Capacity requirements and demand management strategies in meal delivery

Ramon Auad*[1, 2], Alan Erera†[1], and Martin Savelsbergh‡[1]

[1]School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA, USA
[2]Departamento de Ingeniería Industrial, Universidad Católica del Norte, Antofagasta, Chile

Abstract

Online restaurant aggregators, which connect diners with restaurants and organize the delivery of ordered meals, have experienced significant growth in recent years. Meal delivery logistics is quite challenging, primarily due to the difficulty in managing the supply of delivery resources, i.e., crowdsourced couriers, to satisfy dynamic and uncertain customer demand under very tight time constraints. In this paper, we study several questions in meal delivery operations focused on matching the correct levels of supply with demand. To ensure excellent customer service, delivery aggregators may, for example, decide to temporarily decrease demand during an operating day by temporarily reducing the delivery area for one or more restaurants. We show that such simple demand restriction strategies allow a significantly smaller fleet to meet service requirements. To simplify analysis, we focus on problem geometries that enable the use of stylized mixed integer programs to optimally deploy a fleet of couriers serving large numbers of orders. Applying the proposed framework to several scenarios with one and two depots, we conduct an extensive experimental study of the effects on system performance of (i) allowing courier sharing between multiple depots, (ii) relaxing the delivery deadlines of placed orders, and (iii) restricting demand through limited adjustment of the coverage of restaurants. The results demonstrate the potential effectiveness of different dispatch control and demand management mechanisms, in terms of both the required courier fleet size to serve requests and the coverage level of orders.

Keywords: Logistics; Meal delivery; Capacity planning; Last-mile delivery

*Corresponding author, rauad@gatech.edu, rauad@ucn.cl
†alan.erera@isye.gatech.edu
‡martin.savelsbergh@isye.gatech.edu

1 Introduction

Advances in technology are changing the logistics and transportation industry in profound ways and at a rapid rate. One important trend has been the rise of digital transportation platforms, including those provided by the online restaurant aggregator market segment. Companies in this segment operate meal-ordering platforms which provide a list of restaurants where customers can place orders, and then subsequently arrange for the ordered meals to be delivered upon request. In 2015, the food delivery market was worth \$11 billion in the US, and was predicted to grow at a 16% annual compound rate up to 2022, potentially to \$210 billion [1]. Similar growth has been observed in the online restaurant aggregator segment [2].

Online meal ordering is convenient for customers. In 2017, more than 40% of the US population reportedly replaced meals at restaurants by ordering food for delivery [1]. At the same time, more people appear to dislike shopping at grocery stores and cooking at home [3].

Although the statistics above focus on the US food delivery market, this phenomenon is also occurring in other parts of the world [4]. In 2018, the worldwide online food delivery market revenue reached \$82.7 billion (40% of which corresponds to China, and 20% to the US), and worldwide revenue is expected to increase to \$164 billion by 2024 [5]. India is also exhibiting similar trends, where the popularity and number of food delivery apps is steadily growing [6].

Technological advances have not only enabled new business models but have also led to new and complex decision problems in the efficient operation of meal delivery systems. At the heart of operations is the matching of delivery resources to orders over time. Customer service expectations are high: meals should be delivered by so-called *couriers* a short time after being placed and an even shorter time after becoming ready; [7] refer to meal delivery as the ultimate challenge in last-mile logistics. Not only is there significant variability in order arrival patterns [8], there is also significant variability in courier behavior since most systems allow couriers to reject some delivery requests and cannot control courier repositioning after deliveries.

This operating environment requires both effective matching technology, i.e., assigning orders to couriers, but also effective demand *and* supply management. Traditionally, demand management techniques use dynamic pricing, i.e., adjusting the price charged for deliveries from a particular restaurant, but, given the fact that the prices charged for deliveries are very low, this may not be effective. Furthermore, as real-time pricing of delivery is likely to cause a negative reaction in diners, platforms avoid its usage [9, 10]. Alternative strategies involve adjusting the service coverage area associated with a restaurant and/or redirecting diner demand to restaurants that are easier to serve by reordering search results. Some research to date covers the former, whereas the latter has received little attention [11]. Our research seeks to better understand the potential of demand management by adjusting the service coverage area (what we refer to later on as radius management).

The meal delivery environment we consider is as follows: during a given operating period, diners place orders to restaurants (in the remainder sometimes called depots to stay close to the terminology commonly used in the vehicle routing literature), and the aggregator must assign these orders to couriers in such a way to deliver (most) orders on time (i.e., at or before the delivery time promised to the diner when the order is placed) while minimizing operating costs. The goal of this paper is to analyze the fundamental relationship between service and cost metrics in these systems. We consider two different models of customer service, one which requires orders to be delivered by a hard deadline and the other which has a target delivery time but allows some fraction of orders to be delivered between the target time and a (later) hard deadline. We measure cost primarily by the number of couriers required during the operating period.

We develop optimization models that seek to determine the minimum number of couriers required to meet service requirements. To simplify analysis, we study the relationship between service and cost in three stylized settings: (i) a single depot at one extreme of a single line segment serving orders that must be delivered at points on the line segment, (ii) a single depot at the end of multiple line segments serving orders that must be delivered at points on the line segments, and (iii) two depots at the opposite extremes of a line segment serving orders that must be delivered at points on the line segment from a specific depot. For simplicity, we assume that couriers follow the instructions of the decision maker and never reject offered delivery orders. For each of the settings, we provide an integer programming (IP) formulation that assumes perfect information, i.e., order placement times and delivery locations are known in advance. The results from these models allow us to provide insights to the following fundamental questions:

- For a given service requirement, a given number of couriers, and a given order arrival rate, what fraction of orders can be served (i.e., delivered at or before the delivery time promise)?
- For a given service requirement and a given order arrival rate, what is the minimum number of couriers needed to serve all orders?
- For a given service guarantee, a given number of couriers, and a given order arrival rate, what is the largest coverage area that a depot can serve?

To summarize, the main contributions of our research are:

- Developing an IP framework for studying supply and demand management mechanisms for online meal delivery environments; and
- Performing an extensive experimental analysis of the fundamental trade-offs in meal delivery operations, which provides valuable insight into the benefits of supply and demand management mechanisms.

The remainder of the paper is organized as follows. Section 2 discusses the relevant literature. Section 3 presents a detailed description of the settings considered in our research and introduces the associated IP formulations. Section 4 summarizes the results of our computational experiments. Finally, Section 5 gives concluding remarks.

2 Literature review

In its most general setting, the problem we study in this work is one in the family of dynamic vehicle routing problems [12] and more specifically is a dynamic pickup and delivery problem (dPDP) [13]. The existing literature on these type of problems is vast and has grown significantly over the past few decades, mainly due to the advances in technology and telecommunication.

One of the applications for recent research in dPDP problems is transport of persons. Examples of such are dial-a-ride, dial-a-flight and ride-sharing. The latter problem is similar to our problem, with a common fleet of drivers that must satisfy transportation requests on short-notice, each characterized by an origin-destination pair with time-based service requirements [14]. Meal delivery problems are also part of the growing research area of dynamic delivery problems (dDP) [7], including same-day delivery problems. The growth of online retail in the last decades has attracted researchers to same-day delivery operations, with focus on both simplified analytic settings [15, 16, 17, 18] and real world situations [7, 19, 20, 21, 22]. In dDP problems, once a vehicle is dispatched to satisfy a set of deliveries, adjusting the route does not produce any benefit if travel times and costs do not change.

In the dDP literature, problems can be classified based on the availability of information. *Static* problems are those where all the information about orders and travel times is deterministic and known in advance [16, 17, 21]. *Dynamic* problems on the other hand, consist on settings where orders are revealed over time, and decisions are made only based on the revealed information [7, 20, 22]. If in addition, some of the parameters follow a probabilistic distribution and such information is available to the decision maker, then the problem is said to be stochastic. In our work we study a static meal delivery routing problem

Routing problems with time constraints are reviewed in [23]. Static dDP problems are closely related to the multi-trip VRP with release dates (VRP-rd), and with both release dates and deadlines (VRP-rdd). In these problems, couriers may perform multiple trips from the depot to serve orders that have an earliest ready time (release date) at the depot. Additionally, in VRP-rdd settings, deliveries to customers must occur before a deadline. An example of the VRP-rd is the work by [24]. Here, the authors propose the multi-trip VRP-rd with time windows, where a fleet of capacitated couriers must serve all orders minimizing total travel distance. They propose a hybrid genetic algorithm to solve the problem, and empirically show its effectiveness. [25] study the VRP-rdd and develop a path relinking algorithm to minimize the convex sum of the total distance and total positive deviations (delays) from the target order delivery times.

The problems we analyze use simplified network topologies (i.e., customers distributed on a line or a star network), which helps avoiding the complexity of the routing sub-problem (by turning them into a scheduling problem), thus primarily focusing on aspects related to capacity and demand management. There are multiple studies in the literature of transportation problems that simplify the routing component to focus on specific problem features, such as [11, 15, 16, 17, 26, 27, 28], some of which are able to meaningful insights despite the simplification. As we will see later, doing this allows us to easily model and solve different settings of interest that these works do not consider (e.g., multiple couriers, multiple depots, and problems with different objectives), and obtain managerial insights about the system's delivery capacity and its capability of managing demand. To handle these additional complexities, we use integer programming formulations built on underlying time-expanded networks, directed networks whose vertices are pairs with both a location and a time point component. The use of these networks allows more flexibility when modeling time dependencies. Some applications of time-expanded networks include service network design [29, 30] and the time dependent TSP with time windows [31]. However, flexibility comes at the cost of efficiency in solving [32].

3 Problem description

3.1 Single depot setting

We consider a single depot located at one end, $\tau_0 = 0$, of the line segment $[0, U]$ with $U > 0$. A set of orders, $N = \{1, \ldots, n\}$ is placed on the depot, where each order $j \in N$ specifies:

- a ready time $r_j \in [0, U] \cap \mathbb{Z}$, which defines the earliest time it can be dispatched for delivery;
- a location $\tau_j \in (0, U] \cap \mathbb{Z}$, representing its delivery location measured in travel time from the depot; and
- a due time $\Delta_j \geq r_j + \tau_j, \Delta_j \in \mathbb{Z}$ where if order j is not delivered by time Δ_j, it is considered late (and is potentially lost).

Let $\mathbf{T} \equiv [0, T]$ be the operating period. Without loss of generality, we assume $r_1 = 0$ and $r_j \leq r_{j+1}, \forall j \in N$ (with $r_{n+1} \equiv T$). Furthermore, at time r_j an available courier at the depot can be dispatched to deliver j along with any other orders i with $r_i \leq r_j$ (no courier capacity). We assume that the times required for a courier to pick up or deliver orders are negligible when compared to travel times. Thus, given an order set $J \subseteq N$ with $\tau_J \equiv \max_{j \in J}\{\tau_j\}$, a courier can deliver J and return to the depot in time $2\tau_J$. When J includes more than a single order, we say that the orders in J are *bundled*.

Suppose that there are $m \geq 1$ couriers that can make deliveries, each located at the depot at time 0 and required to return after their final delivery by time T (and therefore, $2U \leq T$; moreover, each order $j \in N$ is assumed to satisfy $r_j + 2\tau_j \leq T$ so it can be feasibly served). Let $S \geq \tau_N$ be the (common) maximum acceptable service time for each order, which implies that $\Delta_j \equiv r_j + S$ for $j \in N$.

In this setting, we consider two optimization problems: (1) maximize the number of orders that can be served on-time given m couriers, and (2) minimize the number of couriers m needed to serve all orders. Formally:

Problem 1 (Order maximization)**.** *Given m identical couriers, find a feasible delivery schedule for each of them that maximizes the total number of orders served, where a feasible delivery schedule for a courier specifies a number of delivery trips, each with a given departure time and a set of orders to deliver, such that all served orders $j \in N$ are ready at the time of departure and are delivered by their due time Δ_j.*

Problem 2 (Courier minimization)**.** *Find the minimum number of couriers (and a feasible delivery schedule for each of them) required to serve all orders $j \in N$ by their due time Δ_j.*

In the rest of this section we develop a mathematical framework for analyzing these problems which relies on integer programs defined on time-expanded networks.

3.1.1 Creation of a time-expanded network

Before giving a mathematical model for Problems 1 and 2, we provide a useful proposition; proofs for this and later results can all be found in Appendix A.

Proposition 1. *Consider an optimal schedule and let $J \subseteq N$ be a set of orders in that schedule with the same dispatch time t. Then, there exists an optimal schedule in which the orders in J are served by a single courier.*

The next result shows how to determine a sufficient finite subset of time points in $\mathbf{T}$ with the property that there exists an optimal schedule that only dispatches couriers at a subset of these points. Consider then the following definition:

Definition 1 (Active order)**.** We say that order j is active at time $t \in \mathbf{T}$ if $t \in \{r_j, r_j+1, \ldots, \Delta_j - \tau_j\}$ and it has not yet been dispatched by t. We denote the set of active orders at time t by $A(t)$.

Active orders at time t can be dispatched feasibly. We now introduce a lemma useful when modeling Problems 1 and 2.

Lemma 2. *Given $j \in N$, let $t \in [r_j, r_{j+1})$ be the earliest time that a courier is available for dispatch at the depot. Then there exists an optimal schedule for Problems 1 and 2 in which no courier is dispatched at any time $(t, r_{j+1}) \cap \mathbb{Z}$.*

From Lemma 2 it follows that the only necessary dispatch times at the depot are the ready times $\{r_j\}_{j \in N}$ and the courier return times $r_j + 2\sum_{k \in K} \tau_k$, for some $K \subseteq N$. We denote the set of such time points by $\mathcal{T}_0$.

The time-expanded networks we build also model couriers moving from one order delivery location to another or back to the depot. To determine which time points are required to model these movement decisions, let $t \in \mathbf{T}$ be a time point such that an optimal solution dispatches a courier from the depot at t with orders $J \subseteq A(t)$, and let $\{\tau_{(i)}\}_{i=1}^{|J|}$ be the locations of orders J sorted in non-decreasing order from the depot such that $\tau_{(1)}$ is closest. Then there exists an optimal solution where the courier visits locations $\tau_{(i)}$ sequentially at times $t + \tau_{(i)}$ for $i = 1, 2, \ldots, |J|$. After visiting location $\tau_{(|J|)}$ the courier returns to the depot, arriving at time $t + 2\tau_{(|J|)}$ either to be dispatched again immediately or, by Lemma 2, to wait until the next order arrival time.

At any dispatch time $t \in \mathcal{T}_0$ at the depot, an optimal solution will either decide not to dispatch a courier or to dispatch a courier with a subset $J' \subseteq A(t)$. The only optimal subsets are those that include *all* orders with locations $\tau_j \leq \tau_{(i^*)}$ where $\tau_{(i^*)}$ is the furthest order in J', and so each order j in the subset is delivered exactly at time $t + \tau_j$.

Thus, it should be clear that these problems can be solved by considering models that include a discrete set of time points, specifically a subset of the time points specified in Proposition 3 below:

Proposition 3. *To solve Problems 1 and 2, it suffices to consider courier schedule decisions at ready times $r_j, j \in N$, at potential return times to the depot $r_j + 2\sum_{k \in K} \tau_k, j \in N, K \subseteq N$, and at potential delivery times at the customers $r_j + \sum_{k \in K} 2\tau_k + \tau_i, j \in N, K \subseteq N, i \in A(r_j + \sum_{k \in K} 2\tau_k)$.*

Each time point of interest is also associated with a specific spatial location: all dispatches occur at the depot $\tau_0 = 0$, while deliveries are performed at locations $x = \tau_j, j \in N$. Consequently, we will define the nodes of our time-expanded network in the form (t, s), representing a location s in the line segment $[0, U]$ and an associated time point t. Nodes in the time-expanded network belong to one of two types, which we present in the following definition.

Definition 2 (Depot and non-depot node)**.** A node of a time-expanded network $(t, s) \in \mathcal{V}$ is called *depot node* if its spatial component s corresponds to a depot location; otherwise it is labeled as *non-depot node*.

The network construction routine is shown in detail in Appendix B.

3.1.2 Integer programming formulations

Once the time-expanded network $\mathcal{N} = (\mathcal{V}, \mathcal{A})$ is constructed, we can formulate Problems 1 and 2 as integer programs. For each $j \in N$, let $\mathcal{V}_j \equiv \{(t, \tau_j) \in \mathcal{V} : t \in \{r_j + \tau_j, \ldots, \Delta_j\}\}$ be the set of non-depot nodes at which order j may be delivered, and $\mathcal{V}_0 \equiv \{(t, 0) \in \mathcal{V} : t \in \mathcal{T}_0\}$ be the set of depot nodes (and note that $\mathcal{V} \equiv \bigcup_{j \in N \cup \{0\}} \mathcal{V}_j$). Moreover, for each $p \in \mathcal{V}$, let $\alpha_p^- \equiv \{q \in \mathcal{V} : (q,p) \in \mathcal{A}\}$ and $\alpha_p^+ \equiv \{q \in \mathcal{V} : (p,q) \in \mathcal{A}\}$.

