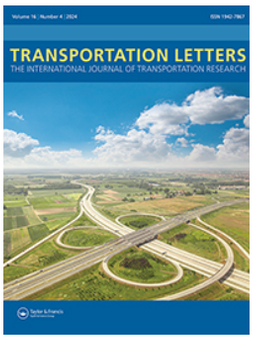


Taylor & Francis
Taylor & Francis Group

Transportation Letters

The International Journal of Transportation Research

ISSN: (Print) (Online) Journal homepage: www.tandfonline.com/journals/ytrl20

Online flash delivery from multiple depots

Maximilian Kronmüller, Andres Fielbaum & Javier Alonso-Mora

To cite this article: Maximilian Kronmüller, Andres Fielbaum & Javier Alonso-Mora (18 Nov 2023): Online flash delivery from multiple depots, Transportation Letters, DOI: 10.1080/19427867.2023.2278859

To link to this article: https://doi.org/10.1080/19427867.2023.2278859

Published online: 18 Nov 2023.

TRANSPORTATION LETTERS
https://doi.org/10.1080/19427867.2023.2278859

Taylor & Francis
Taylor & Francis Group

Online flash delivery from multiple depots

Maximilian Kronmüller[a], Andres Fielbaum[b] and Javier Alonso-Mora[a]

[a]Autonomous Multi-Robots Lab, Delft University of Technology, Delft, Netherlands; [b]TransportLab, School of Civil Engineering at the University of Sydney

ABSTRACT

We study routing for on-demand last-mile logistics with two crucial novel features: i) Multiple depots, optimizing where to pick-up every order, ii) Allowing vehicles to perform depot returns prior to being empty, thus adapting their routes to include new orders online. Both features result in shorter distances and more agile planning.We propose a scalable dynamic method to deliver orders as fast as possible. Following a rolling horizon approach, each time step the following is executed. First, define potential pick-up locations and identify which groups of orders can be transported together, with which vehicle and following which route. Then, decide which of these potential groups of orders will be executed and by which vehicle by solving an integer linear program. We simulate one day of service in Amsterdam that considers 10,000 requests, compare results to several strategies and test different scenarios. Results underpin the advantages of the proposed method.

ARTICLE HISTORY
Received 20 January 2023
Accepted 30 October 2023

KEYWORDS
Flash delivery problem; vehicle routing; same-day delivery; multi-depot VRP; on-demand Logistics

Introduction

The possibility to order and have one's goods delivered within the next minutes is appreciated by many customers. For groceries and products of daily need, such services are summarized under the term Flash Deliveries. Young companies offering such services established themselves in recent years. Examples such as Gorillas, Flink, Getir, or GoPuff promise to deliver groceries to customers' homes in minutes. During the last months of 2021, in the Netherlands alone, consumers spent around 40 million euros per month on Flash Deliveries, a trend that is continuously rising (Kantar 2022). Even some supermarket chains are starting their first trials of Flash Deliveries. For instance, a recent collaboration in the Netherlands between the supermarket chain Albert Heijn and the food delivery companies Thuisbezergd and Deliveroo aims to provide faster delivery of groceries (Albert Heijn Nieuws 2022). Similarly, in several countries in South and North America, the delivery company Cornershop has recently merged with Uber with a similar purpose (Cbinsights Research Briefs 2021).[1]

This work tackles the real-world problem of Flash Deliveries, especially planning and routing algorithms that are necessary to compute vehicle plans during operation. This problem has not been formalized yet, and methods to solve it are also unknown, so this paper is devoted to filling that research gap.

The Flash Delivery Problem (FDP) can be described as follows: Orders are placed continuously throughout the day and need to be delivered within a short time window after they get known. The goods need to be picked up at depots and delivered to customers' locations, leveraging a fleet of vehicles. For each vehicle, a trip needs to be found such that a given objective function is optimized, for example, maximizing the number of delivered orders or minimizing customers' waiting time. This paper formally defines the Flash Delivery Problem and proposes a method to find high-quality solutions. As such, the FDP forms a variant of the Same-Day Delivery Problem (SDDP). Moreover, most on-demand last-mile deliveries, such as SDDP, are operated using a single depot and with vehicles' trips planned and fixed when leaving the depot. This paper relaxes these two assumptions, proposing methods to choose the best depot and to update the vehicles' trips online. In all, the here studied problem combines several NP-hard problems, including the capacitated vehicle routing problem (Bernardo, Du, and Pannek 2021; Ralphs et al. 2003) and the multi-depot vehicle routing problem (Montoya-Torres et al. 2015). Moreover, it requires dynamic optimization, and can easily scale to large problem sizes.

