsf{Pres}(\lambda_v) \wedge \mathsf{Pres}(\delta_j) \qquad \forall (v,j) \in \mathcal{A}, v \in \mathcal{V} \tag{6}$$

$$x_{iv} \implies \mathsf{Pres}(\delta_i) \wedge \mathsf{Pres}(\lambda_v) \qquad \forall (i,v) \in \mathcal{A}, v \in \mathcal{V} \tag{7}$$

$$x_{ij} \implies \mathsf{Pres}(\delta_i) \wedge \mathsf{Pres}(\delta_j) \qquad \forall (i,j) \in \mathcal{A}, i,j \notin \mathcal{V} \tag{8}$$

$$\mathsf{Pres}(\lambda_v) \implies \mathsf{AtLeastOne}(\{x_{vj} \mid (v,j) \in \mathcal{A}\}) \qquad \forall v \in \mathcal{V} \tag{9}$$

$$\mathsf{Pres}(\lambda_v) \implies \mathsf{AtLeastOne}(\{x_{iv} \mid (i,v) \in \mathcal{A}\}) \qquad \forall v \in \mathcal{V} \tag{10}$$

$$\mathsf{Pres}(\delta_j) \implies \mathsf{AtLeastOne}(\{x_{ij} \mid (i,j) \in \mathcal{A}\}) \qquad \forall j \in \mathcal{R} \cup \mathcal{O} \tag{11}$$

$$\mathsf{Pres}(\delta_i) \implies \mathsf{AtLeastOne}(\{x_{ij} \mid (i,j) \in \mathcal{A}\}) \qquad \forall i \in \mathcal{R} \cup \mathcal{O} \tag{12}$$

$$x_{vj} \implies \mathsf{Start}(\delta_j) \geq \mathsf{Start}(\lambda_v) + c_{vj} \qquad \forall (v,j) \in \mathcal{A}, v \in \mathcal{V} \tag{13}$$

$$x_{iv} \implies \mathsf{End}(\lambda_v) \geq \mathsf{End}(\delta_i) + c_{iv} \qquad \forall (i,v) \in \mathcal{A}, v \in \mathcal{V} \tag{14}$$

$$x_{ij} \implies \mathsf{Start}(\delta_j) \geq \mathsf{End}(\delta_i) + c_{ij} \qquad \forall (i,j) \in \mathcal{A}, i,j \notin \mathcal{V} \tag{15}$$

$$x_{ij} \in \{\bot, \top\} \qquad \forall (i,j) \in \mathcal{A} \tag{16}$$

$$\delta_i : \mathsf{optIntervalVar}(s_i, [a_i, b_i]) \qquad \forall i \in \mathcal{R} \cup \mathcal{O} \tag{17}$$

$$\lambda_v : \mathsf{optIntervalVar}([0, b_v - a_v], [a_v, b_v]) \qquad \forall v \in \mathcal{V}. \tag{18}$$

The constraints defined so far are insufficient to construct valid tours; a tour may start with one vehicle v with $x_{vj} = \top$ for some j while ending at some i with $x_{iv'} = \top$, where $v \neq v'$. We explicitly define a decision variable y_{rv} in Constraint (26) representing request r is served by vehicle v and ensure consistency with the tour. Constraint (19) ensures that at most one vehicle is assigned to a request, and Constraint (20) ensures that a vehicle must be assigned to a served request. Then, the remaining constraints ensure that if edge (i,j) is used, they must be served by the same vehicle. Constraint (21) ensures that $y_{rv} = \top$ if request r is served at the beginning of the tour, and Constraint (22) is for the end.

$$\mathsf{AtMostOne}(\{y_{rv} \mid v \in \mathcal{V}\}) \qquad \forall r \in \mathcal{R} \tag{19}$$

$$\mathsf{Pres}(\delta_r) \implies \mathsf{AtLeastOne}(\{y_{rv} \mid v \in \mathcal{V}\}) \qquad \forall r \in \mathcal{R} \tag{20}$$

$$x_{vr} \implies y_{rv} \qquad \forall (v,r) \in \mathcal{A}, v \in \mathcal{V}, r \in \mathcal{R}^{\text{pickup}} \tag{21}$$

$$x_{rv} \implies y_{rv} \qquad \forall (r,v) \in \mathcal{A}, r \in \mathcal{R}^{\text{delivery}}, v \in \mathcal{V} \tag{22}$$

$$x_{ij} \implies y_{iv} = y_{jv} \qquad \forall v \in \mathcal{V}, \forall (i,j) \in \mathcal{A}, i,j \in \mathcal{R} \tag{23}$$

$$x_{ro} \implies y_{rv} \qquad \forall v \in \mathcal{V}, \forall (r,o) \in \mathcal{A}, r \in \mathcal{R}, o \in \mathcal{O}_v \tag{24}$$

$$x_{or} \implies y_{rv} \qquad \forall v \in \mathcal{V}, \forall (o,r) \in \mathcal{A}, o \in \mathcal{O}_v, r \in \mathcal{R} \tag{25}$$

$$y_{rv} \in \{\bot, \top\} \qquad \forall r \in \mathcal{R}, \forall v \in \mathcal{V}. \tag{26}$$

These constraints are defined only for $\mathcal{R}^{\text{pickup}}$ or $\mathcal{R}^{\text{delivery}}$ as an edge (v,o) or (o,v) for a vehicle $v \in \mathcal{V}$ and a distribution center $o \in \mathcal{O}$ is defined only if $o \in \mathcal{O}_v$. Constraint (23) is for edges between request locations, and Constraints (24) and (25) are for edges between request locations and distribution centers.

Alternative Model with the Circuit Global Constraint

As an alternative CP model, we implement Constraints (2)–(12) using the circuit global constraint [19], which defines a Hamiltonian circuit in a graph. We use Google OR-Tools CP-SAT [21] as a CP solver, where the circuit constraint takes a set of triples as input. Each triple (i, j, x_{ij}) represents $x_{ij} = \top$ if and only if edge (i, j) in the graph is used in a Hamilton circuit. To represent an optional visit, we can also add a self-loop edge (i, i, x_{ii}), representing $x_{ii} = \top$ if and only if i is not included in the circuit. We represent a solution using a single Hamilton circuit; we order vehicles and denote the next vehicle of v in this order by $\mathsf{next}(v)$, introduce nodes α_v and ω_v, representing the start and end depots for each vehicle $v \in \mathcal{V}$, and consider edge $(\omega_v, \alpha_{\mathrm{next}(v)})$. If v is the last vehicle, then $\mathsf{next}(v)$ is the first vehicle. If vehicle v is not used, edge (α_v, ω_v) is used. Overall, in the alternative model, we have the following instead of Constraints (2)–(12):

$$\mathsf{Circuit}\left(\begin{array}{l}\{(i,j,x_{ij}) \mid (i,j) \in \mathcal{A}, i, j \notin \mathcal{V}\} \cup \{(i,i,\neg\mathsf{Pres}(\delta_i)) \mid i \in \mathcal{R} \cup \mathcal{O}\} \\ \cup\{(\alpha_v, j) \mid (v,j) \in \mathcal{A}, v \in \mathcal{V}\} \cup \{(i,\omega_v) \mid (i,v) \in \mathcal{A}, v \in \mathcal{V}\} \\ \cup\{(\omega_v, \alpha_{\mathrm{next}(v)}, \top) \mid v \in \mathcal{V}\} \cup \{(\alpha_v, \omega_v, \neg\mathsf{Pres}(\lambda_v)) \mid v \in \mathcal{V}\}\end{array}\right)$$

4.3 Driver Constraints

We use a Boolean variable z_{dv} in Constraint (33) indicating that driver d uses vehicle v. Constraint (27) ensures that each driver is assigned at most one vehicle, Constraint (28) ensures that each vehicle is assigned at most one driver, and Constraint (29) ensures that a vehicle v must be assigned a driver if it is used ($\mathsf{Pres}(\lambda_v) = \top$). Constraints (30)–(32) ensure that the start, the end, and the duration of a tour must respect the assigned driver's shift.