The decision variables in these problems are:

$$
\begin{aligned}
z_{pq} &= \text{Number of couriers that traverse arc } (p,q) \in \mathcal{A} \\
v_{jp} &= \begin{cases} 1 & \text{if order } j \in N \text{ is delivered at node } p \in \mathcal{V}_j \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
$$

For a fixed courier fleet size m, a valid mixed-integer programming formulation for Problem 1 is given by

$$\max \sum_{j \in N} \sum_{p \in \mathcal{V}_j} v_{jp} \tag{1a}$$

$$\text{s.t.} \sum_{p \in \mathcal{V}_j} v_{jp} \leq 1, \quad \forall j \in N \tag{1b}$$

$$v_{jp} \leq \sum_{q \in \alpha_p^-} z_{qp}, \quad \forall j \in N, \quad \forall p \in \mathcal{V}_j \tag{1c}$$

$$\sum_{q \in \alpha_{(0,0)}^+} z_{(0,0),q} = m \tag{1d}$$

$$\sum_{p \in \alpha_{(0,T)}^-} z_{p,(0,T)} = m \tag{1e}$$

$$\sum_{p \in \alpha_q^-} z_{pq} = \sum_{r \in \alpha_q^+} z_{qr}, \quad \forall q \in \mathcal{V} \setminus \{(0,0),(0,T)\} \tag{1f}$$

$$v_{jp} \in \{0,1\}, \forall j \in N, \quad \forall p \in \mathcal{V}_j \tag{1g}$$

$$z_{pq} \in \begin{cases} \mathbb{R}_+ & \text{if } p,q \in \mathcal{V}_0, \\ \{0,1\} & \text{otherwise} \end{cases} \quad \forall (p,q) \in \mathcal{A} \tag{1h}$$

Objective (1a) seeks to maximize the number of served orders. Constraints (1b) and (1c) are related to order acceptance; each order $j \in N$ can only be delivered once and if this occurs at node $p \in \mathcal{V}_j$, then some courier must travel from a node $q \in \alpha_p^-$ to p. Constraints (1d) - (1f) are courier flow conservation constraints for all the network nodes. Constraints (1g) and (1h) enforce non-negative flows on depot arcs and binary flows elsewhere (due to Proposition 1).

Using a similar set of constraints and redefining the courier fleet size m as a decision variable, Problem 2 can be posed as

$$\begin{aligned}
&\min \quad m && \text{(2a)}\\
&\text{s.t.} \sum_{p\in\mathcal{V}_j} v_{jp} = 1, \quad \forall j \in N && \text{(2b)}\\
&\text{(1c)} - \text{(1h)} &&\\
&m \in \mathbb{R}_+ && \text{(2c)}
\end{aligned}$$

Note that Model (1) and (2) are always feasible. Moreover, the structure of these models grants them the property that for fixed values of variables v, the feasible-set polyhedron formed by variables z corresponds to one of a network flow model with integer extreme points. As a direct consequence, for each binary vector v there exists[1] an optimal vector z with only integer components. This is formalized next.

Proposition 4. *For fixed binary v, the set of feasible z in Models* (1) *and* (2) *describe a network flow polyhedron. Thus, for integer values of m, decision variables z will take integer values in an optimal solution.*

3.1.3 Incorporating lateness

In practical delivery problems, it is common that when a customer places an order, an *estimated time of arrival (ETA)* is announced and the operator seeks to serve the order no later than this time. In Models (1) and (2) we represent this idea by assuming that each order $j \in N$ must be served by Δ_j (if served at all). In this section, we consider alternative models that allow some orders to be served if they arrive late. To do so, in addition to the hard due time Δ_j of each order j, we introduce a *target* delivery time δ_j to represent the ETA by which order $j \in N$ is sought to be delivered. An order j delivered at $t \in (\delta_j, \Delta_j]$ is then considered *late*.

Mathematically, let $s \in \{\tau_N, \tau_N + 1, \dots, S\}$ be a (common) target service time for all orders. Then similar to how the maximum service time S determines the due time Δ_j for each $j \in N$, we now define $\delta_j \equiv r_j + s \leq \Delta_j$ as the target delivery time by which order $j \in N$ is desired to be delivered. From this definition we present a problem that seeks to minimize delivery lateness measured as the number of orders delivered after their target delivery time δ_j.

Problem 3 (Late Orders Minimization)**.** *Given a fleet of couriers of size m, find a schedule for each courier that serves every order $j \in N$ by Δ_j and such that the number of orders served later than δ_j is minimized.*

For a given order $j \in N$, let $\mathcal{L}_j \equiv \{(t, \tau_j) \in \mathcal{V}_j : \delta_j + 1 \leq t \leq \Delta_j\}$ be the set of late service nodes of j. Then Problem 3 is solved by the following integer program.

$$\min \sum_{j\in N} \sum_{p\in\mathcal{L}_j} v_{jp} \qquad \text{(3a)}$$
$$\text{s.t.} \;\; \text{(2b)}, \;\text{(1c)} - \text{(1h)}$$

Objective (3a) minimizes the number of orders served later than the target service time δ_j by penalizing the objective

[1]This is true for Model (1), as long as the fixed value of m is integer and allows feasibility for the fixed v

every time this occurs while all orders must be served by their due time Δ_j. Note that Problem 3 is feasible if and only if the number of couriers m in the input is at least the optimal value of Problem 2, as otherwise Constraint (2b) will lead to infeasibility.

Note that Problem 3 could use an alternative lateness-based objective. For example, the decision maker may prefer to minimize the *total aggregated lateness* over all the orders, giving a larger penalty to orders that are served closer to their maximum acceptable delivery time Δ_i. In our current formulation, this would only require replacing (3a) by the expression $\min \sum_{j \in N} \sum_{p \in \mathcal{L}_j} (t - \delta_j) v_{jp}$.

3.1.4 Radius management

In the earlier formulations, when determining the maximum number of orders that can be served by a fixed fleet of couriers the assumption was that the optimization model can selectively choose to provide or deny service to any individual order. Such a strategy is reasonable when determining an upper bound on maximum orders served in hindsight or with complete information. A potentially more realistic model for accepting or rejecting orders is to use a *service radius*: if an order is attempted to be placed at time t when the service radius is ρ, then the order must be served if $\tau \leq \rho$ and must be denied service otherwise.

In this section, we introduce modifications of the models to handle such radius-based order management decisions. In the basic model, we assume that a service radius is set at the beginning of the horizon and remains unchanged through the operating horizon. We formally state the decision problem as follows:

Problem 4 (Single Service Radius Maximization). *Given a fleet of m couriers, find a schedule for each and a service radius ρ that maximize the number of served orders, where each order $j \in N$ is served if and only if $\tau_j \leq \rho$.*

From Problem 4 we can develop a natural extension that selects a (potentially different) service radius at R different fixed times $\{t_1, \ldots, t_R\} \subseteq \mathbf{T}$, where $t_1 \equiv 0$. For any $t \in \mathbf{T}$, let ρ_ℓ be the active radius during time interval $[t_\ell, t_{\ell+1})$, $\ell \in \{1, \ldots, R\}$ (with $t_{R+1} \equiv T$). Then for order $j \in N$ where $r_j \in [t_\ell, t_{\ell+1})$, j is served if and only if $\tau_j \leq \rho_\ell$. We mathematically formulate this extension as follows.

Problem 5 (Fixed-Time Radius Management Problem). *Given a fleet of m couriers, find a schedule for each of them and service radii $\{\rho_\ell \in [0, \tau_N]\}_{\ell=1}^R$ that maximize the number of served orders, where if order $j \in N$ is such that $r_j \in [t_\ell, t_{\ell+1})$, then j is served if and only if $\tau_j \leq \rho_\ell$.*

Note from the problem definition that if some ready time r_j coincides with a radius shifting time t_ℓ, we assume the radius adjustment is performed right before the order is placed.

Since $\{t_\ell\}_{\ell=1}^R$ are given, we can model Problem 5 by augmenting Model (1) with a few additional constraints.

Proposition 5. *For each $\ell \in \{1, \ldots, R\}$, let $B_\ell \subseteq N$ be a list of orders such that (i) $j \in B_\ell$ if and only if $r_j \in [t_\ell, t_{\ell+1})$; and (ii) elements of B_ℓ are sorted in ascending order of travel time from the depot to their delivery location, with $B_{\ell,i}$ denoting the i-th element of list B_ℓ (and so $\tau_{B_{\ell,i}} \leq \tau_{B_{\ell,i+1}}$). Then for solving Problem 5, it* *suffices to solve the integer program resulting from combining Model* (1) *with the extra linear constraints*

$$\sum_{p\in\mathcal{V}_{B_{\ell,i}}} v_{B_{\ell,i},p} \begin{cases} \geq \sum_{p\in\mathcal{V}_{B_{\ell,i+1}}} v_{B_{\ell,i+1},p} & \textit{if } \tau_{B_{\ell,i}} < \tau_{B_{\ell,i+1}} \\ = \sum_{p\in\mathcal{V}_{B_{\ell,i+1}}} v_{B_{\ell,i+1},p} & \textit{if } \tau_{B_{\ell,i}} = \tau_{B_{\ell,i+1}} \end{cases}, \quad \begin{array}{c} \forall \ell \in \{1,\ldots,R\} \\ \forall i \in \{1,\ldots,|B_\ell|-1\} \end{array} \tag{4}$$

Moreover, given an optimal solution (v^*, z^*) *of the resulting model, each optimal service radius can be recovered by computing*

$$\rho^*_\ell = \max_{j\in B_\ell} \left\{ \tau_j : \sum_{p\in\mathcal{V}_j} v^*_{jp} = 1 \right\}, \quad \forall \ell \in \{1,\ldots,R\}$$

Adding Constraint set (4) forces an order to be served if the next furthest order placed from the depot during the same time interval ℓ is served while also forcing all orders during interval ℓ to be either served or not served if they have the same value of τ.

3.1.5 *L*-star extension

Although the setting considered up to now assumes that all the orders delivery locations lie in a single line segment with the depot at one of its extremes, our framework can easily be adapted to the more general case with an arbitrary L number of line segments radiating from the depot point. Note that this network topology assumes that all travel between line segments must transit the depot, and thus the only reasonable order bundles for dispatches are those where all orders are to be delivered in a common segment.

For $h \in \{1,\ldots,L\}$, let $N_h \subseteq N$ be the subset of N containing orders to be delivered in line segment h, with $n_h \equiv |N_h|$ and $\sum_{h=1}^{L} n_h = n$. Moreover, we assume that orders in each subset N_h are in ascending order of ready time. In addition, since $L \geq 1$ the delivery location of order $j \in N_h$ is now characterized by the pair (τ_j, h), representing a distance τ_j from the depot along the line segment h. As a result, nodes in the time-expanded network encode a time, a line segment and a distance from the depot. Defining $h = 0$ for nodes at the depot, we redefine depot nodes as $(t, 0, 0)$, and non-depot nodes as (t, τ_j, h).

Aside from these minor adjustments, the only procedure that requires a few additional considerations is the routine that creates the time-expanded networks. This is due to the dispatches at a returning time: at the time a courier returns from delivering orders at a particular line segment, it can either remain in the depot until the next order is ready or else be immediately dispatched into any of the L line segments with a new set of active orders. This implies that any depot node defined by the return of a courier defines an outbound arc to each of the L line segments containing the delivery location of an active order at that time. The adapted network building routine can be found in Appendix C.

Once the time-expanded network $\mathcal{N} = (\mathcal{V}, \mathcal{A})$ is created, consider the following redefinition of the sets of nodes: for $j \in N_h$, let $\mathcal{V}_j \equiv \{(t, \tau_j, h) \in \mathcal{V} : r_j + \tau_j \leq t \leq \Delta_j\}$ be the set of nodes where order j can be served. Similarly, let $\mathcal{V}_0 = \{(t, 0, 0) \in \mathcal{V}\}$. Lastly, for each $p \in \mathcal{V}$ the sets of adjacent nodes α^-_p and α^+_p are defined as in the previous section. With these modifications, the integer program formulations provided for problems from the previous section exactly model the corresponding L-star variant.

3.2 Two-depot setting

In Section 3.1 it is assumed that every order was placed to a single depot. In this section, we extend this setting to a single line segment with two depots, one located at each of its ends, that share a courier fleet to make deliveries. Customers place an order that is to be filled by a specific depot; for example, these locations may represent two different restaurants. Subject to some minor changes, we model and solve this case employing the same framework presented in Section 3.1.

Consider a line segment $[0, U]$ with $U > 0$, and two depots 1 and 2 located at $\tau_0 = 0$ and $\tau_U = U$, respectively. For depot $d \in \{1, 2\}$, let N_d be the set of orders that must be picked up from d, with $n_d \equiv |N_d|$ and $n \equiv n_1 + n_2$. Moreover, for each depot d we define

- A n_d-dimensional vector of ready times corresponding to orders placed at depot d, $\boldsymbol{r}^d \equiv (r_1^d, \ldots, r_{n_d}^d)$, whose components are sorted in increasing order. In the following, we label an order from depot d by j if the order has the j-th earliest ready time among the orders in such depot. Furthermore, without lost of generality, we assume $r_1^1 = 0$ and $r_1^2 \geq 0$.
- A corresponding n_d-dimensional vector of delivery locations $\boldsymbol{\tau}^d \equiv (\tau_1^d, \ldots, \tau_{n_d}^d)$, where the j-th component denotes the delivery location of order $j \in N_d$ *measured with respect to depot 1.*
- A corresponding n_d-dimensional vectors of target delivery times $\boldsymbol{\delta}^d \equiv (\delta_1^d, \ldots, \delta_{n_d}^d)$, and due times $\boldsymbol{\Delta}^d \equiv (\Delta_1^d, \ldots, \Delta_{n_d}^d)$.

Also, consider

$$\overline{d} \equiv \begin{cases} 2 & \text{if } d = 1 \\ 1 & \text{if } d = 2 \end{cases} \qquad \tilde{\tau}_j^d \equiv \begin{cases} \tau_j^d & \text{if } d = 1 \\ U - \tau_j^{\overline{d}} & \text{if } d = 2 \end{cases}$$

where $\overline{d}$ denotes the complement of depot d, and $\tilde{\tau}_j^d$ corresponds to the delivery location of order $j \in N_d$ measured from its depot d. Now we pose the two-depot version of the early problems as follows:

Problem 6 (Two-depot Order Maximization)**.** *Given depots 1 and 2, and a fleet of m identical couriers. Find a schedule for each courier that maximizes the total number of orders served in $N_1 \cup N_2$.*

Problem 7 (Two-depot Courier Minimization)**.** *Given two depots 1 and 2. Find the minimum number of identical couriers needed and a schedule for each of them, such that every order in $N_1 \cup N_2$ is served on time.*

An important assumption made is that for all the considered two-depot problems, the decision maker has the ability to select at which depot each courier starts operating.

3.2.1 Time-expanded network construction

Next we present a routine that constructs a time-expanded network $\mathcal{N} = (\mathcal{V}, \mathcal{A})$ with two depots. This new algorithm creates a time-expanded sub-network for each of the two depots (with their corresponding depot and non-depot nodes) that are connected through depot nodes. In order to make the distinction between both sub-networks explicit we redefine every node in the network as a tuple $v = (t, s, d)$, where s represents a spatial location, t a time point, and $d \in \{1, 2\}$ an associated sub-network based on a corresponding depot. Similarly, every arc (v_1, v_2) is now associated to a sub-network which is given by the sub-network of v_1. For the network construction routine, see Appendix D.