To illustrate the concept that considering multiple depots and en-rote adaptions can lead to shorter trips that deliver more orders quicker, we give an example. The example is illustrated in Figure 1. Orders 1 and 2 are known and loaded into the vehicle. While the vehicle is on its tour a new order (order 3) occurs. If using depot A only and not allowing for pre-empty depot returns, the vehicle serves the two loaded orders, following the first part of the solid tour, shown in yellow. Subsequently, it needs to return to depot A and then drive to the new customer individually, the second part of the solid tour, shown in orange. If a second depot was available (depot B) and the possibility of depot returns prior to being empty was allowed, the original tour can be altered online. The vehicle can load the new order at depot B after serving order 1, and can then service order 3 before serving customer 2 (dashed green tour). By doing so, the long way back to the depot (orange part) can be saved, and shorter trips are possible. Further, customer 3 is served more quickly at the price of delaying order 2 slightly. As such, both operators and users can benefit.

The FDP is dynamic and, as such evolves with time; new orders arrive throughout the day. The operation needs to be planned and executed simultaneously. We propose an approach, which is given a specific problem state at a specific time t, it takes a decision which is followed till the time at which the next decision is taken. To solve a single state, we first select potential pick-up locations from the set of depots for each order individually. Second, potential feasible trips

CONTACT Maximilian Kronmüller m.kronmueller@outlook.com Autonomous Multi-Robots Lab, Delft University of Technology, Delft, Netherlands

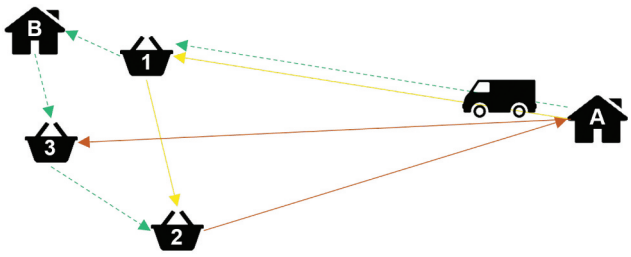


Figure 1. An exemplary tour of one vehicle serving two known orders (1 & 2) and one newly requested order (3) that gets placed after the vehicle has already left depot A. The solid yellow and orange arrows show the trip of the vehicle when there is only one depot and no pre-empty depot returns. The dashed green arrows show the trip when using multiple depots (A & B) and allowing for pre-empty depot returns.

are calculated, i.e. sequences to pick up goods and deliver orders. To assign these trips to vehicles, an integer-linear program is solved. As a result, each vehicle has a constantly updated trip to follow, i.e. which orders to pick up and where, as well as in which sequence to deliver them.

The main contributions of this paper are threefold:

- We formally define the Flash Delivery Problem by modeling it as a Markov Decision Process and propose a method to solve it.
- The proposed method can deal with multiple depots at which orders can be picked up. The method decides endogenously which depot to use for each order. To the best of our knowledge, this is the first work that considers multiple depots per order simultaneously for a dynamic vehicle routing problem without decomposing it into sub-problems, each having a single depot. Further, the approach allows vehicles to visit a depot to load additional orders before distributing their already loaded ones if beneficial.
- Finally, our method can scale up to scenarios with thousands of orders and tens of vehicles. It finds good quality solutions online.

We evaluate the performance of our proposed solution approach by comparing the results to a scenario applying a greedy assignment strategy. Further, we quantify the effects of not allowing the extensions of the second contribution, namely: i) assuming that each order is picked up at its closest depot and ii) prohibiting pre-empty depot returns. A comprehensive sensitivity study analyzes the effects of the number of considered stores, the total number of stores, the effect of allowing to reinsert orders into the problem to enable longer delivery times, the number of used vehicles and the used cost function.