$$\mathsf{AtMostOne}(\{z_{dv} \mid v \in \mathcal{V}_d\}) \qquad \forall d \in \mathcal{D} \tag{27}$$

$$\mathsf{AtMostOne}(\{z_{dv} \mid d \in \mathcal{D}_v\}) \qquad \forall v \in \mathcal{V} \tag{28}$$

$$\mathsf{Pres}(\lambda_v) \implies \mathsf{AtLeastOne}(\{z_{vd} \mid d \in \mathcal{D}_v\}) \qquad \forall v \in \mathcal{V} \tag{29}$$

$$z_{dv} \implies \mathsf{Start}(\lambda_v) \geq a_d \qquad \forall d \in \mathcal{D}, \forall v \in \mathcal{V}_d \tag{30}$$

$$z_{dv} \implies \mathsf{End}(\lambda_v) \leq b_d \qquad \forall d \in \mathcal{D}, \forall v \in \mathcal{V}_d \tag{31}$$

$$z_{dv} \implies \mathsf{Length}(\lambda_v) \leq T_d \qquad \forall d \in \mathcal{D}, \forall v \in \mathcal{V}_d \tag{32}$$

$$z_{dv} \in \{\bot, \top\} \qquad \forall d \in \mathcal{D}, \forall v \in \mathcal{V}_d. \tag{33}$$

4.4 Capacity Constraints

We keep track of the load for commodity group g of a vehicle that arrives at location i by l_{gi}, defined in Constraints (41)–(44). For a service request $r \in \mathcal{R}$, the upper bound of l_{gi} is $\max_{v \in \mathcal{V}} Q_v$ because the capacity of the vehicle that serves r is unknown. In addition, l_{gi} must be at least d_r for a delivery request $r \in \mathcal{R}_g$, and l_{gi} must be at most $\max_{v \in \mathcal{V}} Q_v - d_r$ for a pickup request $r \in \mathcal{R}_g$. For a distribution center o for vehicle v, l_{go} is at most Q_v. Constraints (34) and (35) ensure the capacity constraint when each request location or distribution center is visited. The remaining constraints ensure the change of the load when an edge is traversed. Constraint (36) ensures that at the beginning of a tour, the load for each commodity group with delivery requests is empty, unless the first location visited after the depot is the distribution center for that group; we explain this exceptional case below. Constraint (37) ensures that at the end of a tour, the load for each commodity group with pickup requests is empty, unless the last location visited before the depot is the distribution

center for that group. Constraints (38) and (39) ensure the change of the load after serving a request. Constraint (40) ensures that the load for group g is not changed after visiting a location irrelevant to g.

$$y_{vr} \implies \sum_{g \in \mathcal{G}} l_{gr} \leq Q_v \qquad \forall v \in \mathcal{V}, \forall r \in \mathcal{R} \tag{34}$$

$$\sum_{g \in \mathcal{G}} l_{go} \leq Q_v \qquad \forall v \in \mathcal{V}, \forall o \in \mathcal{O}_v \tag{35}$$

$$x_{vj} \implies l_{gj} = 0 \qquad \forall g \in \mathcal{G}^{\text{delivery}}, \forall (v,j) \in \mathcal{A}, v \in \mathcal{V}, j \neq o^1_{vg} \tag{36}$$

$$x_{iv} \implies l_{gi} = 0 \qquad \forall g \in \mathcal{G}^{\text{pickup}}, \forall (i,v) \in \mathcal{A}, i \neq o^1_{vg}, v \in \mathcal{V} \tag{37}$$

$$x_{rj} \implies l_{gj} = l_{gr} - q_r \qquad \forall g \in \mathcal{G}^{\text{delivery}}, \forall (r,j) \in \mathcal{A}, r \in \mathcal{R}_g, j \notin \mathcal{V} \tag{38}$$

$$x_{rj} \implies l_{gj} = l_{gr} + q_r \qquad \forall g \in \mathcal{G}^{\text{pickup}}, \forall (r,j) \in \mathcal{A}, r \in \mathcal{R}_g, j \notin \mathcal{V} \tag{39}$$

$$x_{ij} \implies l_{gj} = l_{gi} \qquad \forall g \in \mathcal{G}, \forall (i,j) \in \mathcal{A}, i \notin \mathcal{R}_g \cup \mathcal{O}_g, j \notin \mathcal{V} \tag{40}$$

$$l_{gr} \in \left\{ q_r, ..., \max_{v \in \mathcal{V}} Q_v \right\} \qquad \forall g \in \mathcal{G}^{\text{delivery}}, \forall r \in \mathcal{R}_g \tag{41}$$

$$l_{gr} \in \left\{ 0, ..., \max_{v \in \mathcal{V}} Q_v - q_r \right\} \qquad \forall g \in \mathcal{G}^{\text{pickup}}, \forall r \in \mathcal{R}_g \tag{42}$$

$$l_{gi} \in \left\{ 0, ..., \max_{v \in \mathcal{V}} Q_v \right\} \qquad \forall g \in \mathcal{G}, \forall r \in \mathcal{R} \setminus \mathcal{R}_g \tag{43}$$

$$l_{go} \in \{0, ..., Q_v\} \qquad \forall v \in \mathcal{V}, \forall g \in \mathcal{G}, \forall o \in \mathcal{O}_{vg}. \tag{44}$$

We do not constrain l_{gj} at location j immediately after visiting the distribution center $o \in \mathcal{O}_g$. Therefore, l_{gj} can be arbitrarily increased or decreased under the domain and Constraints (34) and (35), allowing a vehicle to load or unload an arbitrary number of units of the commodity group g after visiting a distribution center $o \in \mathcal{O}_g$. A vehicle may decrease the load for group g even though $g \in \mathcal{G}^{\text{delivery}}$ or increase the load for g even though $g \in \mathcal{G}^{\text{pickup}}$ while there is no benefit in doing so. In such a case, we can obtain a valid solution by omitting the current visit to the distribution center and adjusting the load change in the previous or next visit to the same distribution center. Similarly, in Constraint (36), we do not constrain l_{gj} at the first location j if g is a commodity group for pickup or j is the distribution center o^1_{vg} for group g. If g is for pickup, starting with $l_{gj} > 0$ does not have any benefit, and we can obtain a valid solution by assuming that $l_{gj} = 0$ in postprocessing. If g is for delivery and $j = o^1_{vg}$, then the load for g at the next location can be an arbitrary value anyway, so the value of l_{gj} does not matter. We also do not constrain l_{gi} at the last location i if g is a group for delivery, so l_{gi} can be greater than 0. In such a case, a vehicle should have loaded more than necessary at the last visit to the distribution center of group g, and we can obtain a valid solution by decreasing the load change there.

4.5 Transportation Time Constraints

We use τ_{gi} to represent the transportation time for group $g \in \mathcal{G}^{\text{perishable}}$ for the vehicle that arrives at location i. At a delivery request location $r \in \mathcal{R}_g$, τ_{gi} is at most T_r as defined by Constraint (47). For the request locations and distribution centers of other groups, the upper bound is the maximum possible time ensured by Constraints (48) and (49). Constraint (45) ensures that τ_{go} is set to be 0 at the distribution center $o \in \mathcal{O}_g$ of the group g. Constraint (46) ensures that the transportation time is increased by the difference of the arrival times when a location j is visited from another location i.

$$\tau_{go} = 0 \qquad \forall g \in \mathcal{G}^{\text{perishable}}, \forall o \in \mathcal{O}_g \quad (45)$$

$$x_{ij} \implies \tau_{gj} = \tau_{gi} + \mathsf{Start}(\delta_j) - \mathsf{Start}(\delta_i) \qquad \forall g \in \mathcal{G}^{\text{perishable}}, \forall (i,j) \in \mathcal{A}, j \notin \mathcal{O}_g \quad (46)$$

$$\tau_{gr} \in \{0, ..., T_r\} \qquad \forall g \in \mathcal{G}^{\text{perishable}}, \forall r \in \mathcal{R}_g \quad (47)$$

$$\tau_{gr} \in \left\{0, ..., b_r - \min_{d \in \mathcal{D}} a_d\right\} \qquad \forall g \in \mathcal{G}^{\text{perishable}}, \forall r \in \mathcal{R} \setminus \mathcal{R}_g \quad (48)$$

$$\tau_{go} \in \{0, ..., \min\{b_g, b_v\} - a_v\} \qquad \forall g \in \mathcal{G}^{\text{perishable}}, \forall v \in \mathcal{V}, \forall o \in \mathcal{O}_v \setminus \mathcal{O}_{vg}. \quad (49)$$

We only maintain the transportation time since the last visit to the distribution center, assuming that all units of the commodity loaded at a distribution center are delivered before visiting the same distribution center again. When a solution violates this assumption, some units loaded at the first visit remain in the vehicle after the second visit. Loading those units at the second visit, rather than at the first, does not change the vehicle's total load after the second visit and does not violate the capacity constraints. Therefore, in postprocessing, we can obtain a valid solution by reducing the number of units loaded at the first visit and increasing the number at the second visit so that all units loaded at the first visit are delivered before the second visit.