3.2.2 Integer program formulations

It is not hard to extend models from Section 3.1 to formulate Problems 6 and 7. Indeed, it suffices to redefine the decision variables and constraints used in the single depot models but incorporating into the existing notation the depot $d \in \{1,2\}$ sub-network to which each node belongs to. First consider the sets of time stamps at which depot nodes for each depot are defined by Algorithm 5, namely $\mathcal{T}_0$ and $\mathcal{T}_U$ for depots 1 and 2, respectively. For depot $d \in \{1,2\}$ and $j \in N_d$, let $\mathcal{V}_j^d \equiv \{(t, \tau_j, d) \in \mathcal{V} : r_j^d + \tilde{\tau}_j^d \le t \le \Delta_j^d\}$ be the nodes set where order j can be served. Similarly, let $\mathcal{V}_0 = \{(t,0,1) \in \mathcal{V}\}$ and $\mathcal{V}_U = \{(t,U,2) \in \mathcal{V}\}$ the sets of depot nodes at depot 1 and 2, respectively. Lastly, for each node $p \in \mathcal{V}$ consider the sets of adjacent nodes α_p^- and α_p^+ defined as in past sections. Then consider the following decision variables:

$$z_{pq} = \text{Number of couriers that traverse arc } (p,q) \in \mathcal{A}$$

$$v_{jp}^d = \begin{cases} 1 & \text{if order } j \in N_d \text{ is served at node } p \in \mathcal{V}_j^d,\ d \in \{1,2\} \\ 0 & \text{otherwise} \end{cases}$$

We can formulate Problem 6 as the following integer program.

$$\max \sum_{d=1}^{2} \sum_{j \in N_d} \sum_{p \in \mathcal{V}_j^d} v_{jp}^d \tag{5a}$$

$$\text{s.t.} \sum_{t \in \mathcal{T}_j} v_{jt}^d \le 1, \quad \forall d \in \{1,2\}, \quad \forall j \in N_d \tag{5b}$$

$$v_{jp}^d \le \sum_{q \in \alpha_p^-} z_{qp}, \quad \forall d \in \{1,2\}, \quad \forall j \in N_d, \quad \forall p \in \mathcal{V}_j^d \tag{5c}$$

$$\sum_{q \in \alpha_{(0,0,1)}^+} z_{(0,0,1),q} = m \tag{5d}$$

$$\sum_{p \in \alpha_{(T,0,1)}^-} z_{p,(T,0,1)} = m \tag{5e}$$

$$\sum_{p \in \alpha_q^-} z_{pq} = \sum_{r \in \alpha_q^+} z_{qr}, \quad \forall q \in \mathcal{V} \setminus \{(0,0,1),(T,0,1)\} \tag{5f}$$

$$v_{jp}^d \in \{0,1\}, \quad \forall d \in \{1,2\}, \quad \forall j \in N_d, \quad \forall p \in \mathcal{V}_j^d \tag{5g}$$

$$z_{pq} \in \begin{cases} \mathbb{R}_+ & \text{if } p,q \in \mathcal{V}_0 \cup \mathcal{V}_U \\ \{0,1\} & \text{otherwise} \end{cases}, \quad \forall (p,q) \in \mathcal{A} \tag{5h}$$

Model (5) is very similar to Model (1). Objective (5a) seeks to maximize the total number of served orders. For each depot $d \in \{1,2\}$, Constraint (5b) enforces that each order $j \in N_d$ is served at most once. Constraint (5c) requires a courier at node $(t, \tau_j, d) \in \mathcal{V}_j^d$ for order $j \in N_d$ to be served at time t. Constraints (5d) - (5f) are courier flow constraints; note that the insertion of arc $((0,0,1),(r_1^2,U,2))$ allows the model to select how many of the m couriers start the operating horizon at each depot, and arc $((T,U,2),(T,0,1))$ leaves $(T,0,1)$ as the unique sink in network $\mathcal{N}$. Lastly, constraints (5g) and (5h) specify the domain of the decision variables.

Similarly, a formulation for Problem 7 can be obtained by extending Model (2) as follows:

$$
\begin{aligned}
\min \quad & m && \text{(6a)}\\
\text{s.t.} \quad & \sum_{p \in \mathcal{V}_j^d} v_{jp}^d = 1, \quad \forall d \in \{1,2\}, \quad \forall j \in N_d && \text{(6b)}\\
& \text{(5c)} - \text{(5h)} \\
& m \in \mathbb{R}_+ && \text{(6c)}
\end{aligned}
$$

The similarities between formulations for the single depot and two-depot cases allow to preserve the result in Proposition 4 for two-depot models.

3.2.3 Incorporating lateness

Now we introduce the notion of lateness for two-depot models by incorporating the target delivery time δ_j^d.

Problem 8 (Two-depot Late Orders Minimization)**.** *Given depots 1 and 2 and a fleet of m identical couriers. Find a schedule for each courier such that every order $j \in N_d$, $d \in \{1,2\}$ is served by its due time Δ_j^d and such that the number of orders served after the target delivery time δ_j^d is minimized.*

For $d \in \{1,2\}$ and $j \in N_d$, let $\mathcal{L}_j^d \equiv \{(\tau_j, t, d) \in \mathcal{V}_j^d : \delta_j^d + 1 \leq t \leq \Delta_j^d\}$. Then Problem 8 is formulated as

$$
\begin{aligned}
&\min \sum_{d=1}^{2} \sum_{j \in N_d} \sum_{p \in \mathcal{L}_j^d} v_{jp}^d && \text{(7a)}\\
&\text{s.t. (6b), (5c)} - \text{(5h)}
\end{aligned}
$$

Model (7) minimizes the number of orders that are served after the corresponding target delivery δ_j^d subject to every orders being served by its due time Δ_j^d (Constraint (6b)) and courier flow constraints (5c) - (5h). Feasibility of Problem 8 is equivalent to the input number of couriers m being at least the optimal value of Problem 7, as otherwise Constraint (6b) cannot be satisfied.

3.2.4 Radius management

We adapt the radius management models from Section 3.1 to the two-depot case. In this setting, each depot $d \in \{1,2\}$ may select a service radius up to $R_d \geq 1$ times during the operating horizon, with the first radius being selected at $t_1^d \equiv r_j^d$, and any order whose delivery location lies inside the active radius at the moment of its placement must be served. Mathematically, let $\{t_\ell^d\}_{\ell=1}^{R_d}$ be the times at which depot d changes its radius, and let ρ_ℓ^d be the radius selected at time t_ℓ^d. If order $j \in N_d$ satisfies $r_j^d \in [t_\ell^d, t_{\ell+1}^d)$ (with $t_{R_d+1}^d \equiv T$), then j is served if and only if and $\tilde{\tau}_j^d \leq \rho_\ell^d$.

Now we present the two-depot version of the problem:

Problem 9 (Two-depot Fixed-Time Service Radius Management Problem)**.** *Given depots 1 and 2 and a fleet of*

m identical couriers. Find a schedule for each courier and service radii $\{\rho_\ell^d \in [0, \max_{i\in N_d}\{\tilde{\tau}_i^d\}]\}_{\ell=1}^{R_d}$ for depots $d \in \{1,2\}$ such that the total number of served orders is maximized, where if some order $j \in N_d$ is such that $r_j^d \in [t_\ell^d, t_{\ell+1}^d)$, then such order is served if and only if $\tilde{\tau}_j^d \leq \rho_\ell^d$.

In order to solve Problem 9, it is enough to add a small set of linear constraints to Model (5):

Proposition 6. *Let $\{B_\ell^d\}_{\ell=1}^{R_d}$ be the list of orders j such that $r_j^d \in [t_\ell^d, t_{\ell+1}^d)$ for depot $d \in \{1,2\}$, obtained from Algorithm 1 by replacing τ_j^d in its input by $\tilde{\tau}_j{}^d$; and let $B_{\ell,i}^d$ be the i-th element of list B_ℓ^d. Then Problem 9 can be formulated by adding the following linear constraints to Model* (5)*:*

$$\sum_{p\in\mathcal{V}^d_{B^d_{\ell,i}}} v^d_{B^d_{\ell,i},p} \begin{cases} \geq \sum_{p\in\mathcal{V}^d_{B^d_{\ell,i+1}}} v_{B^d_{\ell,i+1},p} & \text{if } \tilde{\tau}^d_{B^d_{\ell,i}} < \tilde{\tau}^d_{B^d_{\ell,i+1}} \\ = \sum_{p\in\mathcal{V}^d_{B^d_{\ell,i+1}}} v_{B^d_{\ell,i+1},p} & \text{if } \tilde{\tau}_{B^d_{\ell,i}} = \tilde{\tau}_{B^d_{\ell,i+1}} \end{cases}, \quad \begin{array}{c} \forall d \in \{1,2\} \\ \forall \ell \in \{1,\ldots,R_d\} \\ \forall i \in \{1,\ldots,|B_\ell^d|-1\} \end{array} \tag{8}$$

Moreover, given an optimal solution (v^{1*}, v^{2*}, z^*) for the resulting model, the optimal service radii can be determined as

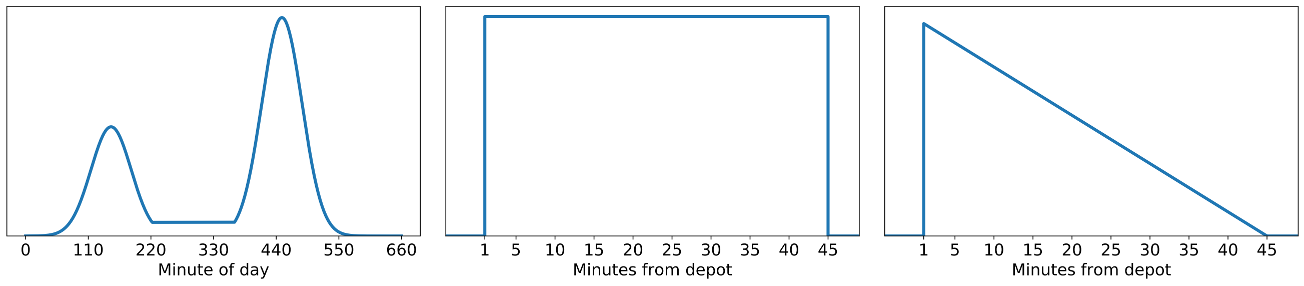
$$\rho_\ell^{d*} = \max_{j\in B_\ell}\left\{\tilde{\tau}_j^d : \sum_{p\in\mathcal{V}_j^p} v_{jp}^{d*} = 1\right\}, \;\; \forall d \in \{1,2\}, \;\; \forall \ell \in \{1,\ldots,R_d\}$$

4 Experimental results

In this section we report results from solving the proposed models on various instances to gain insights about required fleet sizes and demand management strategies in meal delivery systems. The conducted experiments are separated into three subsections which, respectively, provide understanding about (i) the minimum fleet sizes required to serve delivery requests for various instances, (ii) the value of establishing an individual target delivery time to manage delivery lateness, and (iii) the potential benefits of dynamically adjusting a depot coverage radius as a demand management strategy.

Tested values for parameters n, S, m and s vary with the type of experiment and are shown at the beginning of each experiment subsection. The remaining parameters are either assumed constant or defined as a function of the aforementioned ones. In particular:

- All the instances consider a time horizon length of $T = 660$ minutes (11 hours).
- Order ready times r_i^d are obtained by randomly sampling from a continuous bimodal distribution and rounding each element to its nearest integer, as shown in Figure 1a.
- Travel times τ_j^d are obtained by first drawing a random number from a continuous distribution and then rounding each element to its nearest integer (minute). The distribution we use depends on the number of depots considered:
  - For single depot settings we use a uniform distribution and a triangular distribution to sample delivery locations. These are illustrated in Figures 1b and 1c, respectively.



(a) Bimodal distribution of ready times r_j (b) $\tau_j \sim Unif(1,45)$ (c) $\tau_j \sim Tria(1,1,45)$

Figure 1: Sample distributions

Setting	n	S
Single depot	$\{75, 100, 120, 150\}$	$\{45, 50, 55, 60\}$
Two depots	$\{150, 200, 240, 300\}$	$\{45, 50, 55, 60\}$

Table 1: Parameter values used for fleet size minimization experiments

  – For the two-depot scenario we test two different levels of separation, $U \in \{60, 90\}$ minutes, each with its own sampling scheme. For $U = 60$ travel time to delivery locations of orders from depot 1 and 2 are sampled from triangular distributions $Tria(1, 1, 45)$ and $Tria(15, 59, 59)$, respectively. On the other hand, for $U = 90$ travel times to delivery locations of orders from depot 1 and 2 are generated from triangular distributions $Tria(1, 1, 45)$ and $Tria(45, 89, 89)$, respectively.

- All single depot instance settings consider $L = 4$ line segments. Moreover, orders are assumed to be distributed between the L line segments in such a way that for any two line segments, the numbers of orders to be delivered in each never differ by more than one order.

- For the two-depot scenario we further consider that each order is assigned to a specific depot at random with equal probability.

The bimodal sampling distribution for orders ready times in Figure 1a is motivated in realistic meal delivery operations, where it is observed that meal delivery demand is highly concentrated at lunch and dinner times. For the single depot setup, we consider two configurations for customer locations, namely one where they are mostly in the proximity of the depot (triangular distribution), and another one where customers are homogeneously located across the line segments (uniform distribution); these configurations allow us to study the effect of customer locations with the system's delivery capacity requirements and order coverage. For two depots, we consider two levels of separation between them, to study its effect in the system's capability to have couriers serve orders from both depots, and ultimately analyze the benefits of sharing couriers. For the instances we optimally solve, the corresponding running times are provided in Appendix E.

4.1 Minimum fleet size

In this section we solve problems that determine the minimum courier fleet size required to serve all orders in a specific instance. Parameter values considered for this section are summarized in Table 1. We run 50 replications for each possible combination of settings and parameter values and report statistics on optimal courier fleet size

m^*, and on bundle size, namely the number of orders in a single dispatch. Additionally, for the two-depot setting we analyze the number of crosses between both depots, proposing an auxiliary integer program that is able to show exactly when fleet sharing between depots yields a better operational cost than having each depot delivering orders with its exclusive fleet.

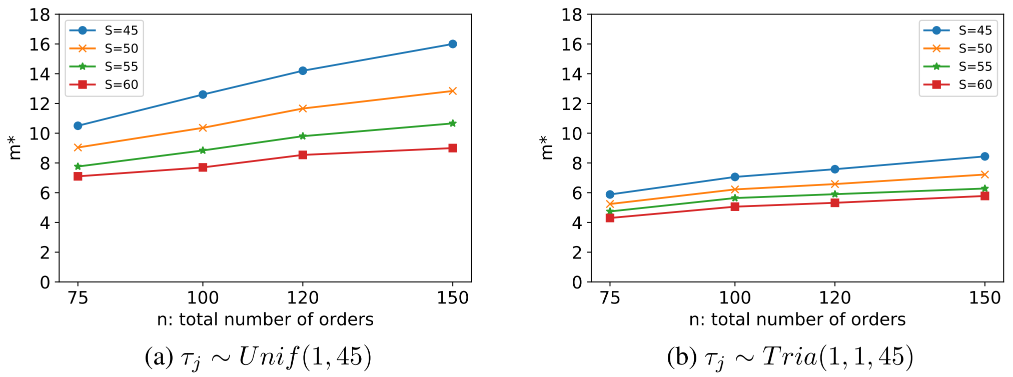
4.1.1 Single depot case

Table 2 presents the obtained average fleet size values m^* for some of the considered single depot instances, and Figure 2 illustrates the effect of the different parameters involved on the optimal fleet size. As expected, m^* increases as either n increases, S decreases, or order delivery locations become more distant to the depot. However, these effects on m^* differ in magnitude. To illustrate this, consider the scenario $(n, S) = (75, 60)$ as a base case, which corresponds to the case with fewest orders and most flexible due times. Observe that everything else equal, doubling the number of orders to $n = 150$ leads to a 30% increase in m^*; on the other hand, decreasing S by 25% to $S = 45$ while keeping all other values constant requires a relatively higher 44% increase in fleet size.

This difference in the effect on m^* is explained by both the bundling of multiple orders in a single dispatch and by the reduction in the dwell time of a courier who has returned to the depot before leaving for a subsequent dispatch. Since couriers are modeled assuming no limit on bundled orders, increasing the number of orders n tends to increase the number of orders bundled per dispatch and reduce the courier dwell time at the depot thus increasing courier productivity; the fleet size growth is modest with n. However, increasing the tightness of the delivery windows by decreasing S makes bundling orders less likely overall thus resulting in faster growth in required fleet sizes when n grows for smaller S. We note that, although bundle sizes are not constrained, the average numbers of orders bundled together for delivery lies between 1.5 and 3 for all instances; these averages are illustrated in Figure 3.

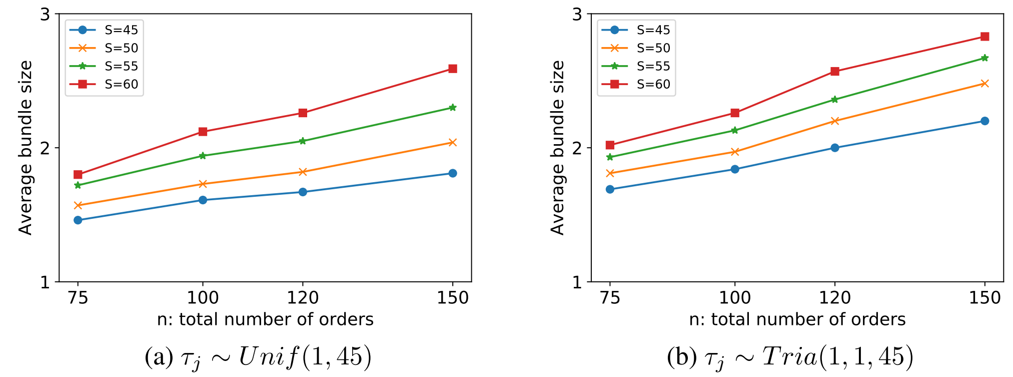
(n, S)	$\tau_j \sim Unif(1, 45)$	$\tau_j \sim Tria(1, 1, 45)$	Average
$(75, 45)$	10.50	5.88	8.19
$(150, 45)$	16.00	8.44	12.22
$(75, 60)$	7.10	4.30	5.70
$(150, 60)$	9.00	5.78	7.39
Average	10.65	6.10	

Table 2: Average m^* for some single depot configurations



(a) $\tau_j \sim Unif(1, 45)$ (b) $\tau_j \sim Tria(1, 1, 45)$

Figure 2: Average m^* for a single depot



(a) $\tau_j \sim Unif(1, 45)$ (b) $\tau_j \sim Tria(1, 1, 45)$

Figure 3: Average bundle size for a single depot

The customer location distribution also plays an important role in determining the optimal fleet size. Indeed, switching the location distribution from triangular to uniform results in an average 75% increase in the required minimum fleet size; we note that this change substantially increases the average distance from the depot to a customer and concomitantly the duration of any given delivery dispatch.