Related work

The FDP is a variant of the SDDP. The SDDP transposes into the FDP if each order needs to be delivered within minutes after being placed instead of until the end of the day. The FDP is a deterministic and dynamic problem following the definition of (Bernardo, Du, and Matias 2023). To the best of our knowledge, there are no works tackling routing for the FDP up to now.[2] For clarity, this work is an extension to the conference paper (Kronmueller, Fielbaum, and Alonso-Mora 2021) and presents a novel, more rigorous formulation of the FDP, additional explanation, clarification and experiments. As such, in Section 2.1, we discuss the most relevant SDDP works. In Section 2.2, we have a look at other related works.

Same-day delivery problem

Both the FDP and the SDDP evolve dynamically over one operational day and must incorporate newly requested orders while executing the trips. The main difference is the deadline in which orders need to be delivered to the customers; in the SDDP the deadline is the end of the day, which can be hours away; in contrast, flash deliveries aim to deliver each order in minutes after receiving them.

This related work section focuses on routing optimizations for the SDDP (Voccia, Campbell, and Thomas 2017), (Ulmer, Thomas, and Mattfeld 2019) and (Côté et al. 2021), routing refers to actively deciding on the routes of vehicles. This excludes works on order assignment or the sole dispatching of vehicles (Azi, Gendreau, and Potvin 2012; Ghiani et al. 2009; Klapp, Erera, and Toriello 2016, 2018, 2020; Ulmer and Streng 2019).

(Voccia, Campbell, and Thomas 2017) use a multi-scenario sampling approach, first introduced by (Bent and Van Hentenryck 2004). They are leveraging waiting strategies and test on scenarios with up to 800 orders and up to 13 vehicles. Similar to our work, [(Ulmer, Thomas, and Mattfeld 2019)] allows for preemptive depot returns, i.e. depot returns before finishing the currently planned tour based on expectations of future events. The authors proposed a method that builds on approximate dynamic programming combined with an insertion routing heuristic. The method allows vehicles to return to depots before finishing their current trips. The method by (Ulmer, Thomas, and Mattfeld 2019) can plan for a single vehicle. [(Côté et al. 2021)] proposes different large neighborhood search-based approaches for the SDDP problem ranging from a re-optimization heuristic to a branch-and-regret heuristic. They rely on a multi-scenario approach to anticipate future events; and their approach is capable of performing preemptive depot returns as well. Algorithms were tested based on the same scenarios as (Voccia, Campbell, and Thomas 2017). Scenarios of up to 10 vehicles were analyzed. A SDDP with micro-hubs was tackled by (Ackva and Ulmer 2022) using a two-stage stochastic programming approach. Also, (Zhen et al. 2023) studies general instant delivery services with deadlines of up to hours. They focus on heterogeneous types of orders and apply a column generation approach. Order acceptance and scheduling for the instant delivery problem, here a deadline of 45 minutes was used, was looked at by (Xue and Wang 2023). The problem is divided into a series of static problems. Orders are inserted online into trajectories based on a similarity measure.

Our work adds to the introduced works by scaling to larger problem sizes and allowing us to consider picking up orders at multiple depots. Further, our approach differs because pre-empty depot returns do not use anticipation of the unknown future but only use currently available information. In contrast, the proposed approach works myopically.

Other related problems

Additional to the SDDP, other problems are related to the FDP. The meal delivery routing problem (Reyes et al. 2018; Yildiz and Savelsbergh 2019; Ulmer et al. 2021) shares the same nature of quick deliveries but has longer lead times and a fixed pick-up location for each order. Similarly, multi-robot task assignment problems (Khamis, Hussein, and Elmogy 2015), but often differing in their focus. They become specifically challenging if incorporating heterogeneous and unreliable robots, each equipped with different capabilities needed to serve different kinds of tasks.

Additionally, vehicle routing to transport people, the dial-a-ride problem (Alonso-Mora et al. 2017; Cordeau and Laporte 2007) is related, especially, pooled dial-a-ride problems. The FDP mainly

differs in two aspects. First, customers do not mind where their goods are picked up from. As such, this is up to the approach to decide unless there is a single option. Some ridesharing works also try to loosen fixed pick-up points, as in (Fielbaum, Bai, and Alonso-Mora 2021), who consider the option that passengers walk short distances. Second, the urgency of picking up an order fast is lower for delivering goods than for transporting people, as humans dislike waiting times. An overview of ridesharing methods can be found in (Agatz et al. 2012; Mourad, Puchinger, and Chu 2019; Narayanan, Chaniotakis, and Antoniou 2020).