5 Mixed-Integer Programming Model

Our mixed-integer programming (MIP) model is based on a similar idea to the CP model.

5.1 Decision Variables

We replace Boolean variables in the CP model with binary variables with the domain $\{0, 1\}$. Instead of the optional interval variable δ_i for each location i, we introduce a continuous variable t_i representing the arrival time at i. For the optional interval variable λ_v for each vehicle v, we introduce continuous variables t_v^{start} and t_v^{end} representing the start time and the end time of the tour, respectively. We summarize the decision variables in Table 4.

Table 4 Summary of the decision variables in the MIP model.

Variable	Description
x_{ij}	binary variable indicating location j is visited directly after i for $(i,j) \in \mathcal{A}$
t_i	the arrival time at location $i \in \mathcal{R} \cup \mathcal{O}$
t_v^{start}	the start time of vehicle $v \in \mathcal{V}$
t_v^{end}	the end time of vehicle $v \in \mathcal{V}$
y_{rv}	binary variable indicating request $r \in \mathcal{R}$ is served by vehicle $v \in \mathcal{V}$
z_{dv}	binary variable indicating driver $d \in \mathcal{D}$ uses vehicle $v \in \mathcal{V}$
l_{gi}	load of group $g \in \mathcal{G}$ when arriving at location $i \in \mathcal{R} \cup \mathcal{O}$
τ_{gi}	transportation time of group $g \in \mathcal{G}^{\text{perishable}}$ when arriving at location $i \in \mathcal{R} \cup \mathcal{O}$

5.2 Objective, Tour Constraints, and Driver Constraints

We consider a tour as a flow in a directed graph and the driver assignment to a vehicle as a flow from the driver to the vehicle. Then, a flow goes through customers from the vehicle and returns to the same vehicle. We use a binary variable z_{dv} representing that driver d uses vehicle v and a binary variable x_{ij} for edge (i,j). In Objective (50), we use $\sum_{(i,r) \in \mathcal{A}} x_{ir}$ to represent the visit to a request location r. Constraint (51) ensures that a

driver uses at most one vehicle. Constraint (52) ensures that a vehicle has at most one tour starting at some location j and ending at some location i if and only if a driver is assigned, and a vehicle is assigned at most one driver. Constraint (53) ensures that each location is visited at most once, and only one incoming edge and only one outgoing edge are used if visited. Constraints (54)–(56) ensure the arriving time at each location satisfies the travel time and service time constraints. Here, M is a large enough value: $M = b_v + c_{vj} - a_j$ in Constraint (54), $M = b_i + s_i + c_{iv} - a_v$ in Constraint (55), and $M = b_i + s_i + c_{ij} - a_j$ in Constraint (56). Constraints (57)–(59) ensure that the tour respects the driver's shift, where $M = b_v - b_d$ in Constraint (58) and $M = b_v - a_v - T_d$ in Constraint (59).

$$\max \sum_{r \in \mathcal{R}} p_r \sum_{(i,r) \in \mathcal{A}} x_{ir} \tag{50}$$

$$\sum_{v \in \mathcal{V}_d} z_{dv} \leq 1 \qquad \forall d \in \mathcal{D} \tag{51}$$

$$\sum_{d \in \mathcal{D}_v} z_{dv} = \sum_{(v,j) \in \mathcal{A}} x_{vj} = \sum_{(i,v) \in \mathcal{A}} x_{iv} \leq 1 \qquad \forall v \in \mathcal{V} \tag{52}$$

$$\sum_{(j,i) \in \mathcal{A}} x_{ji} = \sum_{(i,j) \in \mathcal{A}} x_{ij} \leq 1 \qquad \forall i \in \mathcal{R} \cup \mathcal{O} \tag{53}$$

$$t_j \geq t_v^{\text{start}} + c_{vj} - M(1 - x_{vj}) \qquad \forall (v,j) \in \mathcal{A}, v \in \mathcal{V} \tag{54}$$

$$t_v^{\text{end}} \geq t_i + s_i + c_{iv} - M(1 - x_{iv}) \qquad \forall (i,v) \in \mathcal{A}, v \in \mathcal{V} \tag{55}$$

$$t_j \geq t_i + s_i + c_{ij} - M(1 - x_{ij}) \qquad \forall (i,j) \in \mathcal{A}, i,j \notin \mathcal{V} \tag{56}$$

$$t_v^{\text{start}} \geq a_d z_{dv} \qquad \forall d \in \mathcal{D}, \forall v \in \mathcal{V}_d \tag{57}$$

$$t_v^{\text{end}} \leq b_d + M(1 - z_{dv}) \qquad \forall d \in \mathcal{D}, \forall v \in \mathcal{V}_d \tag{58}$$

$$t_v^{\text{end}} - t_v^{\text{start}} \leq T_d + M(1 - z_{dv}) \qquad \forall d \in \mathcal{D}, \forall v \in \mathcal{V}_d \tag{59}$$

$$a_v \leq t_v^{\text{start}}, t_v^{\text{end}} \leq b_v \qquad \forall v \in \mathcal{V} \tag{60}$$

$$a_i \leq t_i \leq b_i \qquad \forall i \in \mathcal{R} \cup \mathcal{O} \tag{61}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in \mathcal{A} \tag{62}$$

$$z_{dv} \in \{0,1\} \qquad \forall d \in \mathcal{D}, \forall v \in \mathcal{V}_d. \tag{63}$$

We use a binary variable y_{rv} that indicates whether request r is served by vehicle v. Constraint (64) ensures that a vehicle serves a request if and only if the request location is visited. With Constraint (53), at most one vehicle serves each request. Constraints (65)–(69) ensure that the locations i and j are visited by the same vehicle if an edge (i,j) is used.

$$\sum_{v \in \mathcal{V}} y_{rv} = \sum_{(i,r) \in \mathcal{A}} x_{ir} \qquad \forall r \in \mathcal{R} \tag{64}$$

$$y_{rv} \geq x_{rv} \qquad \forall r \in \mathcal{R}, \forall v \in \mathcal{V} \tag{65}$$

$$y_{rv} \geq x_{vr} \qquad \forall v \in \mathcal{V}, \forall r \in \mathcal{R} \tag{66}$$

$$y_{jv} \geq y_{iv} + x_{ij} - 1 \qquad \forall v \in \mathcal{V}, \forall (i,j) \in \mathcal{A}, i,j \notin \mathcal{V} \tag{67}$$

$$y_{rv} \geq x_{ro} \qquad \forall v \in \mathcal{V}, \forall (r,o) \in \mathcal{A}, r \in \mathcal{R}, o \in \mathcal{O}_v \tag{68}$$

$$y_{rv} \geq x_{or} \qquad \forall v \in \mathcal{V}, \forall (r,o) \in \mathcal{A}, o \in \mathcal{O}_v, r \in \mathcal{R} \tag{69}$$

$$y_{rv} \in \{0,1\} \qquad \forall r \in \mathcal{R}, \forall v \in \mathcal{V}. \tag{70}$$

5.3 Capacity Constraints

A continuous variable l_{gi} represents the load of the vehicle that arrives at location i. Constraints (71) and (72) ensure the capacity constraint, where $M = \max_{v' \in \mathcal{V}} Q_{v'} - Q_v$. Constraint (73) ensures that at the beginning of a tour, the load of each commodity group for delivery requests is empty, unless the first location visited after the depot is the distribution center for that group. Constraint (74) ensures that at the end of a tour, the load of each commodity group for pickup requests is empty, unless the last location visited before the depot is the distribution center for that group. In Constraints (73) and (74), $M = \max_{v' \in \mathcal{V}} Q_{v'}$.