4.1.2 Two-depot case

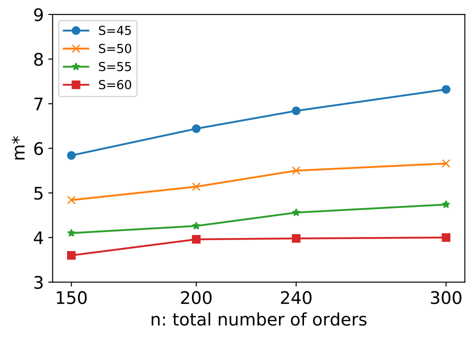
Results for the two-depot minimum fleet sizes m^* and how they vary with n, S and U are depicted by Figure 4, and partial results are reported in Table 3. As in the single depot case, consider the base case $(n, S) = (150, 60)$ which has the least number of orders and the most flexible delivery due times. Note that doubling n while preserving S results in an average 15.6% larger m^*, whereas only decreasing S to 45 minutes leads to a substantial increase of 60.4% in the required fleet size. Again, the ability to build larger bundles with larger values of S is critical to keeping fleet sizes from growing too large.

(n,S)	$U=60$	$U=90$	Average
$(150,45)$	5.84	6.70	6.27
$(300,45)$	7.32	8.56	7.94
$(150,60)$	3.60	4.22	3.91
$(300,60)$	4.00	5.04	4.52
Average	5.19	6.13	

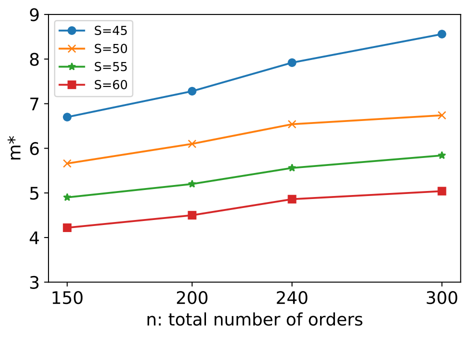
Table 3: Average m^* for two depots

(n,S)	Average m^*
$(150,45)$	6.77
$(300,45)$	8.77
$(150,60)$	4.25
$(300,60)$	5.33
Average	6.28

Table 4: Average m^* for two depots (without fleet sharing)



(a) $U=60$

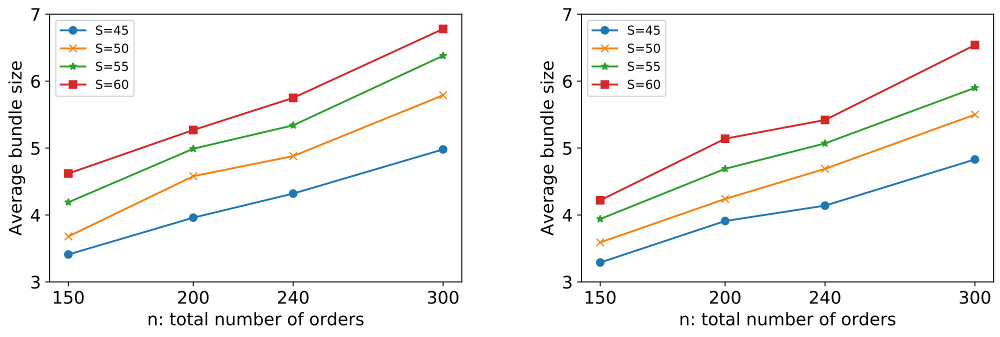


(b) $U=90$

Figure 4: Average m^* for two depots

Taking a closer look at the level of separation between depots, we observe that increasing U from 60 to 90 requires an 18% larger average fleet size. This difference is explained by the potential benefit for sharing couriers in the fleet between depots and this benefit diminishes when the time required to transfer from one depot to another (after serving a final customer) grows large when compared to the time required to return to the original depot. Table 3 summarizes specific minimum fleet sizes for some representative values of n and S, and can be compared directly to the results in Table 4 that compute minimum fleet sizes when dedicated fleets are used at each depot. Our findings from this comparison show that without fleet sharing, the system would require a 21% larger number of couriers compared to the shared fleet size for $U=60$. On the other hand, for a larger separation of $U=90$ the benefit from fleet sharing is only 2.4%.

In terms of bundling, the average bundle size and its relationship with n, S and U are presented in Figure 5. As previously mentioned, we observe that the average bundle size follows a similar pattern with respect to n and S as the one observed for a single depot. We also see that the average bundle size is slightly larger comparatively for the $U=60$ instances versus the $U=90$ instances and this is consistent with the smaller fleet sizes required for the former. Finally, it should also be noted that the bundle sizes in these two depot instances are roughly twice the size



(a) $U = 60$ (b) $U = 90$

Figure 5: Average bundle size for two depots

of those for single depot instances. This is due to the fact that the same number of total orders are distributed over two line segments (one from depot 1 and the other from depot 2) in these instances and distributed over four line segments in the single depot instances, so the order density per time is effectively doubled. It is also interesting to analyze how often couriers cross the line segment from one depot to the other in these two-depot instances. To do so, we solve a second integer program for each instance that, for a given optimal fleet size m^*, computes the *minimum number of courier crosses*, i.e., the minimum number of times that a courier is required to go from one depot to the other, in order to serve all the orders feasibly. Let $\mathcal{A}^*$ be the set of arcs that traverse from a non-depot node to a depot node such that both nodes are associated with different depot sub-networks, and let m^* be the optimal fleet size obtained from solving Problem 7. Then the integer program is formulated as follows:

$$\min \sum_{(p,q)\in\mathcal{A}^*} z_{pq} \tag{9a}$$

$$\text{s.t.} \quad (6b), (5c), (5f) - (5h)$$

$$\sum_{q\in\delta^+_{(0,0,1)}} z_{(0,0,1),q} = m^* \tag{9b}$$

$$\sum_{p\in\delta^-_{(T,0,1)}} z_{p,(T,0,1)} = m^* \tag{9c}$$

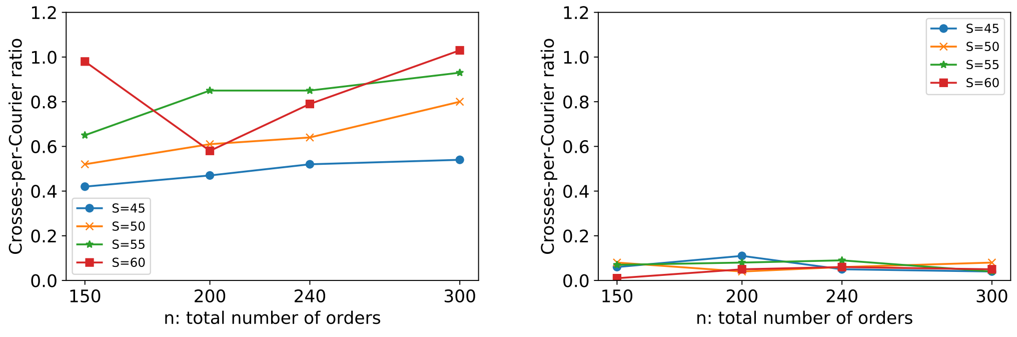
Objective (9a) minimizes the number of courier crosses between depots. In addition, we replace the decision variable m by the optimal fleet size m^* in constraints (9b) and (9c).

Due to the optimality of m^*, a key property of this auxiliary integer program is that it yields an optimal value of 0 crosses if and only if solving the two-depot instance with fleet sharing does not give any savings in the number of couriers with respect to employing individual fleets. Therefore, a strictly positive number of crosses reveals potential benefits from allowing a shared fleet for both depots. Table 5 shows the percentage of solved instances for which fleet sharing results in strictly fewer couriers than using dedicated courier fleets for each of the depots. Almost all instances yield a strictly positive minimum number of crossings for $U = 60$, but when $U = 90$ the benefit is much more limited.

(n, S)	$U = 60$	$U = 90$
(150, 45)	84%	28%
(150, 60)	62%	4%
(300, 45)	96%	28%
(300, 60)	90%	22%

Table 5: Percentage of instances whose optimal number of crossings is strictly positive

In order to control for the possible impact of the fleet size m^* on the optimal number of crosses, consider the ratio between the number of crosses and m^*, which we denote by γ and depict in Figure 6. The effect of the level of separation U on γ is evident: the operation exploits the short inter-depot traveling times when $U = 60$ by dynamically reallocating couriers between depots. On the other hand, the larger value $U = 90$ leads to inter-depot traveling that is too time consuming to be effective, and therefore the number of crosses per courier is usually below one in ten.



(a) $U = 60$ (b) $U = 90$

Figure 6: Average γ for different instance settings

Interestingly, when $U = 60$ we observe that in general, γ is non-decreasing in both S and n, with the exception of the case $S = 60$. In particular, we observe a significant decrease in γ when orders are increased from 150 to 200 which also corresponds to a significant increase in the minimum fleet size m^*. As n is further increased, the fleet size does not increase and more orders are handled by the same number of couriers, many of which execute crosses from one depot to the other. Hence, more crosses-per-courier are observed.

4.2 Analysis of lateness via target delivery time

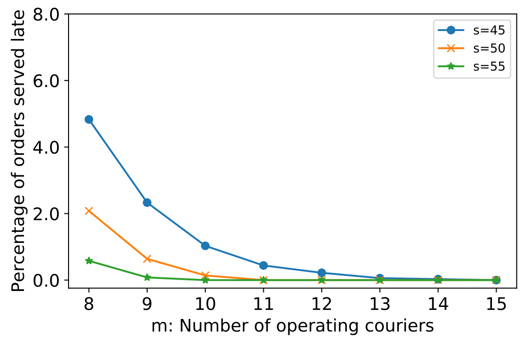
Now we report findings from solving problems that minimize the number of late orders. The results in this section demonstrate the benefit of introducing a target due time $\delta_j = r_j + s$ as a simple approach for balancing the flexibility of the system between setting too large and too restrictive due times $\Delta_j = r_j + S$. We empirically show that combining a restrictive target service level s with a flexible service level S can be effective, leading to solutions with very few late orders that use significantly fewer couriers than more restrictive settings with tighter values of S. The experiments considered in this section were conducted using the parameter values listed in Table 6. Note that the selected value of S used in this section corresponds to its most flexible value in Section 4.1, whereas the range of values of s begins with the most restrictive value. For each combination of parameters (n, S, s, U) we randomly generate 50 stream of orders and compute the minimum fraction of orders delivered after their target due time δ_j for different number of available couriers m. To preserve feasibility, we only consider fleet sizes no less than the minimum number of couriers required to serve all orders by their due time Δ_j.

Setting	n	S	s
Single depot	$\{75, 150\}$	$\{60\}$	$\{45, 50, 55\}$
Two depots	$\{150, 300\}$	$\{60\}$	$\{45, 50, 55\}$

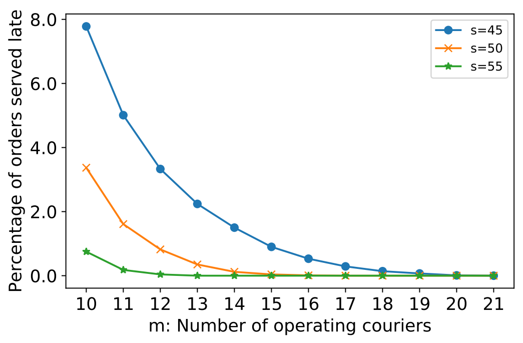
Table 6: Parameter values used for lateness experiments

4.2.1 Single depot case

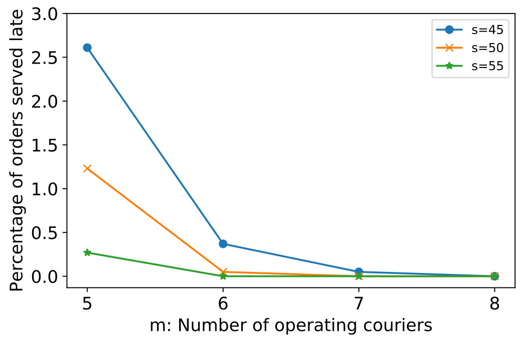
The results for this section are presented in Figure 7. We observe that the delivery location proximity to the depot has a considerable effect on the fleet size required to maintain a given level of lateness. Indeed, for some values of (n, S, s), maintaining a given percentage of late orders requires a fleet size that can be over 100% larger for the uniformly-distributed locations case when compared to the triangular distribution locations.



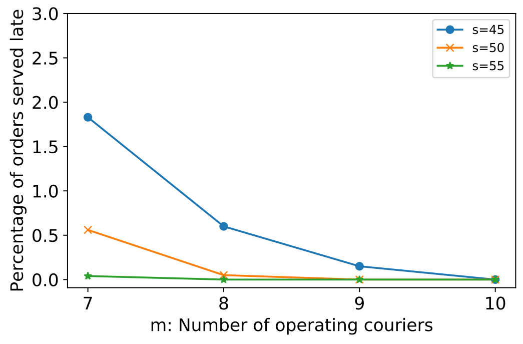
(a) $n = 75, \tau_j \sim Unif(1, 45)$



(b) $n = 150, \tau_j \sim Unif(1, 45)$



(c) $n = 75, \tau_j \sim Tria(1, 1, 45)$



(d) $n = 150, \tau_j \sim Tria(1, 1, 45)$

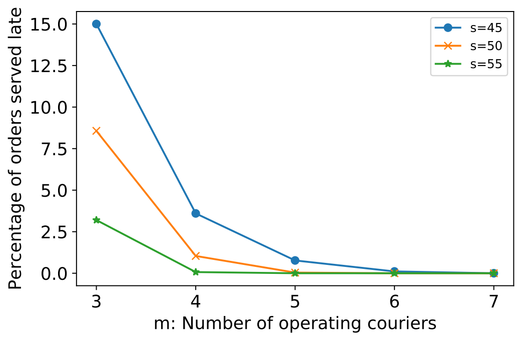
Figure 7: Average fraction of late orders for a single depot

As expected, our findings show that the most critical factor for determining the fraction of late orders is s. A small enough value of this parameter strongly restricts the flexibility of the system, leading to a substantial increase in the fraction of late orders for smaller fleet sizes. This is caused since lower values of s restrict the bundling opportunities for orders to be delivered before the target delivery time.

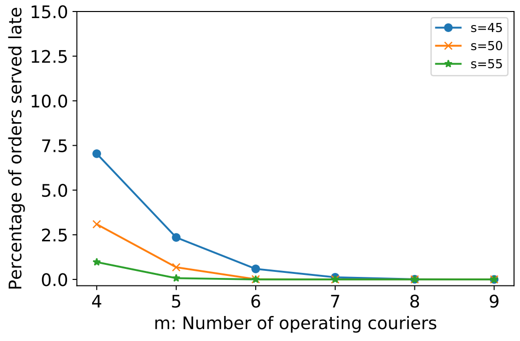
Lastly, we compare the setting that considers both target and hard due times δ_j and Δ_j against the case that only includes due times Δ_j. Note that only considering a hard due time corresponds to the particular scenario of having a target due time that satisfies $s = S$. To illustrate the benefits from having a target due time, consider the instances with $n = 150$ orders where delivery locations are uniformly distributed. Note that if $S = s = 45$ minutes, then on average about $m = 16$ couriers are needed to achieve on-time service. However, if the maximum delivery time S is relaxed to 60 minutes while maintaining $s = 45$, a 33% smaller courier fleet still manages to serve all 150 orders with only 5% of them delivered late (after δ_j). Of course, even fewer couriers would be required by setting $S = s = 60$ minutes but the average time to delivery of the orders would increase.

4.2.2 Two-depot case

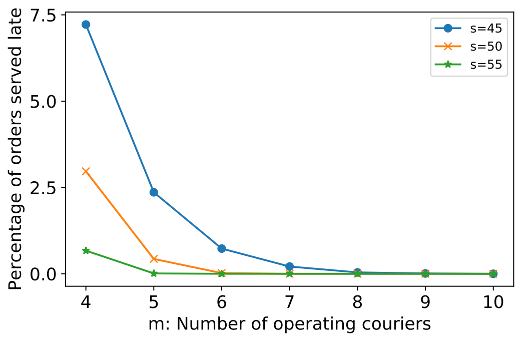
The effects from s and n on the fraction of late orders in instances with two depots are similar to those observed for a single depot and are shown in Figure 8. We see that varying U has a significant effect on the number of late orders. For small fleet sizes for which flexibility is limited, increasing U from 60 to 90 can on average scale up the fraction of late orders by a factor of 2.