The approach proposed in the work is based upon a routing method for a ridesharing system (Alonso-Mora et al. 2017) that transports people in metropolitan areas. This method is called Vehicle-Group Assignment Method (VGA). VGA splits the procedure into two steps: First, it generates potential groups of orders that each vehicle can serve, and second, an optimal assignment of these potential groups to individual vehicles is computed. With realistic enough computation time, the method can solve large-scale real-world instances, up to thousands of vehicles, in an anytime optimal manner.

In regard to considering multiple depots for dynamic problems, this work is connected to the dynamic multi-depot vehicle routing problem (DMDVRP). Only a few works tackled this problem. It has been tackled by decomposing the problem into multiple single-depot dynamic vehicle routing problems (DVRP), where each order is assigned to one fixed depot, and each sub-problem is solved separately (Yu et al. 2013; Xu, Pu, and Duan 2018). In contrast, we include the decision of which depot should be used within the routing decision itself, and thus, this paper is the first, up to our knowledge, to consider multiple depots simultaneously for a DVRP.

Problem formulation

This section presents our mathematical model of the FDP. Because the problem is dynamic, we model it as a Markov Decision Process (MDP). In the FDP, a vehicle fleet must pick up orders at one of the multiple depots and deliver them to the customer's goal locations. Orders are placed dynamically over the course of the operation. Time is denoted as t. The operation starts at $t = T_{start}$ and ends at $t = T_{end}$.

The fleet $\mathcal{V}$ consists of M identical vehicles. Vehicles v are ground-bound, have a maximum capacity of C, and are assumed to drive with constant speed μ along the roads of a street network.

This street network, the operational environment, is described using a weighted directed graph $G = (N, \mathcal{A})$ where N defines a set of nodes and $\mathcal{A}$ defines a set of weighted arcs. Each node represents a potential delivery location. The arcs' weights represent the traveling times between two connected nodes.[3] We denote the shortest travel time between any two locations $n_1, n_2 \in N$ by τ_{n_1,n_2}, which is calculated as the sum of all weights of traversed arcs following the shortest-path. A depot or store $\xi \in N$ is a specific node where goods can be picked up. There are $\mathcal{H}$ depots in total, which are summarized in the set of depots $\Xi \subset N$. We assume that every depot has all goods that customers can order in stock, meaning every order can be picked up at any depot.[4]

The demand set is denoted by $\mathcal{O}$ and consists of all individual orders placed by customers. A total of $U = |\mathcal{O}|$ orders are placed. Each order $o = (t_o, g_o) \in \mathcal{O}$ is revealed at time t_o and has to be delivered to its destination $g_o \in N$. We assume $t_o \in [T_{start}, T_{end} - \delta_T]$, where δ_T is a constant time span before the end of the operation, in which no more orders are placed. For simplicity, we assume all orders are the same size,[5] set to one. This assumption can easily be extended to variable order sizes. Note that an order itself does not specify a depot to use (pick-up location) $p_o \in \Xi$.[6] With time, the status of an order evolves. As such, at time t, the demand set $\mathcal{O}$ can be split into subsets depending on the status of each order $o \in \mathcal{O}$: The set $\mathcal{LO}_t$ consists of all orders $o \in \mathcal{O}$ that are currently loaded to any vehicle $v \in \mathcal{V}$. The set $\mathcal{DO}_t$ consists of all orders $o \in \mathcal{O}$ that were delivered to their destinations g_o before t. The set $\mathcal{JO}_t$ consists of all ignored orders that can not be delivered within the problem's constraints at time t. The set $\mathcal{PO}_t$ consists of all orders $o \in \mathcal{O}$ that are already known (i.e. $t_o \leq t$) but have not been picked-up, delivered or ignored yet. For completeness, $\mathcal{UO}_t$ is the set of all unknown orders, consisting of all orders $o \in \mathcal{O}$ such that $t_o > t$. The subsets are defined such that each order only belongs to one subset at time t, thus they are disjoint, and fulfill $\mathcal{O} = \mathcal{UO}_t \cup \mathcal{PO}_t \cup \mathcal{LO}_t \cup \mathcal{DO}_t \cup \mathcal{JO}_t$. At the beginning of the day ($t = T_{start}$), all orders are unknown, i.e. $\mathcal{UO}_{T_{start}} = \mathcal{O}$. At the end of the day ($t = T_{end}$), all orders are either delivered or ignored, i.e. $\mathcal{DO}_{T_{end}} \cup \mathcal{IO}_{T_{end}} = \mathcal{O}$ and $\mathcal{UO}_{T_{end}} = \mathcal{PO}_{T_{end}} = \mathcal{LO}_{T_{end}} = \emptyset$.