Constraints (75)–(78) describe the demand change by visiting each location, where $M = \bar{l}_{gj} - \underline{l}_{gr} + q_r$ in Constraint (75), $M = \bar{l}_{gr} + q_r - \underline{l}_{gj}$ in Constraint (76), $M = \bar{l}_{gj} - \underline{l}_{gi}$ in Constraint (77), $M = \bar{l}_{gi} - \underline{l}_{gj}$ in Constraint (78), and $\bar{l}_{gj}$ and $\underline{l}_{gj}$ are upper and lower bounds on l_{gj} defined in Constraints (79)–(82). As we use inequalities, the load for group $g \in \mathcal{G}^{\text{delivery}}$ may decrease arbitrarily at any location, and the load for group $g \in \mathcal{G}^{\text{pickup}}$ may increase arbitrarily at any location, while doing so does not have any benefit. If a solution contains such a decrease or increase, we can obtain a valid solution by ignoring it and adjusting the load change at the corresponding distribution centers.

$$\sum_{g \in \mathcal{G}} l_{gr} \leq Q_v + M(1 - y_{rv}) \qquad \forall v \in \mathcal{V}, \forall r \in \mathcal{R} \qquad (71)$$

$$\sum_{g \in \mathcal{G}} l_{go} \leq Q_v \qquad \forall v \in \mathcal{V}, \forall o \in \mathcal{O}_v \qquad (72)$$

$$l_{gj} \leq M(1 - x_{vj}) \qquad \forall g \in \mathcal{G}^{\text{delivery}}, \forall (v,j) \in \mathcal{A}, v \in \mathcal{V}, j \neq o^1_{vg} \qquad (73)$$

$$l_{gi} \leq M(1 - x_{iv}) \qquad \forall g \in \mathcal{G}^{\text{pickup}}, \forall (i,v) \in \mathcal{A}, i \neq o^1_{vg}, v \in \mathcal{V} \qquad (74)$$

$$l_{gj} \leq l_{gr} - q_r + M(1 - x_{rj}) \qquad \forall g \in \mathcal{G}^{\text{delivery}}, \forall (r,j) \in \mathcal{A}, r \in \mathcal{R}_g, j \notin \mathcal{V} \qquad (75)$$

$$l_{gj} \geq l_{gr} + q_r - M(1 - x_{rj}) \qquad \forall g \in \mathcal{G}^{\text{pickup}}, \forall (r,j) \in \mathcal{A}, r \in \mathcal{R}_g, j \notin \mathcal{V} \qquad (76)$$

$$l_{gj} \leq l_{gi} + M(1 - x_{ij}) \qquad \forall g \in \mathcal{G}^{\text{delivery}}, \forall (i,j) \in \mathcal{A}, i \notin \mathcal{R}_g \cup \mathcal{O}_g, j \notin \mathcal{V} \qquad (77)$$

$$l_{gj} \geq l_{gi} - M(1 - x_{ij}) \qquad \forall g \in \mathcal{G}^{\text{pickup}}, \forall (i,j) \in \mathcal{A}, i \notin \mathcal{R}_g \cup \mathcal{O}_g, j \notin \mathcal{V} \qquad (78)$$

$$q_r \leq l_{gr} \leq \max_{v \in \mathcal{V}} Q_v \qquad \forall g \in \mathcal{G}^{\text{delivery}}, \forall r \in \mathcal{R}_g \qquad (79)$$

$$0 \leq l_{gr} \leq \max_{v \in \mathcal{V}} Q_v - d_r \qquad \forall g \in \mathcal{G}^{\text{pickup}}, \forall r \in \mathcal{R}_g \qquad (80)$$

$$0 \leq l_{gi} \leq \max_{v \in \mathcal{V}} Q_v \qquad \forall g \in \mathcal{G}, \forall r \in \mathcal{R} \setminus \mathcal{R}_g \qquad (81)$$

$$0 \leq l_{go} \leq Q_v \qquad \forall v \in \mathcal{V}, \forall g \in \mathcal{G}, \forall o \in \mathcal{O}_{vg}. \qquad (82)$$

5.4 Transportation Time Constraints

We use a continuous variable τ_{gi} and Constraints (83)–(86) to ensure the transportation time constraints. In Constraint (84), $M = \overline{\tau}_{gi} + b_j - a_i$, where $\overline{\tau}_{gi}$ is the upper bound on τ_{gi} defined in Constraints (85)–(87).

$$\tau_{go} = 0 \qquad \forall g \in \mathcal{G}^{\text{perishable}}, \forall o \in \mathcal{O}_g \qquad (83)$$

$$\tau_{gj} \geq \tau_{gi} + t_j - t_i - M(1 - x_{ij}) \qquad \forall g \in \mathcal{G}^{\text{perishable}}, \forall (i,j) \in \mathcal{A}, j \notin \mathcal{O}_g \qquad (84)$$

$$0 \leq \tau_{gr} \leq T_r \qquad \forall g \in \mathcal{G}^{\text{perishable}}, \forall r \in \mathcal{R}_g \qquad (85)$$

$$0 \leq \tau_{gr} \leq b_r - \min_{d \in \mathcal{D}} a_d \qquad \forall g \in \mathcal{G}^{\text{perishable}}, \forall r \in \mathcal{R} \setminus \mathcal{R}_g \qquad (86)$$

$$0 \leq \tau_{go} \leq \min\{b_g, b_v\} - a_v \qquad \forall g \in \mathcal{G}^{\text{perishable}}, \forall v \in \mathcal{V}, \forall o \in \mathcal{O}_v \setminus \mathcal{O}_{vg}. \qquad (87)$$

6 Enhancements

We introduce three enhancements in solving the MIP and CP models: preprocessing, postprocessing, and an incremental warm-starting strategy.

6.1 Preprocessing

We tighten time windows of vehicles, drivers, and locations given a problem instance, considering the service and travel time (see Appendix A for details). Then, using the tightened time windows, we compute k_{vg}, the maximum number of times that vehicle v can visit the distribution center for commodity group g, and the set of edges $\mathcal{A}$, as described in Section 4.1.

6.2 Postprocessing

After obtaining a solution, we perform postprocessing on it. As mentioned in Sections 4.4 and 5.3, we adjust the load change to obtain a valid solution if necessary. In addition, we perform the following procedures to obtain a more natural solution for human while the objective value is not changed.

- Remove unnecessary visits to distribution centers. After visiting the distribution center of group g for delivery, if no delivery request for g is served before the next visit to the same distribution center, the first visit is unnecessary and thus removed. Similarly, if a distribution center of group g for pickup is visited twice, and no pickup request for g is served in between, the second visit is unnecessary and thus removed.
- Adjust the start time at each location as early as possible. If a vehicle leaves a location i at time t, the start time at the next location j is updated to $\max\{t + c_{ij}, a_j\}$ unless j is a distribution center for delivery requests with a maximum transportation time.
- Adjust the tour start as late as possible. If a vehicle visits the first location j after the depot at time t, then the tour start time is updated to $t - c_{vj}$.

6.3 Incremental Warm-Starting Strategy

While we have multiple commodities, a solution that ignores some of the commodities is still valid because serving a request is optional. Therefore, instead of solving the problem at once, we consider the following incremental warm-starting strategy:

1. Solve the problem considering only one commodity group and obtain a solution.
2. Repeatedly solve the problem with one more commodity group while giving the current solution as an initial solution to a solver.