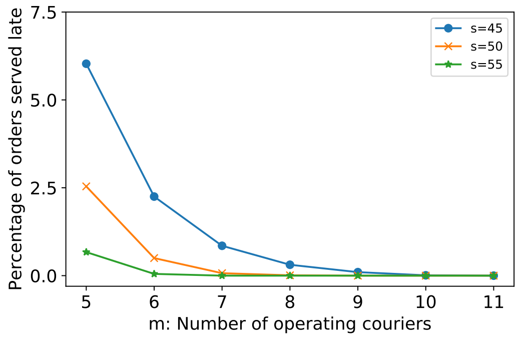
(a) $n = 150, U = 60$



(b) $n = 150, U = 90$



(c) $n = 300, U = 60$



(d) $n = 300, U = 90$

Figure 8: Average fraction of late orders for two depots

Additionally, we again observe some advantages from considering both a target delivery time and a hard due time. As shown in Figure 8 for $n = 300$ orders and a time between depots of $U = 60$, the simple case $S = s = 45$ results in a conservative solution that requires a total of $m = 8$ couriers in order to achieve full service with no late orders on average. Alternatively, relaxing S to 60 minutes while keeping $s = 45$ offers a reasonably more flexible solution that is able to serve all orders with an approximately 30% smaller fleet with only 2.5% of orders served late. Similar results can be observed for the remaining (n, U) pairs: a substantial reduction of 33% of the fleet that serves all orders without lateness results in a mild increase of late orders of less than 5%.

4.3 Demand management via service radius adjustments

In this section we study *order demand management* using our modeling framework. Specifically, we consider demand management strategies that are driven by a selected *service radius* from the depot (from which the order is placed). Radius-based strategies are simple: once a radius length ρ is selected, deliveries must be made to any order placed by a customer with a delivery location τ within the disk around the depot with radius ρ; in our simple geometric settings, this corresponds to $\tau \leq \rho$. We will compare the performance of demand management strategies when the service radius is selected and fixed in advance to those where the radius may change during the operating day using Problems 5 and 9.

Tables 7 and 8 summarize the parameters used in this section, where R_d measures the number of times the service radius is adjusted during the operating day. The case $R_d = 1$ is referred to as the *base case* and consists of simply setting a unique radius beginning at time $t = 0$. On the other hand, the case $R_d = \infty$ is referred to as the *selective-service case* and, since the radius can change at every order ready time, is equivalent to the settings of Problems 1 and 6 where the operator selectively chooses to either accept or reject each order; while this case is unrealistic in practice, it provides an upper bound on system performance. To measure the effectiveness of radius-based demand management, we experiment with 50 randomly-generated order streams and focus primarily on the fraction of the n orders served as a function of the amount of available resources.

Setting	n	S	R_d
Single depot	$\{75, 150\}$	$\{45, 60\}$	$\{1, 2, 5, \infty\}$
Two depots	$\{75, 150\}$	$\{45, 60\}$	$\{1, 2, 5, \infty\}$

Table 7: Parameter values used for demand management experiments

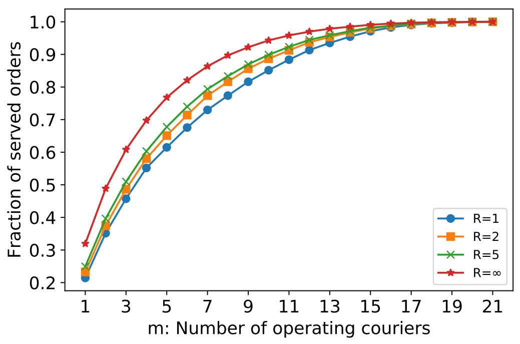
R_d	1	2	5	∞
$\{t_l^d\}_{l=1}^{R_d}$	$\{0\}$	$\{0, 300\}$	$\{0, 100, 200, 400, 500\}$	$\{r_j^d\}_{j \in N_d}$

Table 8: Times at which the service radius may change

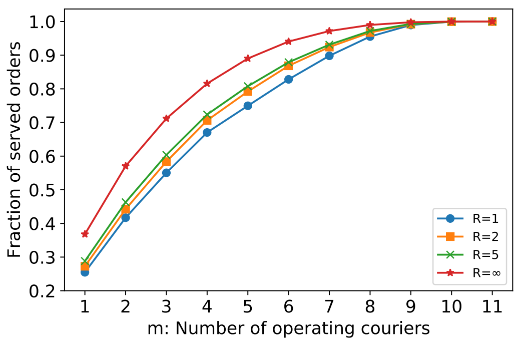
4.3.1 Single depot case

Figure 9 shows the fraction of served orders in optimal solutions to Problem 5 for different values of R when $n = 150$; the subfigures provide results for different combinations of S and the travel time distribution. Problems were solved for fleet sizes m from one to the minimum fleet size required to serve all orders in all instances.

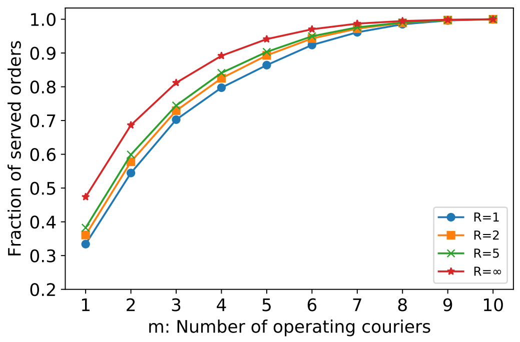
Although increasing R results in more flexibility for the operator to decide when and where to accept orders, the value of this flexibility depends on the available fleet size. In general, the results indicate that the largest benefit of increasing R occurs for instances with fleets of medium size (not too small and not too large). In such cases, increasing R from $R = 1$ to $R = 2$ can on average close approximately 30% of the gap to the upper bound; increasing to $R = 5$ closes the gap by approximately 50%. Systems with larger fleets intuitively benefit less from



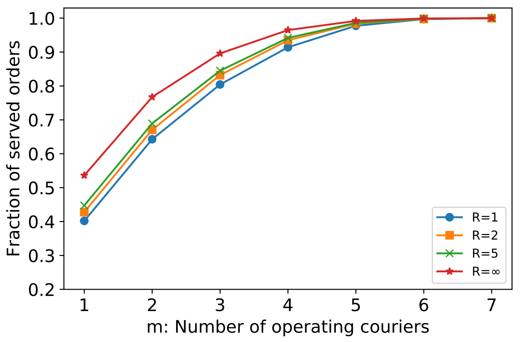
(a) $S = 45, \tau_j \sim Unif(1, 45)$



(b) $S = 60, \tau_j \sim Unif(1, 45)$



(c) $S = 45, \tau_j \sim Tria(1, 1, 45)$



(d) $S = 60, \tau_j \sim Tria(1, 1, 45)$

Figure 9: Fraction of served orders with $n = 150$ for a single depot

increased values of R. However, we also see that the smallest fleets that can only serve a small fraction of the orders do not benefit substantially from small increases in R; larger jumps in the fraction of orders served only occur when individual orders can be accepted or rejected (as in the $R = \infty$ case).

The results also help us understand the potential fleet size savings that can be achieved when the objective is to serve some fixed fraction of potential orders as R is increased. For example, when $n = 150$, $S = 45$, and the order location distribution is uniform, 21 couriers are required to serve all orders. However, using a radius-based demand management scheme with $R = 1$ leads to a fleet size requirement of 14 couriers to serve 95% of all orders. This fleet can be reduced again to 13 couriers when $R = 2$. Table 9 summarizes the results for a large set of scenarios and show that significantly smaller fleets can lead to reasonable order coverage fractions. We also see that it is typical when we require 80% or 95% demand coverage that when $R = 5$, the number of couriers required is often either the same or just one more than the fleet required in the selective-service $R = \infty$ case.

Lastly, we observe that S and the distribution of τ_j lead to large variations in the fleet size required to achieve a certain demand coverage fraction. In particular, decreasing S from 60 to 45 minutes may result in an average increase in the fleet size ranging between 30% and 60% to maintain a fixed level of service, while changes in the delivery location distribution from $Tria(1, 1, 45)$ to $Unif(1, 45)$ may require even doubling the courier fleet. Interestingly, we observe that in order to increase the average coverage from 95% to 100%, it might be necessary an increase of the fleet size ranging between 20% to nearly 100%.

Average order coverage		$\tau_j \sim Unif(1,45)$				$\tau_j \sim Tria(1,1,45)$			
		$n=75$		$n=150$		$n=75$		$n=150$	
		$S=45$	$S=60$	$S=45$	$S=60$	$S=45$	$S=60$	$S=45$	$S=60$
80%	$R=1$	7	5	9	6	4	3	5	3
	$R=2$	6	4	8	6	3	3	4	3
	$R=5$	5	4	8	5	3	3	4	3
	$R=\infty$	5	3	6	4	3	2	3	3
95%	$R=1$	10	7	14	8	5	4	7	5
	$R=2$	9	6	13	8	5	4	7	5
	$R=5$	9	6	13	8	5	4	7	5
	$R=\infty$	8	5	11	7	4	3	6	4
100%	$R=1$	16	9	21	11	8	5	10	7
	$R=2$	16	9	21	11	8	5	10	7
	$R=5$	16	9	21	11	8	5	10	7
	$R=\infty$	16	9	21	11	8	5	10	7

Table 9: Minimum fleet size m needed to serve 80%, 95%, and 100% of all orders on average, for a single depot

4.3.2 Two-depot case

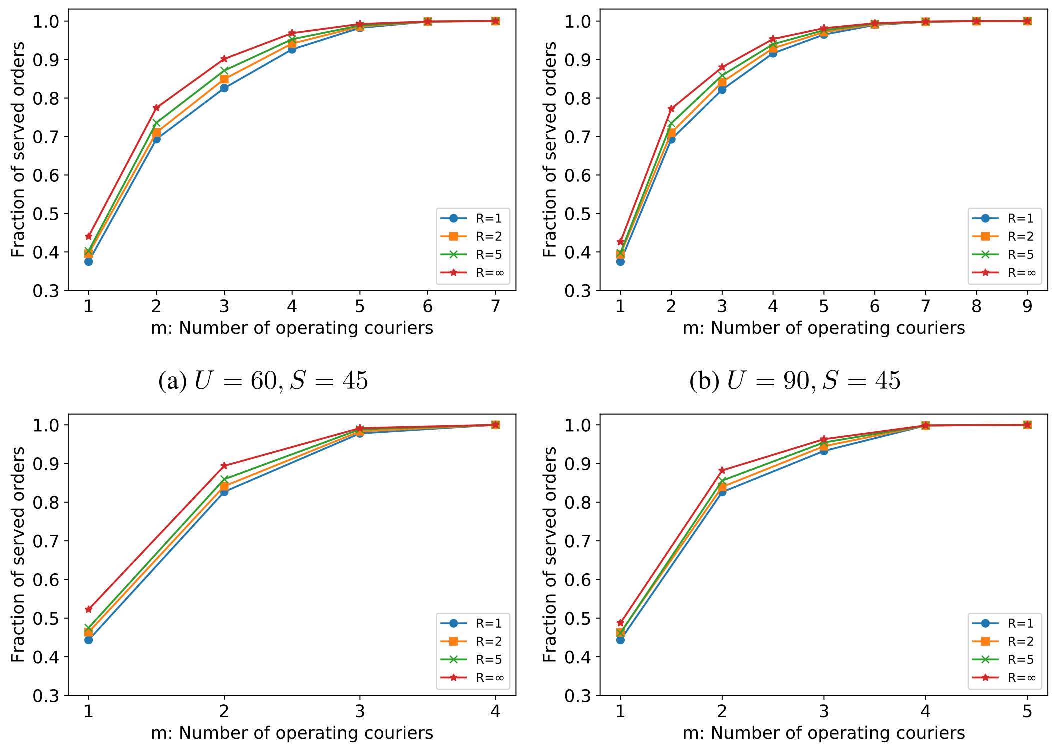
When solving instances with two depots, we found that our proposed formulation has difficulties solving a large number of replications in reasonable times for relatively larger values of n, thus we limit the results in this section to order volumes of $n \in \{75, 150\}$.

For the tested instances with two depots, the results obtained in terms of the maximum fraction of served orders are similar to the ones from the single depot setting. The greatest benefit is obtained for medium-sized fleets of couriers, as shown in Figure 10. For almost every tested fleet size m and values of n, S and U, we observe that increasing the number of radii changes R_d from one to five results in narrowing the performance gap to the upper bound by more than 50%; this is illustrated, for example, by the instances with $m \in \{2, 3\}$ when $U = 90$, where flexibility is most limited. For the largest values of m, performance improvement from increasing R_d is no longer possible since almost every order can be served when $R_d = 1$.

For the results on the fleet sizes required to achieve certain minimum order coverage fractions, we observe that in most cases reducing the fleet size is not possible in this scenario. Indeed, for a coverage requirement of 80% of orders, in almost every scenario the number of couriers needed in the base case coincides with the one from the upper bound, as reported in Table 10. In this case, the flexibility provided by sharing couriers between depots is enough to allow either 3 or 2 couriers to cover the required fraction of orders for almost any value of R. When the coverage requirement is increased to 95%, however, the system becomes less flexible, and for a few cases it becomes possible to reduce the required fleet size by slightly increasing R, as shown in Table 10. Like for a single depot, achieving 100% of order coverage may require a considerable increase in the number of couriers with respect to the ones required for 95% coverage, in some cases even doubling the fleet size.

5 Conclusion and future work

In this paper we have introduced several integer programs built from time-expanded networks to represent different operational situations in food delivery logistics. The results obtained when solving these optimization problems



(a) $U = 60, S = 45$

(b) $U = 90, S = 45$

(c) $U = 60, S = 60$

(d) $U = 90, S = 60$

Figure 10: Fraction of served orders with $n = 150$ for two depots

Average order coverage		$U = 60$				$U = 90$			
		$n = 75$		$n = 150$		$n = 75$		$n = 150$	
		$S = 45$	$S = 60$	$S = 45$	$S = 60$	$S = 45$	$S = 60$	$S = 45$	$S = 60$
80%	$R = 1$	3	2	3	2	3	2	3	3
	$R = 2$	3	2	3	2	3	2	3	2
	$R = 5$	3	2	3	2	3	2	3	2
	$R = \infty$	2	2	3	2	2	2	3	2
95%	$R = 1$	4	3	5	3	4	3	5	4
	$R = 2$	4	3	5	3	4	3	5	4
	$R = 5$	4	3	4	3	4	3	5	3
	$R = \infty$	4	3	4	3	4	3	4	3
100%	$R = 1$	7	4	7	4	8	5	9	5
	$R = 2$	7	4	7	4	8	5	9	5
	$R = 5$	7	4	7	4	8	5	9	5
	$R = \infty$	7	4	7	4	8	5	9	5

Table 10: Minimum fleet size m needed to serve 80%, 95%, and 100% of orders on average, for two depots

provide valuable insights on the effect of different parameters on operational performance metrics, namely customer distributions, target delivery time, order volume and fleet size. In particular, this research seeks to answer basic questions about how to optimize the delivery resources in response to different demand patterns and service level requirements and to explore the effectiveness of optimizing the coverage of orders around a depot given a limited delivery capacity as a demand management mechanism. The flexibility of these formulations can easily be adapted to study further trade-offs. We show through computational experiments the interactions between the analyzed metrics for cases with a single and two depots.

In this paper, we present simplified instances and network construction algorithms by assuming special geometries to avoid the complexity of routing in a general network. Using these simplifications, we are able to measure the benefits of fleet sizing and demand management in meal delivery settings as this allows us to optimally solve the considered problems for a wide variety of instances. Despite this, a natural line of further research is to adapt our framework to general networks that are more representative of urban settings. As this case not only considers dispatch but also routing decisions, we anticipate that the complexity of the corresponding time-expanded network will make computation prohibitively expensive. Under such circumstances, exploring the use of refinement algorithms like column generation and branch-and-price may provide reasonable research directions for the perfect information case, and adaptive approaches using different dispatch technologies when information is partially revealed over time.

Other interesting extensions of our problems include (i) the study of fleet sizing and demand management in settings involving multiple depots with a shared set of couriers, to analyze how these features scale when more complex settings are considered; and (ii) the study of product substitution and the benefit of being able to select the depot which an order should be picked up from, possibly increasing the efficiency of the delivery process.

Acknowledgements

Co-author Ramon Auad thanks the National Agency for Research and Development (ANID) for supporting his research with the scholarship program Doctorado Becas Chile 2017 - 72180404.

References

[1] Morgan Stanley Research . Is online food delivery about to get "amazoned?", 2017. Retrieved from `https://www.morganstanley.com/ideas/online-food-delivery-market-expands/`. Accessed November 11, 2021.

[2] Goch R. and Titone J. The economics of quick service restaurant delivery partnerships, 2018. Retrieved from `https://digitalcommons.molloy.edu/cgi/viewcontent.cgi?article=1043&context=bus_fac`. Accessed November 11, 2021.

[3] Yoon E. The grocery industry confronts a new problem: Only 10% of americans love cooking, 2017. Retrieved from `https://hbr.org/2017/09/the-grocery-industry-confronts-a-new-problem-only-10-of-americans-love-cooking`. Accessed November 11, 2021.