Major point of distinction of the SDDP and the FDP is the latest point when an order must be delivered before being considered failed. We assign each order a maximal drop-off time $t_{drop,o,max} = t_{ideal,o} + \delta_{delay}$, where δ_{delay} is the maximally allowed delay per order, and is predefined by the operator to ensure a desired service level. For the FDP, δ_{delay} is in the order of minutes. Each order is allowed to have a maximum delay of δ_{delay} otherwise, the order is ignored $\theta_o \leq \delta_{delay} \quad o \in \mathcal{O} \backslash \mathcal{JO}$. Hereby, θ_o is the actual delay of order o. It is calculated as the difference between the ideal and the actual delivery time, $\theta_o = t_{drop,o} - t_{ideal,o} \geq 0$. The earliest time an order can be delivered is described by $t_{ideal,o}$. To do so, an idle vehicle needs to be located at the closest depot to the order's destination $\xi_{\text{best},o}$, and start serving the customer immediately without any detours, resulting in $t_{ideal,o} = t_o + \delta_{load} + \tau_{\xi_{\text{best},o},g_o} + \delta_{service}$. Note that we assume that vehicles need some constant time to load or deliver a single order, denoted by δ_{load} and $\delta_{service}$, during which they are parking. Last, the times at which an order o is picked up and dropped off are denoted by $t_{pick,o}$ and $t_{drop,o}$, respectively. A summary of all involved points in time for one order is illustrated in Figure 2.

Following (Ulmer et al. 2020) on modeling MDPs for dynamic vehicle routing problems, we define decision points, the problem state, a decision, a transition between states, a reward and an objective. Further, an initial state at $t = T_{start}$ needs to be set. Generally, given a state at a decision point, a decision is taken based on the reward, and the problem transitions to the next state at the next decision point.

The set of **decision points** is denoted as ψ, which can be determined during operation or beforehand. Individual decisions and corresponding states are enumerated by k. The time at decision point k is t_k and the problem is characterized by the state S_k.

The **state** S_k contains all information needed to fully characterize the problem at t_k and make decisions. In the FDP, the state S_k is fully characterized by the time itself t_k, the vehicle's fleet state, denoted as $\mathcal{V}_k$, and the set of orders to be delivered. Thereby, the fleet's state $\mathcal{V}_k$ are the states of all individual vehicles $v \in \mathcal{V}$ at t_k. At each time t, a single vehicle $v \in \mathcal{V}$ is fully described by its current location $l_{v,t}$, and the orders it has loaded (picked up and not yet dropped off), denoted as the set $\mathcal{LO}_{v,t}$. These definitions allow us to describe the state $\mathcal{S}_k$ formally as

$$\mathcal{S}_k = (t_k, \mathcal{V}_k, \mathcal{PO}_k).$$

For the **initial problem state** $\mathcal{S}_0$, with $k = 0$, at time $t_0 = T_{start}$, we assume that all vehicles $v \in \mathcal{V}$ are equally distributed over all depots $\xi \in \Xi$ and are empty $\mathcal{LO}_{v,T_{start}} = \emptyset \quad v \in \mathcal{V}$.