Typically, CP solvers and MIP solvers can take value assignments to decision variables as hints. Given a solution, for each visited location, we can extract the arrival time, the previous and next locations, the vehicle visiting it, the load of the vehicle, and the transportation time of each commodity group $g \in G^{\text{perishable}}$. In our CP models, these values are used as hints for x_{ij}, δ_i, y_{rv}, z_{dv}, l_{gi}, and τ_{gi}. For each unvisited location, we use $\mathsf{Pres}(\delta_i) = \bot$, $x_{ij} = \bot$ $y_{rv} = \bot$, and assign the lower bounds to the other variables. Similarly, for each vehicle v used in the solution, we extract the start time, the end time, and the driver and use them as hints for λ_v and z_{dv}. For unused vehicles, we set $\mathsf{Pres}(\lambda_v) = \bot$ and $z_{dv} = \bot$ and assign the lower bounds to the others. We do a similar initialization for our MIP model.

7 Experimental Evaluation

We evaluate the models with and without incremental warm-starting on a computer with two Intel(R) Xeon(R) Silver 4214R CPUs (24 physical CPU cores in total) and 128 GB memory. We use Gurobi 13.0.0 [12] for MIP and Google OR-Tools CP-SAT 9.14.6206 for CP with their Python interfaces. For each instance, we run a solver using 24 threads and a 15-minute time limit. The code and benchmark instances are attached as supplementary material.

7.1 Benchmark Instances

While we plan to apply our method to real-world problems, real data is confidential. Therefore, we generate synthetic problem instances based on the prospective real-world application. We provide a detailed explanation in Appendix B and show an overview in this section.

Our motivation is to use optimization for remote rural areas. We assume that there exists a central area in which the single depot for all vehicles and distribution centers are located. Then, we place rural areas, each of which has multiple locations. For each location, we randomly assign a request for each commodity group. We highlight some of the instance parameters (full details are presented in Appendix B).

- There are three commodity groups, where groups 1 and 2 are for delivery, and group 3 is for pickup. Group 1 is for delivering meal, group 2 is for delivering packages, and group 3 is for collecting packages. Group 1 is prioritized over group 2, and group 2 is over group 3. Each delivery request for group 1 has a maximum transportation time of 30 minutes plus the direct travel time from the delivery center to the request location.
- The road network is in the mesh topology, where remote areas are connected to several neighbors. We observe similar trends in the models' relative performance under the star topology, in which remote areas are aligned along a line, and traveling to an area on a different line requires passing through the central area (see Appendix C).
- The number of remote areas is in $\{4, 8, 12, 16\}$. The expected number of service requests in each area is 11, resulting in approximately $\{44, 88, 132, 176\}$ requests in total. In each area, the number of locations is uniformly sampled from $[5, 15]$. For each location, a request for group 1 is generated with a probability of 0.5, a request for group 2 is generated with 0.5, and a request for group 3 is generated with 0.1 (multiple requests with different groups can be assigned to the same location). Each service request in group 1 has a demand of 2 or 4, and each service request in groups 2 and 3 has a demand of 5.
- The number of drivers is the number of areas times a factor in $\left\{\frac{3}{4}, \frac{3}{2}, 3\right\}$. The number of vehicles equals the number of drivers. We have a six-hour planning time horizon, divided into five-second intervals (4,200 intervals in total). One-third of the drivers can work for any consecutive four hours, another one-third for the first three hours, and the remaining one-third for the last three hours. All vehicles are available for all drivers throughout the entire horizon. One-third of the vehicles have a capacity of 70, and the others have 40.

For each number of areas, we randomly generate five instances and consider three variants with different numbers of drivers, resulting in $4 \times 5 \times 3 = 60$ instances in total.

We consider the priority hierarchy between different groups: serving a single request in group i is prioritized over serving all requests in group $i + 1$. All requests in the same group have the same priority score. In incremental warm-starting, we first solve an instance considering only group 1 using a 5-minute time limit. Then, we solve an instance with groups 1 and 2 with a 5-minute time limit and the original instance with all commodity groups with a 5-minute time limit. If the first or second iteration finishes before the time limit, we extend the last iteration's time limit by the remaining time.

Table 5 Comparison of the two CP models in the average ratio of served requests for each commodity group. "Warm" refers to a configuration using incremental warm-starting. For commodity group 1, we bold the highest value among the evaluated methods. For commodity group $i > 1$, we bold the highest value only if the method also achieves the highest values in each group $j < i$.

		Circuit			Circuit Warm			No Circuit			No Circuit Warm		
#areas	#drivers	1	2	3	1	2	3	1	2	3	1	2	3
4	3	0.82	0.90	0.50	**0.93**	0.78	0.34	0.89	0.84	0.46	**0.93**	**0.85**	0.43
8	6	0.57	0.75	0.29	**0.75**	0.68	0.10	0.58	0.80	0.50	0.73	0.69	0.03
12	9	0.50	0.59	0.18	**0.69**	0.44	0.00	0.56	0.57	0.22	0.67	0.50	0.07
16	12	0.40	0.31	0.16	0.57	0.19	0.02	0.41	0.38	0.12	**0.62**	0.28	0.01
4	6	0.99	1.00	0.86	**1.00**	**1.00**	**0.96**	**1.00**	**1.00**	0.94	**1.00**	**1.00**	0.94
8	12	0.91	0.87	0.47	0.99	0.85	0.28	0.91	0.94	0.60	**1.00**	**0.94**	0.41
12	18	0.70	0.62	0.25	**0.98**	0.50	0.07	0.81	0.69	0.25	**0.98**	0.60	0.17
16	24	0.48	0.36	0.09	0.84	0.17	0.05	0.64	0.38	0.06	**0.88**	0.29	0.03
4	12	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
8	24	0.97	0.95	0.75	**1.00**	**0.99**	0.65	**1.00**	0.98	0.89	**1.00**	0.98	0.88
12	36	0.81	0.74	0.27	**1.00**	0.67	0.05	0.94	0.79	0.42	**1.00**	0.71	0.21
16	48	0.49	0.31	0.11	0.97	0.14	0.02	0.61	0.27	0.06	**1.00**	0.26	0.03

7.2 Experimental Results

We evaluate the ratio of served requests for each commodity group, which we are interested in for our prospective application. This metric is proportional to the objective value for each commodity group, and having a higher value in a higher priority group (e.g., group 1) is better even if having lower values in lower priority groups (e.g., groups 2 and 3). We take the average over the five instances with the same parameters.

Table 5 compares the two CP models: one with Constraints (2)–(12) (No Circuit) and another with the circuit global constraint instead (Circuit). Without warm-starting, No Circuit is better than Circuit, serving more requests in higher priority groups. Incremental warm-starting almost always improves the performance, except for No Circuit with 8 areas and 24 drivers. Comparing the two models with warm-starting, Circuit outperforms No Circuit in four out of the twelve settings: 8 areas with 6 drivers, 12 areas with 9 drivers, 4 areas with 6 drivers, and 8 areas with 24 drivers. With 16 areas, Circuit is consistently outperformed by No Circuit, possibly due to a longer time for constraint propagation.

Table 6 compares No Circuit and the MIP model. Incremental warm-starting is always better for MIP. With or without warm-starting, No Circuit outperforms MIP, almost always serving the same number of, or more, requests across all commodity groups on average.

Table 7 shows the number of optimally solved instances. In these instances, all requests are served. Circuit without warm-starting solves one instance with 8 areas that is not solved by Circuit with warm-starting. Otherwise, iterative warm-starting solves the same or more instances. The CP models are better than MIP with or without warm-starting.

8 Conclusion

In this paper, we defined a variant of the team orienteering problem with time windows, extended with multiple commodities and realistic constraints motivated by a real-world application in a home-delivery and pickup service for depopulated rural areas. We developed constraint

Table 6 Comparison of MIP and CP (No Circuit) in the average ratio of served requests for each commodity group. "Warm" refers to a configuration using incremental warm-starting. For commodity group 1, we bold the highest value among the evaluated methods. For commodity group $i > 1$, we bold the highest value only if the method also achieves the highest values in each group $j < i$.