[4] Hirschberg C., Rajko A., Schumacher T., and Wrulich M. The changing market for food delivery, 2016. Retrieved from https://www.mckinsey.com/industries/high-tech/our-insights/the-changing-market-for-food-delivery. Accessed November 11, 2021.

[5] Statista Report . Online food delivery, 2019. Retrieved from https://www.statista.com/outlook/374/100/online-food-delivery/worldwide#market-arpu. Accessed November 11, 2021.

[6] Srinivasan S. India's online food aggregators are taking lessons from china, 2018. Retrieved from https://economictimes.indiatimes.com/small-biz/startups/newsbuzz/indias-online-food-aggregators-are-taking-lessons-from-china/articleshow/63740650.cms. Accessed November 11, 2021.

[7] Reyes D., Erera A., Savelsbergh M., Sahasrabudhe S., and O'Neil R. The meal delivery routing problem. *Optimization Online*, 6571, 2018. http://www.optimization-online.org/DB_FILE/2018/04/6571.pdf.

[8] van Lon R. R., Ferrante E., Turgut A. E., Wenseleers T., Berghe G. V., and Holvoet T. Measures of dynamism and urgency in logistics. *European Journal of Operational Research*, 253(3):614–624, 2016. https://doi.org/10.1016/j.ejor.2016.03.021.

[9] Dholakia U. M. Everyone hates uber's surge pricing – here's how to fix it, 2015. Retrieved from https://hbr.org/2015/12/everyone-hates-ubers-surge-pricing-heres-how-to-fix-it. Accessed November 11, 2021.

[10] Taylor T. A. On-demand service platforms. *Manufacturing & Service Operations Management*, 20(4):704–720, 2018. https://doi.org/10.1287/msom.2017.0678.

[11] Yildiz B. and Savelsbergh M. Service and capacity planning in crowd-sourced delivery. *Transportation Research Part C: Emerging Technologies*, 100:177–199, 2019. https://doi.org/10.1016/j.trc.2019.01.021.

[12] Psaraftis H. N., Wen M., and Kontovas C. A. Dynamic vehicle routing problems: Three decades and counting. *Networks*, 67(1):3–31, 2016. https://doi.org/10.1002/net.21628.

[13] Berbeglia G., Cordeau J.-F., and Laporte G. Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202(1):8–15, 2010. https://doi.org/10.1016/j.ejor.2009.04.024.

[14] Agatz N., Erera A., Savelsbergh M., and Wang X. Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research*, 223(2):295–303, 2012. https://doi.org/10.1016/j.ejor.2012.05.028.

[15] Klapp M. A., Erera A. L., and Toriello A. The one-dimensional dynamic dispatch waves problem. *Transportation Science*, 52(2):402–415, 2016. https://doi.org/10.1287/trsc.2016.0682.

[16] Archetti C., Feillet D., and Speranza M. G. Complexity of routing problems with release dates. *European Journal of Operational Research*, 247(3):797–803, 2015. https://doi.org/10.1016/j.ejor.2015.06.057.

[17] Reyes D., Erera A. L., and Savelsbergh M. W. Complexity of routing problems with release dates and deadlines. *European Journal of Operational Research*, 266(1):29–34, 2018. https://doi.org/10.1016/j.ejor.2017.09.020.

[18] Ulmer M., Thomas B., and Mattfeld D. Preemptive depot returns for dynamic same-day delivery. *EURO Journal on Transportation and Logistics*, 8(4):327–361, 2019. https://doi.org/10.1007/s13676-018-0124-0.

[19] Ulmer M. and Savelsbergh M. Workforce scheduling in the era of crowdsourced delivery. *Transportation Science*, 54(4):1113–1133, 2020. https://doi.org/10.1287/trsc.2020.0977.

[20] Ulmer M. W., Thomas B. W., Campbell A. M., and Woyak N. The restaurant meal delivery problem: Dynamic pick-up and delivery with deadlines and random ready times. *Transportation Science*, 2020. https://doi.org/10.1287/trsc.2020.1000.

[21] Yildiz B. and Savelsbergh M. Provably high-quality solutions for the meal delivery routing problem. *Transportation Science*, 53(5):1372–1388, 2019. https://doi.org/10.1287/trsc.2018.0887.

[22] Klapp M. A., Erera A. L., and Toriello A. The dynamic dispatch waves problem for same-day delivery. *European Journal of Operational Research*, 271(2):519–534, 2018. https://doi.org/10.1016/j.ejor.2018.05.032.

[23] Mor A. and Speranza M. Vehicle routing problems over time: a survey. *4OR*, 18(2):129–149, June 2020. https://doi.org/10.1007/s10288-020-00433-2.

[24] Cattaruzza D., Absi N., and Feillet D. The multi-trip vehicle routing problem with time windows and release dates. *Transportation Science*, 50(2):676–693, 2016. https://doi.org/10.1287/trsc.2015.0608.

[25] Shelbourne B. C., Battarra M., and Potts C. N. The vehicle routing problem with release and due dates. *INFORMS Journal on Computing*, 29(4):705–723, 2017. https://doi.org/10.1287/ijoc.2017.0756.

[26] Angelelli E., Grazia Speranza M., and Savelsbergh M. W. Competitive analysis for dynamic multiperiod uncapacitated routing problems. *Networks: An International Journal*, 49(4):308–317, 2007. https://doi.org/10.1002/net.20180.

[27] Angelelli E., Savelsbergh M. W., and Speranza M. G. Competitive analysis of a dispatch policy for a dynamic multi-period routing problem. *Operations Research Letters*, 35(6):713–721, 2007. https://doi.org/10.1016/j.orl.2007.02.006.

[28] Yildiz B. and Savelsbergh M. Pricing for delivery time flexibility. *Transportation Research Part B: Methodological*, 133:230–256, 2020. https://doi.org/10.1016/j.trb.2020.01.004.

[29] Erera A., Hewitt M., Savelsbergh M., and Zhang Y. Improved load plan design through integer programming based local search. *Transportation Science*, 47(3):412–427, 2013. https://doi.org/10.1287/trsc.1120.0441.

[30] Boland N., Hewitt M., Marshall L., and Savelsbergh M. The continuous-time service network design problem. *Operations Research*, 65(5):1303–1321, 2017. `https://doi.org/10.1287/opre.2017.1624`.

[31] Vu D. M., Hewitt M., Boland N., and Savelsbergh M. Solving time dependent traveling salesman problems with time windows. *Optimization Online*, 6640, 2018. `http://www.optimization-online.org/DB_FILE/2018/05/6640.pdf`.

[32] Skutella M. An introduction to network flows over time. In *Cook W., Lovász L., Vygen J. (eds) Research Trends in Combinatorial Optimization*, pages 451–482. Springer, 2009. `https://doi.org/10.1007/978-3-540-76796-1_21`.

Appendices

A Proofs

Proof of Proposition 1

If $|J| = 1$ the claim trivially follows and therefore we assume $|J| \geq 2$. Initially, let $J = J_1 \cup J_2$, $J_1, J_2 \neq \emptyset$ and $J_1 \cap J_2 = \emptyset$. Without loss of generality we can assume that $\tau_{J_1} \geq \tau_{J_2}$. Consider then a feasible schedule S_1 to Problem 1 that at time t dispatches two couriers c_1 and c_2 with order sets J_1 and J_2, respectively. Note that c_1 is unavailable for picking up other ready orders during time points $I_1 = \{t, t+1, \ldots, t+2\tau_{J_1}\}$, whereas c_2 will be unavailable to serve any new orders from t to $t + 2\tau_{J_2}$.

Alternatively, consider a schedule S_2 that bundles J into a single dispatch for courier c_1 at time t (which is possible since couriers do not have a fixed capacity). Since by definition $\tau_J = \tau_{J_1}$, dispatching c_1 also serves all $|J|$ orders during I_1 while courier c_2 remains at the depot beginning at time t, which is earlier than the return time $t + 2\tau_{J_2}$ in the above schedule. Thus, schedule S_2 dominates S_1. $\square$

Proof of Lemma 2

Let $t' \in (t, r_{j+1})$, and consider schedule $S(t')$ that dispatches a courier at time t' with orders set $A(t')$. From $t \in [r_j, t')$ it follows that $A(t') \subseteq A(t)$: indeed, no new orders are placed in $(t, t']$ although some of the orders in $A(t)$ might not be active by t'. Hence, schedule $S(t')$ can always be improved by the one that moves the dispatch at t' to t, which is always possible by definition of t. $\square$

Proof of Proposition 3

For $j \in N$, the possible times at which a courier might become available at the depot during $[r_j, r_{j+1})$ are r_j and any returning time $r_i + 2\sum_{k \in K} \tau_k \in (r_j, r_{j+1})$ with $i < j$ and $K \subseteq N$ that results from a dispatch previous to r_j, thus Lemma 2 implies that an optimal schedule can be obtained by considering only such dispatch times.

Moreover, denoting the set of all dispatches of interest as $\mathcal{T}_0$, the only time points outside the depot to be considered are the potential delivery times $\{t + \tau_i : t \in \mathcal{T}_0, i \in A(t)\}$, at each of which a dispatched courier decides between traversing to a further delivery location or returning to the depot. $\square$

Proof of Proposition 4

For a fixed binary vector $\overline{v}$ and $m \in \mathbb{Z}_+$, a feasible value of z must satisfy the constraints:

$$\overline{v}_{jp} \leq \sum_{q \in \alpha_p^-} z_{qp}, \quad \forall j \in N, \quad \forall p \in \mathcal{V}_j \tag{10a}$$

$$\sum_{q \in \alpha_\alpha^+} z_{\alpha q} = m \tag{10b}$$

$$\sum_{p \in \alpha_\omega^-} z_{p\omega} = m \tag{10c}$$

$$\sum_{p \in \alpha_q^-} z_{pq} = \sum_{r \in \alpha_q^+} z_{qr}, \quad \forall q \in \mathcal{V} \setminus \{\alpha, \omega\} \tag{10d}$$

$$z_{pq} \in \begin{cases} \mathbb{R}_+ & \text{if } p, q \in \mathcal{V}_0, \\ \{0,1\} & \text{otherwise} \end{cases} \quad \forall (p,q) \in \mathcal{A} \tag{10e}$$

By construction of the underlying time-expanded network, each non-depot node p has a unique arc $a_p \in \mathcal{A}$ inbound to p, thus the right hand side of (10a) can be written as $\sum_{q \in \alpha^-(p)} z_{qp} = z_{a_p}$. Consequently, Constraints (10a) and (10e) give lower and upper bounds on the flow of each arc in the network: the flow of arcs (q,p) with p being a non-depot node is bounded by $[\overline{v}_{jp}, 1]$; for the remaining arcs, the capacity of the ones whose tail is a non-depot node is 1; lastly, arcs between two depot nodes have infinite capacity. Furthermore, Constraints (10b) - (10d) correspond to flow conservation equations at every node of the network. Therefore, for variables z the above constraints a network flow polyhedron with integer coefficients, and hence the optimal z is integral. $\square$

Proof of Proposition 5

As Problem 5 is by definition the setting of Problem 1 with service radius management, it suffices to show that Constraint set (4) accurately models the service radius mechanics described in the formulation of Problem 5 and allows to compute the optimal service radii. Consider Algorithm 1, which partitions the set of orders N into the

Algorithm 1 (R_PARTITION_SORT)

Input: $N, \{(r_j, \tau_j)\}_{j \in N}, \{t_\ell\}_{\ell=1}^R$
Output: Lists of orders $B_1, \ldots, B_R$, each sorted in ascending order of τ_j.

```
1: $B_\ell \leftarrow \emptyset, \forall \ell = 1, \ldots, R$
2: $j \leftarrow 1$
3: for $\ell \in \{1, \ldots, R\}$ do
4:     while $r_j < t_{\ell+1}$ do
5:         $B_\ell \leftarrow B_\ell \cup \{j\}$
6:         $j \leftarrow j + 1$
7:     Sort elements of $B_\ell$ in ascending order of $\tau_j$
   return $\{B_\ell\}_{\ell=1}^R$
```

R sorted lists $\{B_\ell\}_{\ell=1}^R$. Note that Constraint set (4) enforces that for each $\ell = 1, \ldots, R$, whenever order $B_{\ell,i}$ is served, so are all the orders $B_{\ell,j}, \forall j < i$, which is the definition of service radius. Moreover, this formulation allows us to compute the optimal service radius for each radius shift without using explicit decision variables for the service radii: let (v^*, z^*) be the optimal solution to Problem 5, and for each $\ell \in \{1, \ldots, R\}$, let $\mu_\ell = \max\{i :$

$\sum_p v^*_{B_{\ell,i},p} = 1\}$, then by construction of B_ℓ each optimal radius is calculated as $\rho_\ell = \tau_{B_{\ell,\mu_\ell}}$. □

Proof of Proposition 6

Running Algorithm 1 for each depot $d \in \{1,2\}$ and replacing τ_j by $\tilde{\tau}^d_j, \forall j \in N_d$ constructs the sorted lists $\{B^d_\ell\}_{\ell=1}^{R_d}$. Then the claim follows from applying the same argument for Proposition 5 to each depot. □

B Construction of the time-expanded network for the single depot setting

Algorithm 2 specifies how to produce the complete time-expanded network for these optimization problems. The algorithm starts by defining in lines 1 - 3, depot nodes with order ready times r_j, and initializes the set of dispatch times $\mathcal{T}_0$ with ready times $\{r_j\}_{j\in N}$. Then for each dispatch time $t \in \mathcal{T}_0$, the algorithm first finds the the ready time r_{j*} that immediately succeeds t and creates a wait arc from depot node $(t,0)$ to $(r_{j*},0)$ (lines 5 and 6). If there exists active orders at t, then line 7 and 8 sort them in non-decreasing order in terms of the travel time from the depot, and then lines 9 - 12 iterate over the sorted set of active orders, sequentially creating the non-depot node at the time and location at which each of the orders can be delivered with an inbound arc, as well as a return node that originates from that delivery with the corresponding returning arc; if such return node was not previously defined in the network, then the corresponding depot time is added to the set $\mathcal{T}_0$, as this constitutes a potentially new dispatch time. Once the algorithm stops discovering new returning times, it returns the network $\mathcal{N} = (\mathcal{V},\mathcal{A})$.

Algorithm 2 (CREATE_NETWORK)

```
Input: $N, (r_j, \tau_j, \Delta_j)_{j\in N}, T$
Output: Directed network $\mathcal{N} = (\mathcal{V},\mathcal{A})$
 1: $\mathcal{V} \leftarrow \{(r_j,0)\}_{j\in N}$
 2: $\mathcal{A} \leftarrow \emptyset$
 3: $\mathcal{T}_0 \leftarrow \{r_j\}_{j\in N}$
 4: for $t \in \mathcal{T}_0$ do
 5:     Find lowest index $j^* \in N \cup \{n+1\}$ s.t. $t < r_{j^*}$        $\triangleright\ r_{n+1} \equiv T$
 6:     $\mathcal{A} \leftarrow \mathcal{A} \cup \{((t,0),(r_{j^*},0))\}$
 7:     Compute set of active orders $A(t) = \{j \in N : t \geq r_j, t+\tau_j \leq \Delta_j\}$
 8:     Sort $\{\tau_j\}_{j\in A(t)}$ in non-decreasing order, into $\{\tau_{(i)}\}_{i=1}^{|A(t)|}$
 9:     for $i = 1,\ldots,|A(t)|$ do
10:         $\mathcal{V} \leftarrow \mathcal{V} \cup \{(t+\tau_{(i)},\tau_{(i)}),(t+2\tau_{(i)},0)\}$
11:         $\mathcal{A} \leftarrow \mathcal{A} \cup \{((t+\tau_{(i-1)},\tau_{(i-1)}),(t+\tau_{(i)},\tau_{(i)})),((t+\tau_{(i)},\tau_{(i)}),(t+2\tau_{(i)},0))\}$        $\triangleright\ \tau_{(0)} \equiv 0$
12:         $\mathcal{T}_0 \leftarrow \mathcal{T}_0 \cup \{t+2\tau_{(i)}\}$
    return $\mathcal{N} = (\mathcal{V},\mathcal{A})$
```

A numerical example

For an example of the output of Algorithm 1, consider an instance with $T = 6$, $S = 3$, and whose set of orders to be served $N = \{1,2\}$ is characterized by Table 11. The resulting partial time-expanded network is illustrated in Figure 11. In the illustration, nodes that are filled and in the horizontal axis correspond to depot nodes; otherwise they are non-depot nodes, representing when and where orders can be delivered. Arcs inbound to a non-depot node that emerge from a depot node correspond to a dispatch; and arcs inbound to a depot node from a non-depot node

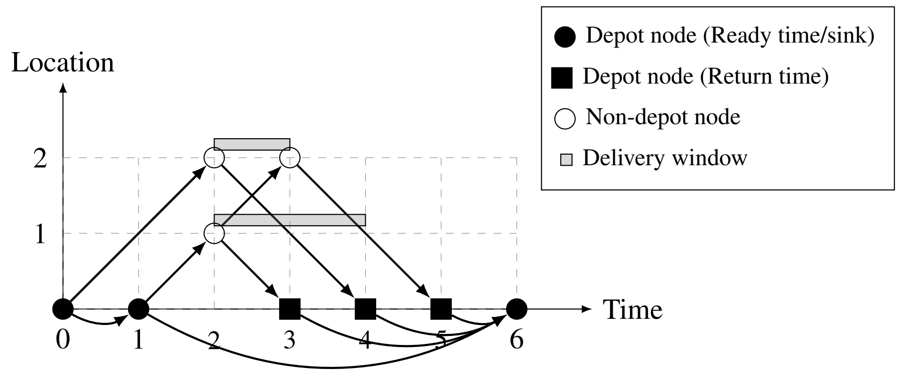


Figure 11: Example of a time-expanded network for an instance with two orders

represent a return. Arcs between depot nodes represent the action of a courier waiting at the depot; arcs between non-depot nodes represent the action of traveling between order destinations. Note that a dispatch from the depot at $t = 3$ to order 2 is not needed since the return node arrived only from already delivering order 2. In this example, a single courier is able to serve both orders when dispatched at time $t = 1$.

j	r_j	τ_j	Δ_j
1	0	2	3
2	1	1	4

Table 11: Characterization of orders of numerical example

C Construction of the time-expanded network for the *L*-star setting

For line segment $h \in \{1, 2, \ldots, L\}$, let $\mathcal{T}_0^h$ be the set of time points of possible dispatches to line segment h. Algorithm 3 constructs the time-expanded network for the L-star setting. This algorithm firstly creates depot nodes at ready times of every order in the system for all line segments, and then for each line segment h, it initializes the set $\mathcal{T}_0^h$ with the ready times of orders to be delivered along h (lines 1 - 5).