		MIP			MIP Warm			No Circuit			No Circuit Warm		
#areas	#drivers	1	2	3	1	2	3	1	2	3	1	2	3
4	3	0.69	0.79	0.22	0.89	0.70	0.14	0.89	0.84	0.46	**0.93**	**0.85**	0.43
8	6	0.36	0.35	0.29	0.63	0.44	0.04	0.58	0.80	0.50	**0.73**	0.69	0.03
12	9	0.11	0.07	0.03	0.58	0.06	0.00	0.56	0.57	0.22	**0.67**	0.50	0.07
16	12	0.03	0.04	0.01	0.41	0.02	0.00	0.41	0.38	0.12	**0.62**	0.28	0.01
4	6	0.87	0.92	0.75	**1.00**	0.99	0.75	**1.00**	**1.00**	**0.94**	**1.00**	**1.00**	**0.94**
8	12	0.46	0.38	0.14	0.94	0.31	0.09	0.91	0.94	0.60	**1.00**	**0.94**	0.41
12	18	0.16	0.09	0.10	0.88	0.07	0.00	0.81	0.69	0.25	**0.98**	0.60	0.17
16	24	0.04	0.03	0.00	0.60	0.00	0.00	0.64	0.38	0.06	**0.88**	0.29	0.03
4	12	0.97	0.99	0.92	**1.00**	**1.00**	0.90	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
8	24	0.59	0.44	0.24	**1.00**	0.63	0.15	**1.00**	**0.98**	**0.89**	**1.00**	**0.98**	0.88
12	36	0.22	0.15	0.05	**1.00**	0.02	0.00	0.94	0.79	0.42	**1.00**	0.71	0.21
16	48	0.00	0.00	0.00	0.77	0.00	0.00	0.61	0.27	0.06	**1.00**	0.26	0.03

Table 7 Number of optimally solved instances per configuration. "Warm" refers to a configuration using incremental warm-starting.

#areas	#drivers	MIP	MIP Warm	Circuit	Circuit Warm	No Circuit	No Circuit Warm
4	4	0	0	0	**1**	**1**	**1**
4	6	1	3	2	**4**	3	3
4	12	3	4	**5**	**5**	**5**	**5**
8	24	0	0	**1**	0	**1**	**1**

programming (CP) and mixed-integer programming (MIP) models and demonstrated that CP outperforms MIP in our experimental evaluation. In addition, our incremental warm-starting strategy improves the performance of both approaches. This strategy is potentially useful to solve optimization models for other routing problems involving multiple commodities. For further performance improvement, we are considering using large neighborhood search [23], where we fix a subset of a solution and solve the remaining part using our CP model.

While our current CP model maximizes the total weighted number of requests served, it does not explicitly minimize the total travel distance. Considering multi-objective CP models or minimizing the travel distance of each route in post-processing is worth investigating.

We are currently working to implement our approach in a real-world application. We will use our problem as part of a complex problem, where we also consider deploying doctors and nurses for home healthcare, transporting residents between rural and urban areas, and providing on-demand bus services. We want to use the idle time of drivers and vehicles in these services to serve delivery and pickup requests, and thus, we will solve the CP model extended with constraints defined by existing routes.

References

1 D. G. Baklagis, G. Dikas, and I. Minis. An exact algorithm for the team orienteering pickup and delivery problem with time windows. *RAIRO – Oper. Res.*, 50:503–517, 2015. doi:10.1051/ro/2015030.

2 Kyle E. C. Booth and J. Christopher Beck. A constraint programming approach to electric vehicle routing with time windows. In *CPAIOR*, pages 129–145, 2019. doi:10.1007/978-3-030-19212-9_9.

3 Jose Caceres-Cruz, Pol Arias, Daniel Guimarans, Daniel Riera, and Angel A. Juan. Rich vehicle routing problem: Survey. *ACM Comput. Surv.*, 47(2), 2014. doi:10.1145/2666003.

4 Cristian Castillo, Eduard J. Alvarez-Palau, Laura Calvet, Javier Panadero, Marta Viu-Roig, Anna Serena-Latre, and Angel A. Juan. Home healthcare in Spanish rural areas: Applying vehicle routing algorithms to health transport management. *Socio-Econ. Plan. Sci.*, 92:101828, 2024. doi:10.1016/j.seps.2024.101828.

5 Diego Cattaruzza, Nabil Absi, and Dominique Feillet. Vehicle routing problems with multiple trips. *4OR*, 14(3):223–259, 2016. doi:10.1007/s10288-016-0306-2.

6 Anxiang Chen, Yoshiyasu Takahashi, and Youichi Nonaka. Modeling and evaluation of coordinated delivery of truck and drones in depopulated areas of Japan. In *IEEE SOLI*, pages 1–6, 2024. doi:10.1109/SOLI63266.2024.10955985.

7 Gustavo de Abreu Rodrigues, Leonardo Junqueira, José Geraldo Vidal Vieira, Orivalde Soares da Silva Júnior, and Claudio Barbieri da Cunha. Rural last-mile distribution using a hybrid heuristic-multi-criteria decision-making framework. *Expert Syst. Appl.*, 289:128331, 2025. doi:10.1016/j.eswa.2025.128331.

8 Rıdvan Gedik, Emre Kirac, Ashlea Bennet Milburn, and Chase Rainwater. A constraint programming approach for the team orienteering problem with time windows. *Comput. Ind. Eng.*, 107:178–195, 2017. doi:10.1016/j.cie.2017.03.017.

9 Asvin Goel. Vehicle scheduling and routing with drivers' working hours. *Transp. Sci.*, 43(1):17–26, 2009. doi:10.1287/trsc.1070.0226.

10 Asvin Goel and Volker Gruhn. A general vehicle routing problem. *Eur. J. Oper. Res.*, 191(3):650–660, 2008. doi:10.1016/j.ejor.2006.12.065.

11 Wenjuan Gu, Claudia Archetti, Diego Cattaruzza, Maxime Ogier, Frédéric Semet, and M. Grazia Speranza. Vehicle routing problems with multiple commodities: A survey. *Eur. J. Oper. Res.*, 317(1):1–15, 2024. doi:10.1016/j.ejor.2023.11.032.

12 Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2026. URL: https://www.gurobi.com.

13 Minh Hoàng Hà, Tat Dat Nguyen, Thinh Nguyen Duy, Hoang Giang Pham, Thuy Do, and Louis-Martin Rousseau. A new constraint programming model and a linear programming-based adaptive large neighborhood search for the vehicle routing problem with synchronization constraints. *Comput. Oper. Res.*, 124:105085, 2020. doi:10.1016/j.cor.2020.105085.

14 International Transport Forum. Innovations for better rural mobility. Technical report, OEC, 2021. doi:10.1787/6dbf832a-en.

15 Philip Kilby and Paul Shaw. Chapter 23 - vehicle routing. In Francesca Rossi, Peter van Beek, and Toby Walsh, editors, *Handbook of Constraint Programming*, volume 2, pages 801–836. Elsevier, 2006. doi:10.1016/S1574-6526(06)80027-1.

16 Ryo Kuroiwa. Program and data. Software, swhId: swh:1:dir:7bd2b51c53965a601023f76248acd43928ca9adb (visited on 2026-06-26). URL: https://github.com/Kurorororo/cp2026-multicommodity-pd-service-code, doi:10.4230/artifacts.26900.

17 Mustafa Küçük and Seyda Topaloglu Yildiz. Constraint programming-based solution approaches for three-dimensional loading capacitated vehicle routing problems. *Comput.Ind. Eng.*, 171:108505, 2022. doi:10.1016/j.cie.2022.108505.

18 Edward Lam, Pascal Van Hentenryck, and Phil Kilby. Joint vehicle and crew routing and scheduling. *Transp. Sci.*, 54(2):488–511, 2020. doi:10.1287/trsc.2019.0907.

19 Jean-Louis Lauriere. A language and a program for stating and solving combinatorial problems. *Artif. Intell.*, 10(1):29–127, 1978. doi:10.1016/0004-3702(78)90029-2.

20 Florian Linß. A constraint programming model for the vehicle routing problem with multiple time windows. In *ICCL*, pages 307–321, 2023. doi:10.1007/978-3-031-43612-3_19.

21 Laurent Perron, Frédéric Didier, and Steven Gay. The CP-SAT-LP solver. In *CP*, volume 280, pages 3:1–3:2, 2023. doi:10.4230/LIPIcs.CP.2023.3.

22 Bochra Rabbouch, Foued Saâdaoui, and Rafaa Mraihi. Constraint programming based algorithm for solving large-scale vehicle routing problems. In *HAIS*, pages 526–539, 2019. doi:10.1007/978-3-030-29859-3_45.

23 Paul Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *CP*, pages 417–431, 1998. doi:10.1007/3-540-49481-2_30.

24 Dana Marsetiya Utama, Shanty Kusuma Dewi, Abdul Wahid, and Imam Santoso. The vehicle routing problem for perishable goods: A systematic review. *Cogent Eng.*, 7(1):1816148, 2020. doi:10.1080/23311916.2020.1816148.

25 Pieter Vansteenwegen and Aldy Gunawan. *Orienteering Problems: Models and Algorithms for Vehicle Routing Problems with Profits*. Springer, 2019. doi:10.1007/978-3-030-29746-6.

26 Fei Yang, Ying Dai, and Zu-Jun Ma. A cooperative rich vehicle routing problem in the last-mile logistics industry in rural areas. *Transp. Res. Part E: Logist. Transp. Rev.*, 141:102024, 2020. doi:10.1016/j.tre.2020.102024.

27 Çağrı Koç, Tolga Bektaş, Ola Jabali, and Gilbert Laporte. Thirty years of heterogeneous vehicle routing. *Eur. J. Oper. Res.*, 249(1):1–21, 2016. doi:10.1016/j.ejor.2015.07.020.

A Tightening Time Windows

Given an instance, we tighten the time windows. First, we tighten the time window $[a_v, b_v]$ for each vehicle $v \in \mathcal{V}$ and $[a_d, b_d]$ for each driver $d \in \mathcal{D}$.

$$[a_v, b_v] \leftarrow \left[\max\left\{a_v, \min_{d\in\mathcal{D}_v} a_d\right\}, \min\left\{b_v, \min_{d\in\mathcal{D}_v} b_d\right\}\right] \qquad \forall v \in \mathcal{V}$$

$$[a_d, b_d] \leftarrow \left[\max\left\{a_d, \min_{v\in\mathcal{V}_d} a_v\right\}, \min\left\{b_d, \min_{v\in\mathcal{V}_d} b_v\right\}\right] \qquad \forall d \in \mathcal{D}.$$

For the earliest arrival time at a distribution center for delivery, we consider the earliest arrival time of a location that can be visited next. We exclude distribution centers for pickup from the next locations because the request locations for such groups must be reachable earlier. In addition, we consider the earliest arrival time from the depot using any vehicle. For the latest arrival time, we consider the latest departure time to serve a request.

$$a_g \leftarrow \max\left\{a_g, \min_{i\in\mathcal{R}_g\cup\mathcal{R}^{\text{pickup}}\cup(\mathcal{O}^{\text{delivery}}\setminus\mathcal{O}_g)} a_i - s_g - c_{gi}, \min_{v\in\mathcal{V}} a_v + c_{vg}\right\} \qquad \forall g \in \mathcal{G}^{\text{delivery}}$$

$$b_g \leftarrow \min\left\{b_g, \max_{r\in\mathcal{R}_g} b_r - s_g - c_{gr}\right\} \qquad \forall g \in \mathcal{G}^{\text{delivery}}.$$

For each delivery request location, we consider the earliest arrival time from the distribution center and the latest arrival time at the depot.

$$a_r \leftarrow \max\{a_r, a_g + s_g + c_{gr}\} \qquad \forall g \in \mathcal{G}^{\text{delivery}}, \forall r \in \mathcal{R}_g$$

$$b_r \leftarrow \min\left\{b_r, \max_{v\in\mathcal{V}} b_v - s_r - c_{rv}\right\} \qquad \forall g \in \mathcal{G}^{\text{delivery}}, \forall r \in \mathcal{R}_g.$$

For the earliest arrival time at a distribution center for pickup, we consider the earliest departure time from a request location. For the latest arrival time, we consider the latest departure time at a previous location, excluding distribution centers for delivery, and the latest arrival time at the depot.

$$a_g \leftarrow \max\left\{a_g, \min_{r\in\mathcal{R}_g} a_r + s_r + c_{rg}\right\} \qquad \forall g \in \mathcal{G}^{\text{pickup}}$$

$$b_g \leftarrow \min\left\{b_g, \max_{i\in\mathcal{R}_g\cup\mathcal{R}^{\text{delivery}}\cup(O^{\text{pickup}}\setminus O_g)} b_i + s_i + c_{ig}, \max_{v\in\mathcal{V}} b_v - s_g - c_{gv}\right\} \quad \forall g \in \mathcal{G}^{\text{pickup}}.$$

For each pickup request, we consider the earliest departure time from the depot and the latest arrival time to the distribution center.

$$a_r \leftarrow \max\left\{a_r, \min_{v\in\mathcal{V}} a_v + c_{vr}\right\} \qquad \forall g \in \mathcal{G}^{\text{pickup}}, \forall r \in \mathcal{R}_g$$

$$b_r \leftarrow \min\left\{b_r, b_g - s_r - c_{rg}\right\} \qquad \forall g \in \mathcal{G}^{\text{pickup}}, \forall r \in \mathcal{R}_g.$$

B Benchmark Instances

We explain the procedure to generate the benchmark instances.

B.1 Areas

We have a single central area and multiple remote areas. Each area is a circle in a 2-dimensional Euclidean space, and we randomly generate request locations within the circle. The travel time between locations in the same area is the Euclidean distance rounded up to an integer. The travel time between locations i and j in different areas A and B is the sum of the travel time between i and the center of A, the travel time between A and B, and the travel time between the center of B and j. The travel time between different areas depends on the topology.

For the mesh topology, we generate remote areas as follows:

1. Place the central area at the origin $(0, 0)$ of a 2-dimensional Euclidean space.
2. Place a remote area at a randomly selected position so that its Euclidean distance from any area is between $[t_{\min}, t_{\max}]$, and its distance from the origin is at most t_{origin}. Repeat it until $n_{\max}$ areas are generated.
3. Construct a graph where each node is an area and is connected to its k-nearest neighbors with the Euclidean distance, and the edge weight is the Euclidean distance between the areas corresponding to the nodes rounded up to an integer.

The travel time between two areas is the shortest path cost between the corresponding nodes in the graph.

For the star topology, we generate remote areas as follows:

1. Maintain a list of road lines $(L_1, ..., L_n)$, where each road line L_i is a sequence of areas $(A_{i0}, A_{i1}, ..., A_{im})$, and A_{i0} is the central area. Originally, we have an empty list $(n = 0)$.
2. Create a new road line $L_{n+1} = (A_{n+1,0})$ with the probability of $\frac{1}{n'+1}$ where n' is the number of existing road lines with less than $m_{\max}$ remote areas. Otherwise, select an existing one L_i with less than $m_{\max}$ remote areas with the probability of $\frac{1}{n'}$.
3. Add a new remote area to the created or selected road line.
4. Repeat 2 and 3 until $n_{\max}$ remote areas are generated.