Subsequently in lines 6 and 7, Algorithm 3 creates arcs and non-depot nodes for dispatches at ready times, and arcs and depot nodes for the corresponding return times, by executing Algorithm 4 once for every line segment. For a given line segment h, this subroutine works similar to Algorithm 2 for the single line segment setting, although this extension also defines new dispatches from a return node to every line segments. This is done with the help of auxiliary sets $\mathcal{S}_0^{h'}$, which keep track of new dispatch times not yet in $\mathcal{T}_0^{h'}$.

Algorithm 3 performs a final iterative step in lines 8 - 14 if new dispatches are yet to be evaluated for some line segments, i.e. if $S_{temp} \neq \emptyset$. For a line segment $h \in S_{temp}$, Algorithm 4 is executed to evaluate and define new dispatch times in the set $\mathcal{S}_0^h$, and to define the corresponding nodes and arcs. Note that this in turn may generate new dispatch times for some other line segment h' due to new return times, in which case these are appended to

$\mathcal{S}_0^{h'}$ and h' is included in S_{temp}. Once the new dispatches are defined, the new dispatch times are appended to the defined dispatch times $\mathcal{T}_0^h$, S_{temp} is computed again. This process repeats until no new dispatch times are left to be evaluated for any line segment, i.e. when $S_{temp} = \emptyset$. At this point, Algorithm 3 returns the network $\mathcal{N} = (\mathcal{V}, \mathcal{A})$.

Algorithm 3 L_STAR_NETWORK_CREATION

```
Input: $L, \{N_h, \{(r_j, \tau_j, \Delta_j)\}_{j\in N_h}\}_{h\in\{1,2,\dots,L\}}, T$
Output: Directed network $\mathcal{N} = (\mathcal{V}, \mathcal{A})$
 1: $\mathcal{V} \leftarrow \{(r_j, 0, 0)\}_{j\in N} \cup \{(T, 0, 0)\}$
 2: $\mathcal{A} \leftarrow \emptyset$
 3: for $h \in \{1, 2, \dots, L\}$ do
 4:     $\mathcal{T}_0^h \leftarrow \{r_j\}_{j\in N_h}$
 5:     $\mathcal{S}_0^h \leftarrow \emptyset$
 6: for $h \in \{1, 2, \dots, L\}$ do
 7:     $(\mathcal{V}, \mathcal{A}, \{\mathcal{S}_0^{h'}\}_{h'=1}^L) =$ L_STAR_ROUTES$(\mathcal{V}, \mathcal{A}, h, \mathcal{T}_0^h, \{\mathcal{T}_0^{h'}\}_{h'=1}^L, \{\mathcal{S}_0^{h'}\}_{h'=1}^L)$
 8: $S_{temp} \leftarrow \{h \in \{1, 2, \dots, L\} : \mathcal{S}_0^h \neq \emptyset\}$
 9: while $S_{temp} \neq \emptyset$ do
10:     Let $h$ be one of the elements in $S_{temp}$
11:     $(\mathcal{V}, \mathcal{A}, \{\mathcal{S}_0^{h'}\}_{h'=1}^L) =$ L_STAR_ROUTES$(\mathcal{V}, \mathcal{A}, h, \mathcal{S}_0^h, \{\mathcal{T}_0^{h'}\}_{h'=1}^L, \{\mathcal{S}_0^{h'}\}_{h'=1}^L)$
12:     $\mathcal{T}_0^h \leftarrow \mathcal{T}_0^h \cup \mathcal{S}_0^h$
13:     $\mathcal{S}_0^h \leftarrow \emptyset$
14:     $S_{temp} \leftarrow \{h' \in \{1, 2, \dots, L\} : \mathcal{S}_0^{h'} \neq \emptyset\}$
    return $\mathcal{N} = (\mathcal{V}, \mathcal{A})$
```

Algorithm 4 L_STAR_ROUTES$(\mathcal{V}, \mathcal{A}, h, \mathcal{T}, \{\mathcal{T}_0^{h'}\}_{h'=1}^L, \{\mathcal{S}_0^{h'}\}_{h'=1}^L)$

```
Input: Node set $\mathcal{V}$, arc set $\mathcal{A}$, line segment $h$, set of time point at the depot $\mathcal{T}$, sets of existing depot time points
    $\{\mathcal{T}_0^{h'}\}_{h'=1}^L$, sets of new returning time point at all line segments $\{\mathcal{S}_0^{h'}\}_{h'=1}^L$
Output: Updated sets $\mathcal{V}, \mathcal{A}, \{\mathcal{S}_0^{h'}\}_{h'=1}^L$
 1: for $t \in \mathcal{T}$ do
 2:     Find lowest $j^* \in N \cup \{n+1\}$ s.t. $t < r_{j^*}$                              ▷ $r_{n+1} \equiv T$
 3:     $\mathcal{A} \leftarrow \mathcal{A} \cup \{((t, 0, 0), (r_{j^*}, 0, 0))\}$
 4:     Compute set of active orders $A_h(t) \subseteq N_h$
 5:     Sort $\{\tau_j\}_{j\in A_h(t)}$ in ascending order, into $\{\tau_{(i)}\}_{i=1}^{|A_h(t)|}$
 6:     for $i \in \{1, 2, \dots, |A_h(t)|\}$ do
 7:         $\mathcal{V} \leftarrow \mathcal{V} \cup \{(t + \tau_{(i)}, \tau_{(i)}, h)\}$
 8:         $\mathcal{A} \leftarrow \mathcal{A} \cup \{((t + \tau_{(i-1)}, \tau_{(i-1)}, \mathbf{1}_{[i\neq 1]} \times h), (t + \tau_{(i)}, \tau_{(i)}, h))\}$      ▷ $\tau_{(0)} \equiv 0$
 9:         for $h' \in \{1, 2, \dots, L\}$ do
10:             if $t + 2\tau_{(i)} \notin \mathcal{T}_0^{h'}$ then
11:                 $\mathcal{S}_0^{h'} \leftarrow \mathcal{S}_0^{h'} \cup \{t + 2\tau_{(i)}\}$
12:         $\mathcal{V} \leftarrow \mathcal{V} \cup \{(t + 2\tau_{(i)}, 0, 0)\}$
13:         $\mathcal{A} \leftarrow \mathcal{A} \cup \{((t + \tau_{(i)}, \tau_{(i)}, h), (t + 2\tau_{(i)}, 0, 0))\}$
    return $(\mathcal{V}, \mathcal{A}, \{\mathcal{S}_0^{h'}\}_{h'=1}^L)$
```

D Construction of the time-expanded network for the two-depot setting

The network construction procedure is presented in Algorithm 5. Here, the location of depot d in the x-axis is denoted as ξ_d, with $\xi_1 = 0$ and $\xi_2 = U$. In lines 1 - 4, the constructor first defines depot nodes with order ready times r_j^d at each depot d, and arcs $((0,0,1),(r_1^2,U,2))$ and $((T,U,2),(T,0,1))$ to ensure that the resulting network has a unique source node $(0,0,1)$ and a unique sink node $(T,0,1)$. Moreover, it also initializes the set $\mathcal{T}_{\xi_d}$ of depot times with ready times $\{r_j^d\}_{j=1}^{n_d}$.

Then in lines 5 - 8, Algorithm 5 runs Algorithm 6 once for each depot $d \in \{1,2\}$ to define the non-depot nodes where orders in N_d may be served, and additional depot nodes that correspond to courier returning times to d (lines 2 - 12 of Algorithm 6). Note that in this setting, each dispatch from d allows to travel beyond the furthest delivery location among the dispatched orders to get to the other depot $\overline{d}$, and so Algorithm 6 creates an extra arc $((t+\tilde{\tau}^d_{(|A_d(t)|)}, \tau^d_{(|A_d(t)|)}, d),(t+U,\xi_{\overline{d}},\overline{d}))$ from the furthest non-depot node of a dispatch to the corresponding arrival node at $\overline{d}$. The new arrival node to $\overline{d}$ is in turn a potential dispatch from that depot, and so the corresponding arrival time is stored in the set $\mathcal{T}^{potential}_{\xi_{\overline{d}}}$ so a dispatch can be evaluated later on (lines 13 - 15 of Algorithm 6).

Lastly, Algorithm 5 performs a final step in lines 9 - 14 that iteratively runs Algorithm 6 to evaluate new potential dispatch from each depot d at times in $\mathcal{T}^{potential}_{\xi_d}$ (similar to Algorithm 3 for the L-star setting). These dispatches may produce new arrival times to depot $\overline{d}$ due to crosses between depots, each of them in turn potentially defining a new dispatch time from $\overline{d}$. In such case, the new dispatch time is added to the set $\mathcal{T}^{potential}_{\xi_{\overline{d}}}$. Once the dispatches nodes and arcs are defined for a depot d, the new dispatch times are appended to the set of defined dispatch times $\mathcal{T}_{\xi_d}$. The iterative process repeats until no new dispatch times are discovered for either depot, namely, $\mathcal{T}^{potential}_{\xi_d} \subseteq \mathcal{T}_{\xi_d}$ for some $d \in \{1,2\}$.

Algorithm 5 (2_DEPOT_CREATE_NETWORK)

```
Input: $\{N_d, \boldsymbol{r}^d, \boldsymbol{\tau}^d, \boldsymbol{\Delta}^d\}_{d\in\{1,2\}}, T, U$
Output: Directed network $\mathcal{N} = (\mathcal{V}, \mathcal{A})$
1: $\mathcal{V} \leftarrow \bigcup_{d=1}^{2}\{(r_j^d, \xi_d, d)\}_{j\in N_d} \cup \{(T,0,1),(T,U,2)\}$
2: $\mathcal{A} \leftarrow \{((0,0,1),(r_1^2,U,2)),((T,U,2),(T,0,1))\}$
3: $\mathcal{T}_0 \leftarrow \{r_j^1\}_{j\in N_1}$
4: $\mathcal{T}_U \leftarrow \{r_j^2\}_{j\in N_2}$
5: $(\mathcal{V}, \mathcal{A}, \mathcal{T}_0, \mathcal{T}_U^{potential}) \leftarrow$ 2_DEPOT_ROUTES$(\mathcal{V}, \mathcal{A}, 1, \mathcal{T}_0)$
6: $\mathcal{T}_U \leftarrow \mathcal{T}_U \cup \mathcal{T}_U^{potential}$
7: $\mathcal{T}_U^{potential} \leftarrow \emptyset$
8: $(\mathcal{V}, \mathcal{A}, \mathcal{T}_U, \mathcal{T}_0^{potential}) \leftarrow$ 2_DEPOT_ROUTES$(\mathcal{V}, \mathcal{A}, 2, \mathcal{T}_U)$
9: while True do
10:     for $d \in \{1,2\}$ do
11:         if $\mathcal{T}^{potential}_{\xi_d} \subseteq \mathcal{T}_{\xi_d}$ then return $\mathcal{N} = (\mathcal{V}, \mathcal{A})$
12:         $(\mathcal{V}, \mathcal{A}, \mathcal{T}^{potential}_{\xi_d}, \mathcal{T}^{potential}_{\xi_{\overline{d}}}) \leftarrow$ 2_DEPOT_ROUTES$(\mathcal{V}, \mathcal{A}, 1, \mathcal{T}^{potential}_{\xi_d} \setminus \mathcal{T}_{\xi_d})$
13:         $\mathcal{T}_{\xi_d} \leftarrow \mathcal{T}_{\xi_d} \cup \mathcal{T}^{potential}_{\xi_d}$
14:         $\mathcal{T}^{potential}_{\xi_d} \leftarrow \emptyset$
```

Algorithm 6 2_DEPOT_ROUTES$(\mathcal{V}, \mathcal{A}, d, \mathcal{T}_{\xi_d}^{new_dispatches})$

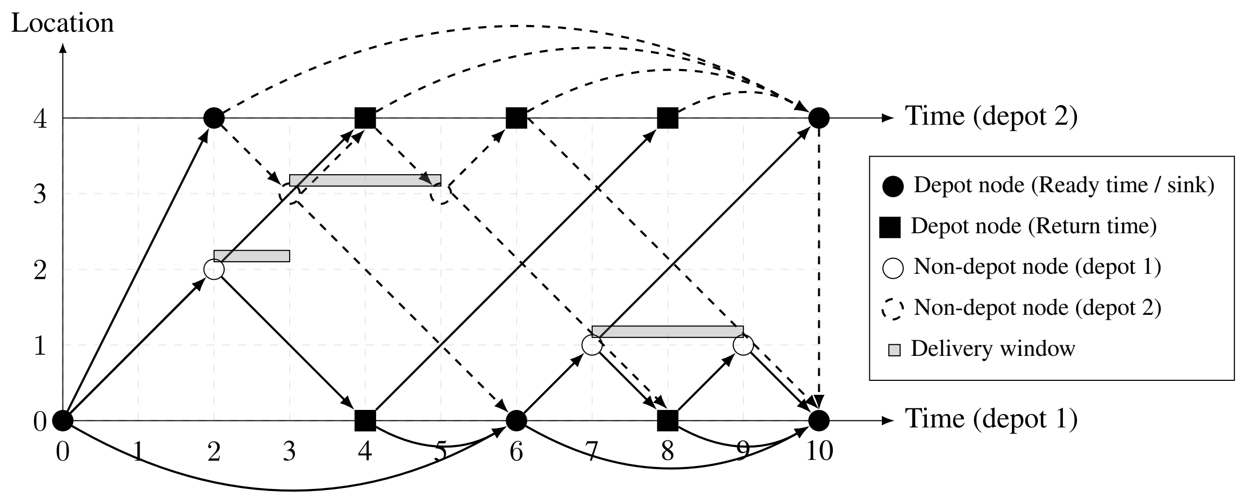
```
Input: Node set $\mathcal{V}$, arc set $\mathcal{A}$, depot $d$, time point set $\mathcal{T}_{\xi_d}^{new\_dispatches}$
Output: Updated sets $\mathcal{V}$,$\mathcal{A}$,$\mathcal{T}_{\xi_d}^{new\_dispatches}$, and set of time points $\mathcal{T}_{\xi_{\overline{d}}}^{potential}$
 1: $\mathcal{T}_{\xi_{\overline{d}}}^{potential} \leftarrow \emptyset$
 2: for $t \in \mathcal{T}_{\xi_d}^{new\_dispatches}$ do
 3:     Find lowest $j^* \in \{1, 2, \ldots, n_d + 1\}$ s.t. $t < r_{j^*}^d$        ▷ $r_{n_d+1}^d \equiv T$
 4:     $\mathcal{A} \leftarrow \mathcal{A} \cup \{((t, \xi_d, d), (r_{j^*}^d, \xi_d, d))\}$
 5:     Compute set of active orders $A_d(t) \subseteq N_d$
 6:     Sort $\{\tau_j^d\}_{j \in A_d(t)}$ in ascending or descending order if $d = 1$ or $d = 2$, respectively (if $\{\tau_{(i)}^d\}_{i=1}^{|A_d(t)|}$ is the
        corresponding sorted sequence, then $\tau_{(i)}^1 \leq \tau_{(i+1)}^1$ and $\tau_{(i)}^2 \geq \tau_{(i+1)}^2, \forall i \in \{1, \ldots, |A_d(t)| - 1\}$)
 7:     for $i \in \{1, 2, \ldots, |A_d(t)|\}$ do
 8:         $\mathcal{V} \leftarrow \mathcal{V} \cup \{(t + \tilde{\tau}_{(i)}^d, \tau_{(i)}^d, d)\}$
 9:         $\mathcal{A} \leftarrow \mathcal{A} \cup \{((t + \tilde{\tau}_{(i-1)}^d, \tau_{(i-1)}^d, d), (t + \tilde{\tau}_{(i)}^d, \tau_{(i)}^d, d))\}$        ▷ $\tau_{(0)}^d \equiv \xi_d$
10:         $\mathcal{T}_{\xi_d}^{new\_dispatches} \leftarrow \mathcal{T}_{\xi_d}^{new\_dispatches} \cup \{t + 2\tilde{\tau}_{(i)}^d\}$
11:         $\mathcal{V} \leftarrow \mathcal{V} \cup \{(t + 2\tilde{\tau}_{(i)}^d, \xi_d, d)\}$
12:         $\mathcal{A} \leftarrow \mathcal{A} \cup \{((t + \tilde{\tau}_{(i)}^d, \tau_{(i)}^d, d), (t + 2\tilde{\tau}_{(i)}^d, \xi_d, d)\}$
13:     $\mathcal{V} \leftarrow \mathcal{V} \cup \{(t + U, \xi_{\overline{d}}, \overline{d})\}$
14:     $\mathcal{T}_{\xi_{\overline{d}}}^{potential} \leftarrow \mathcal{T}_{\xi_{\overline{d}}}^{potential} \cup \{(t + U)\}$
15:     $\mathcal{A} \leftarrow \mathcal{A} \cup \{((t + \tilde{\tau}_{(|A_d(t)|)}^d, \tau_{(|A_d(t)|)}^d, d), (t + U, \xi_{\overline{d}}, \overline{d}))\}$
    return $(\mathcal{V}, \mathcal{A}, \mathcal{T}_{\xi_d}^{new\_dispatches}, \mathcal{T}_{\xi_{\overline{d}}}^{potential})$
```

A numerical example

Figure 12 illustrates a network example with two depots at locations 0 and $U = 4$, with parameter values of $S = 3$, $T = 10$, and orders information as in Table 12. Depot nodes at location 0 and non-depot nodes with solid edge lines correspond to nodes associated to depot 1, whereas depot nodes at location 4 and non-depot nodes with dashed edge lines correspond to depot 2. The origin corresponds to node $(0, 0, 1)$. A flow of courier m is injected to the network through the origin node and arc $((0, 0, 1), (2, 4, 2))$ allows to selectively allocate couriers to begin their shift at either depot. From there couriers can traverse from one depot to the other when being dispatched to serve orders, and by the end they finish the shift at either $(10, 4, 2)$ or $(10, 0, 1)$; in any case, the insertion of arc $((10, 4, 2), (10, 0, 1))$ allows to consider a single sink. In this particular example, a single courier is sufficient to achieve full service.

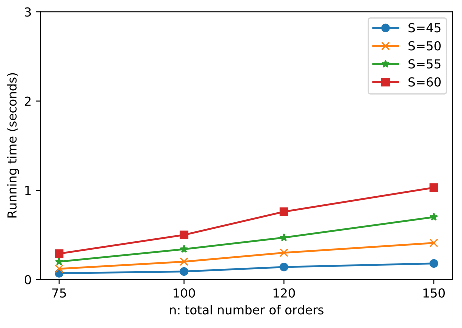
d	j	r_j^d	τ_j^d	$\tilde{\tau}_j^d$	Δ_j^d
1	1	0	2	2	3
1	2	6	1	1	9
2	1	2	3	1	5

Table 12: Characterization of orders of numerical example

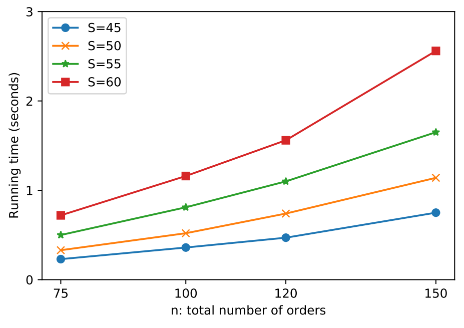


Location
4
3
2
1
0
0
1
2
3
4
5
6
7
8
9
10
Time (depot 2)
Time (depot 1)
● Depot node (Ready time / sink)
■ Depot node (Return time)
○ Non-depot node (depot 1)
◌ Non-depot node (depot 2)
▢ Delivery window

Figure 12: Example of a time-expanded network with 2 depots and 3 orders. Solid arcs are associated to depot 1, and dashed arcs to depot 2



(a) $\tau_j \sim Unif(1, 45)$



(b) $\tau_j \sim Tria(1, 1, 45)$

Figure 13: Average running time for the fleet size minimization problem, single depot

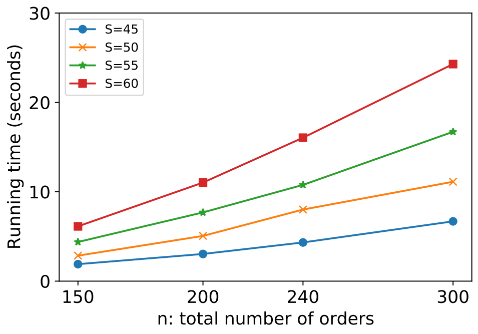
E Running times

Minimum fleet size

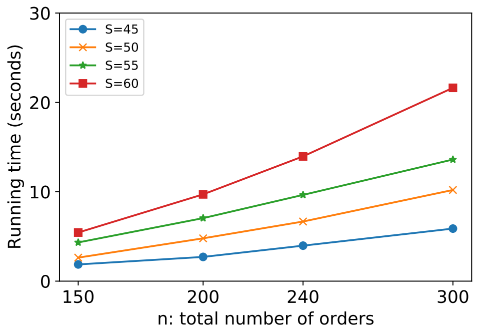
In general, our framework is able to find the minimum fleet size in a few seconds. There is a slight increase in running times as the number of orders and the maximum acceptable service time increase since this translates into larger instances and more options of serving orders. We also note that the two-depot instances require significantly longer solving times than the single-depot setting.

Analysis of lateness via target delivery time

As for fleet minimization, our methods are able to quickly solve all the instances of lateness minimization. Intuitively, a longer acceptable service time translates into more choices of when to serve orders, and hence this causes an increase in the running times. Again, we see an extra computational burden when solving instances with two depots.

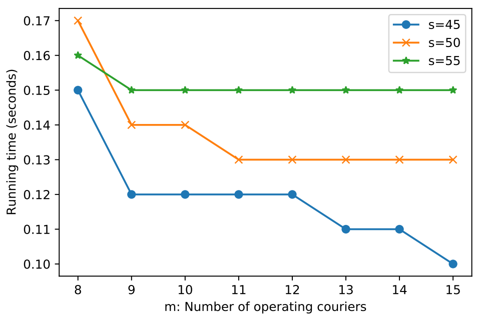


(a) $U = 60$

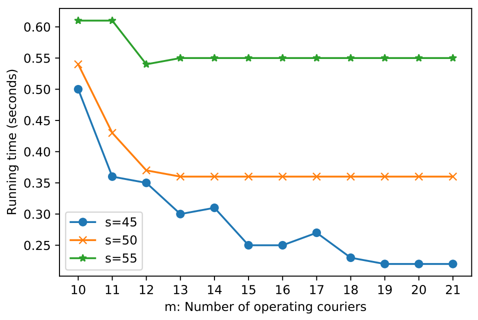


(b) $U = 90$

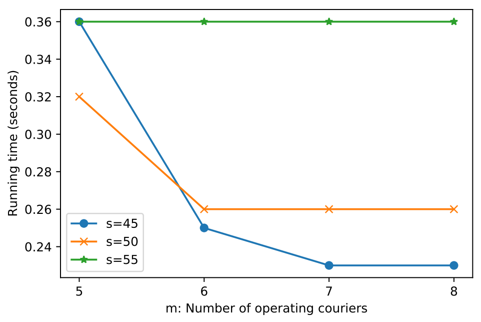
Figure 14: Average running time for the fleet size minimization problem, two depots



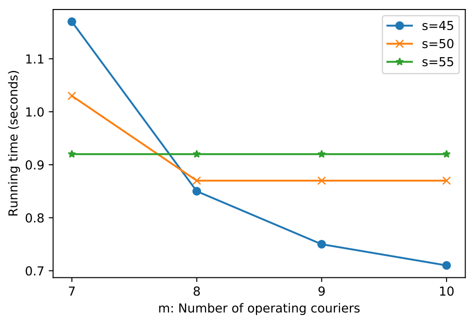
(a) $n = 75, \tau_j \sim Unif(1, 45)$



(b) $n = 150, \tau_j \sim Unif(1, 45)$

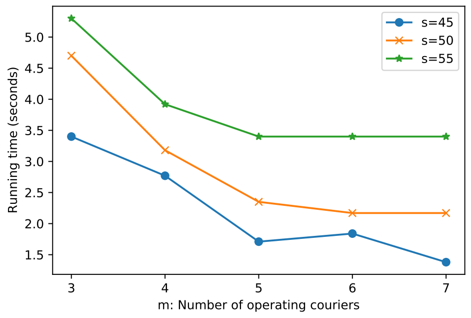


(c) $n = 75, \tau_j \sim Tria(1, 1, 45)$

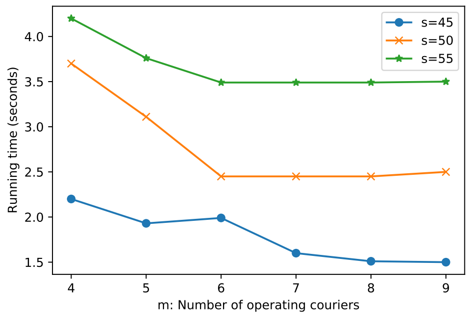


(d) $n = 150, \tau_j \sim Tria(1, 1, 45)$

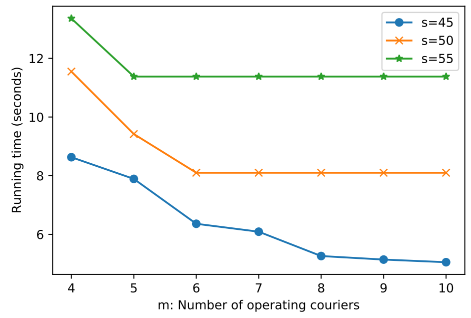
Figure 15: Average running time for the late orders minimization problem, single depot



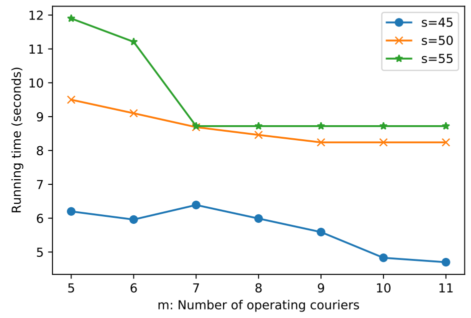
(a) $n = 150, U = 60$



(b) $n = 150, U = 90$



(c) $n = 300, U = 60$

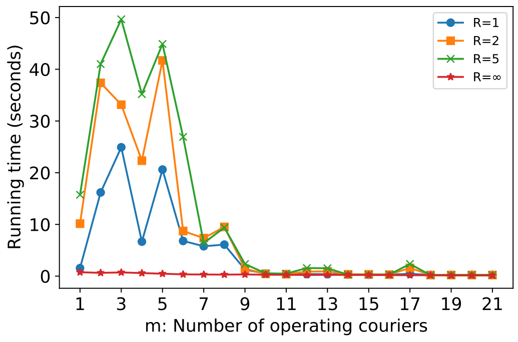


(d) $n = 300, U = 90$

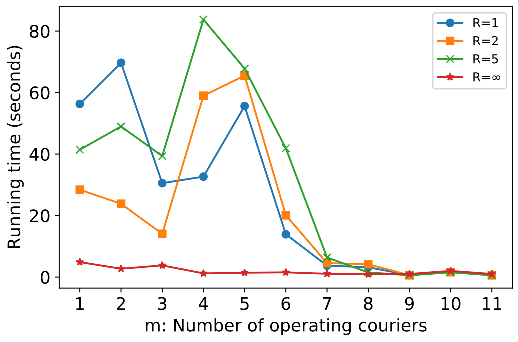
Figure 16: Average running time for the late orders minimization problem, two depots

Demand management via service radius adjustments

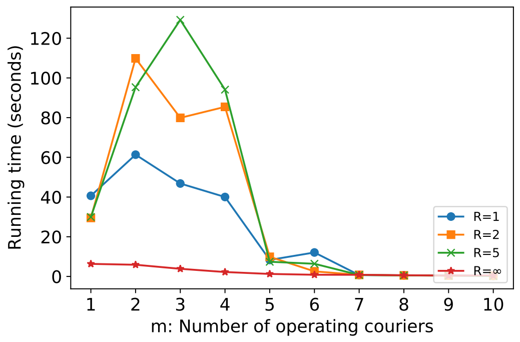
The running times required to solve the service radius management problem are considerably longer than the previous problems. Interestingly, solving the case $R = \infty$ for a single depot is substantially faster than all the other cases; however, for the two-depot setting the case $m = 1$ for $R = \infty$ results in notoriously longer times to optimally solve the instances, possibly because the solver must decide which depot to initially allocate the single courier to. However, despite these large reported times for $m = 1$, it is important to note that all the instances are able to find the optimal solution in under a minute, and the solver spends the rest of the time proving optimality.



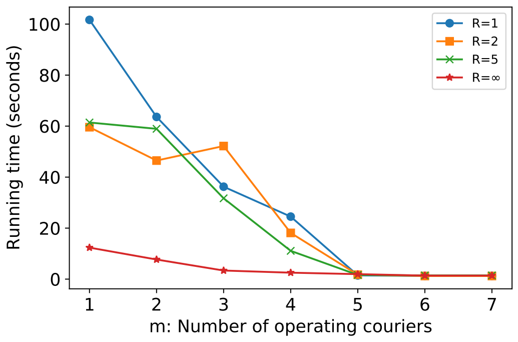
(a) $S = 45, \tau_j \sim Unif(1, 45)$



(b) $S = 60, \tau_j \sim Unif(1, 45)$

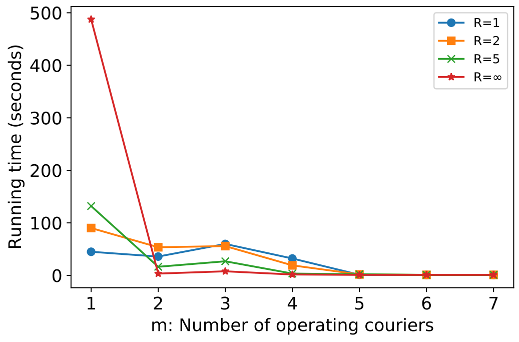


(c) $S = 45, \tau_j \sim Tria(1, 1, 45)$

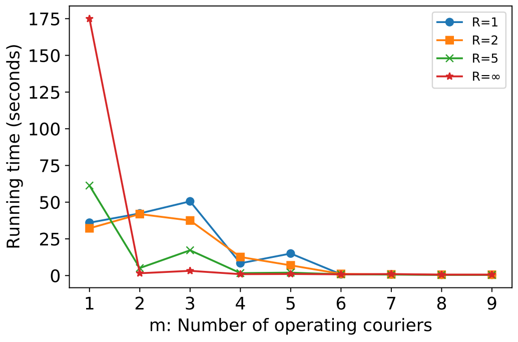


(d) $S = 60, \tau_j \sim Tria(1, 1, 45)$

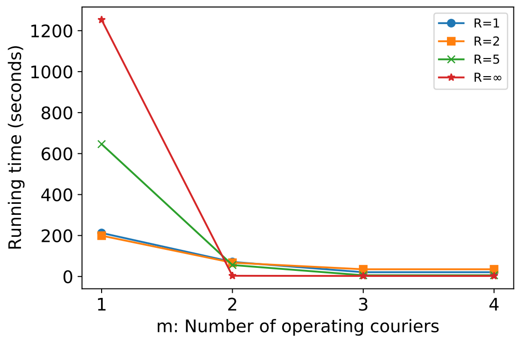
Figure 17: Average running time for the service radius management problem, single depot, $n = 150$



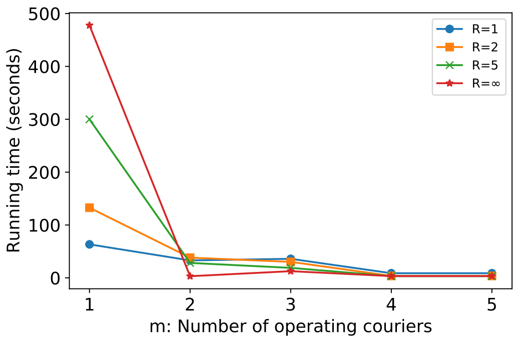
(a) $U = 60, S = 45$



(b) $U = 90, S = 45$



(c) $U = 60, S = 60$



(d) $U = 90, S = 60$

Figure 18: Average running time for the service radius management problem, two depots, $n = 150$