The travel time between areas is defined as follows:

- The travel time $c^{\text{area}}_{i,j,i,j+1}$ between two remote areas A_{ij} and $A_{i,j+1}$ on the same road line L_i is sampled from $[t_{\min}, t_{\max}]$, and $c^{\text{area}}_{i,j+1,i,j} = c^{\text{area}}_{i,j,i,j+1}$.
- The travel time c^{area}_{ijik} between areas A_{ij} and A_{ik} on the same road line L_i with $j < k$ is $\sum_{l=j}^{k-1} c^{\text{area}}_{i,l,i,l+1}$, and $c^{\text{area}}_{ikij} = c^{\text{area}}_{ijik}$
- The travel time $c^{\text{area}}_{iji'j'}$ between different areas on different load lines L_i and $L_{i'}$ is $c^{\text{area}}_{iji0} + c^{\text{area}}_{i'0i'j}$.

We discretize time into five-second intervals, and the radius of each area is 60 (five minutes). In the central area, we have the single depot for all vehicles and the distribution centers for all commodity groups. For each remote area, the number of request locations is uniformly sampled from $[5, 15]$. We use $n_{\max} \in \{4, 8, 12, 16\}$ and $[t_{\min}, t_{\max}] = [120, 240]$ (the travel time between areas is 10 to 20 minutes). For the star topology, we use $m_{\max} = 3$. For the mesh topology, we use $t_{\text{origin}} = 3t_{\max}$ and $k = 2$.

B.2 Requests

For each request location, we generate a request for each commodity group i with the probability of p_i. We use three commodity groups: group 1 for delivery with $p_1 = 0.5$, group 2 for delivery with $p_2 = 0.5$, and group 3 for pickup with $p_3 = 0.1$.

Group 1 is for delivering meal. We consider the time horizon of six hours from 9 am (9:00) to 3 pm (15:00). The distribution center is open for pickup from 10:00 to 12:00, and each request location is open for delivery from 11:00 to 13:00. The maximum transportation time is 30 minutes plus the travel time between the distribution center and the request location. The demand at each location is uniformly sampled from $\{2, 4\}$.

Group 2 is for delivering packages. The distribution center is open for pickup from 9:00 to 15:00, and each request is open for delivery for two hours, where the start time is uniformly sampled from $\{09:00, 09:30, \ldots, 12:30, 13:00\}$. The demand at each location is 5.

Group 3 is for collecting packages. The distribution center is open for delivery from 9:00 to 15:00, and each request is open for pickup for two hours, where the start time is uniformly sampled from $\{09:00, 09:30, \ldots, 12:30, 13:00\}$. The demand at each location is 5.

In all commodity groups, the distribution center requires a service time of five minutes, and each request requires a service time of two minutes. The priority score for each service request in group 3 is 1. The priority score for each service request in group 2 is $n_3 + 1$, where n_3 is the number of service requests in group 3. The priority score for each service request in group 1 is $n_2(n_3 + 1) + n_3 + 1$, where n_2 is the number of service requests in group 2.

B.3 Drivers and Vehicles

We consider the following three drivers and three vehicles as one unit.

- A driver available from 9:00 to 15:00 with a maximum working time of four hours.
- A driver available from 9:00 to 12:00.
- A driver available from 12:00 to 15:00.
- A car with a capacity of 70 available from 9:00 to 15:00.
- Two cars with a capacity of 40 available from 9:00 to 15:00.

Given the number of areas n, we consider a situation where $\frac{n}{4}$ units are available, $\frac{n}{2}$ units are available, and n units are available, resulting in the number of drivers in $\left\{\frac{3n}{4}, \frac{3n}{2}, 3n\right\}$. All cars are available for all drivers.

C Experimental Evaluation Under the Star Topology

We show experimental evaluation results using the instances with the star topology in Tables 8–10. The overall tendency is similar to that of the mesh topology.

Table 8 Comparison of the two CP models in the average ratio of served requests for each commodity group. "Warm" refers to a configuration using incremental warm-starting. For commodity group 1, we bold the highest value among the evaluated methods. For commodity group $i > 1$, we bold the highest value only if the method also achieves the highest values in each group $j < i$.

		Circuit			Circuit Warm			No Circuit			No Circuit Warm		
#areas	#drivers	1	2	3	1	2	3	1	2	3	1	2	3
4	3	0.73	0.86	0.45	**0.84**	0.74	0.33	0.70	0.85	0.69	0.83	0.73	0.38
8	6	0.59	0.66	0.22	**0.79**	0.58	0.05	0.66	0.68	0.22	0.76	0.61	0.00
12	9	0.48	0.53	0.18	0.70	0.49	0.03	0.56	0.60	0.20	**0.72**	0.47	0.05
16	12	0.41	0.38	0.13	0.66	0.26	0.11	0.53	0.38	0.05	**0.72**	0.23	0.00
4	6	0.97	1.00	0.89	**1.00**	0.98	0.80	0.98	0.97	0.92	**1.00**	**1.00**	0.92
8	12	0.82	0.86	0.34	**0.99**	0.83	0.17	0.92	0.89	0.54	**0.99**	**0.91**	0.36
12	18	0.74	0.60	0.25	0.97	0.48	0.10	0.83	0.63	0.20	**0.98**	**0.69**	0.11
16	24	0.52	0.37	0.13	0.95	0.26	0.01	0.69	0.36	0.07	**0.98**	0.33	0.04
4	12	**1.00**	**1.00**	0.96	**1.00**	**1.00**	0.95	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
8	24	0.97	0.94	0.60	**1.00**	0.96	0.40	**1.00**	**0.99**	**0.92**	**1.00**	**0.99**	0.69
12	36	0.76	0.64	0.38	**1.00**	0.65	0.17	0.96	0.66	0.25	**1.00**	**0.81**	0.20
16	48	0.39	0.31	0.19	**1.00**	0.23	0.01	0.58	0.26	0.07	**1.00**	**0.43**	0.11

Table 9 Comparison of MIP and CP (No Circuit) in the average ratio of served requests for each commodity group. "Warm" refers to a configuration using incremental warm-starting. For commodity group 1, we bold the highest value among the evaluated methods. For commodity group $i > 1$, we bold the highest value only if the method also achieves the highest values in each group $j < i$.

		MIP			MIP Warm			No Circuit			No Circuit Warm		
#areas	#drivers	1	2	3	1	2	3	1	2	3	1	2	3
4	3	0.55	0.68	0.54	0.77	0.67	0.23	0.70	0.85	0.69	**0.83**	0.73	0.38
8	6	0.33	0.33	0.14	0.68	0.39	0.00	0.66	0.68	0.22	**0.76**	0.61	0.00
12	9	0.08	0.11	0.02	0.61	0.05	0.05	0.56	0.60	0.20	**0.72**	0.47	0.05
16	12	0.03	0.01	0.03	0.51	0.01	0.00	0.53	0.38	0.05	**0.72**	0.23	0.00
4	6	0.81	0.90	0.71	**1.00**	0.93	0.53	0.98	0.97	0.92	**1.00**	**1.00**	**0.92**
8	12	0.27	0.24	0.12	0.98	0.17	0.00	0.92	0.89	0.54	**0.99**	**0.91**	0.36
12	18	0.17	0.11	0.19	0.91	0.03	0.00	0.83	0.63	0.20	**0.98**	**0.69**	0.11
16	24	0.01	0.00	0.00	0.71	0.00	0.00	0.69	0.36	0.07	**0.98**	0.33	0.04
4	12	0.96	0.95	0.71	**1.00**	**1.00**	0.77	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
8	24	0.47	0.30	0.27	**1.00**	0.41	0.09	**1.00**	**0.99**	**0.92**	**1.00**	**0.99**	0.69
12	36	0.12	0.10	0.12	**1.00**	0.06	0.00	0.96	0.66	0.25	**1.00**	**0.81**	0.20
16	48	0.00	0.00	0.00	0.88	0.00	0.00	0.58	0.26	0.07	**1.00**	**0.43**	**0.11**

Table 10 Number of optimally solved instances per configuration. "Warm" refers to a configuration using incremental warm-starting.

#areas	#drivers	MIP	MIP Warm	Circuit	Circuit Warm	No Circuit	No Circuit Warm
4	4	0	0	0	0	0	0
4	6	1	2	3	**4**	2	**4**
4	12	0	0	4	3	**5**	**5**
8	24	0	0	**1**	0	**1**	0