The **decision/action** a_k at t_k is to assign each vehicle a plan, which it follows till the next decision point at t_{k+1}. For clarity, we refer to the

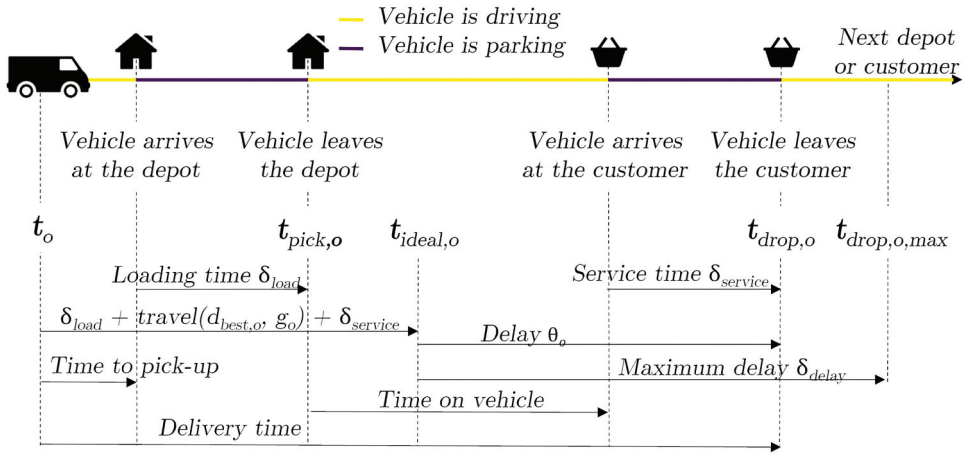


Figure 2. Visualization of the different times and time spans for one order.

plans as trips. A trip $_v$ of a vehicle v is defined as an ordered set of locations $n \in N$, each assigned one of the following activities. At each location, the vehicle either picks up an order, delivers an order to a customer or waits for further instructions. Between locations, the vehicle follows the shortest path. As such, a trip delivers a set of orders which, for simplicity, are denoted as o_T. Note that a trip can be longer than the time span between subsequent decision points, and a previous trip can be updated, followed further or canceled entirely. A decision in the FDP is to decide on a trip T_v for each vehicle v, which it will follow until the next decision point t_{k+1}. The number of orders considered in the decision a_k is $|o_{a_k}|$. Further, each vehicle trip T_v needs to obey the following constraints to be feasible. The vehicle's maximum capacity C needs to be respected, $\mathcal{LO}_{v,t} \leq C \quad v \in \mathcal{V},\ t \in [0, T_{end}]$. Second the orders, which will be delivered through the trip o_{T_v}, need to be delivered before their respective deadline at $t_{drop,o,max}$.

In contrast to (Ulmer et al. 2020), we do not model a **reward** to maximize but equivalently a **cost** to minimize. The cost of a decision a is the sum of costs to execute the trips of all vehicles plus extra costs for the orders that are not considered in any trip. First, we formulate a general cost function that considers the operator's and customers' costs. The customer's cost is based on the orders. The cost of order o is defined as its delay θ_o, so that it measures the quality of service. Thus, the faster an order is delivered, the better. The operator's costs are defined as the traveling time of the vehicle τ_v to serve all orders assigned to it. The two costs are combined convexly via the cost weight β. Last, we add a fixed cost α for each order o that is in the set $\mathcal{PO}_k$, but is not considered in the decision a. The penalty α can be interpreted as a potential cost the operator has to cover if a third party is hired to deliver the respective order. Note that these orders are not necessarily ignored, as they might be included in later decisions. As such the costs for a decision a_k at t_k are calculated following Equation 1,

$$J(a_k, t_k) = \left[(1-\beta) \cdot \sum_{o_{T_v}} \sum_{T_v \in a_k} \theta_o \quad + \beta \cdot \sum_{T_v \in a_k} \tau_{T_v} \quad + \alpha \cdot (|\mathcal{PO}_k| - |o_{a_k}|) \right] \tag{1}$$

In this work, we set α to be considerably larger than the sum of the other two cost terms, meaning that the system first aims at maximizing the number of served orders, and then to minimize the combination of operators' cost and customers' cost.

The **transition** from a current state $\mathcal{S}_k$ to a future state $\mathcal{S}_{k+1}$ can be split into two. On one side, a deterministic part, which consists of two aspects. First, the transition of the vehicle fleet's status $\mathcal{V}_t$. This transition is known and only determined by the made decision a_k. Second, following the trips, some orders get loaded or are considered ignored, thus are not in the set $\mathcal{PO}$ anymore. On the other side, $\mathcal{PO}$ changes as customers place new orders. This transition is unknown exogenous information. We assume to have no knowledge about these future orders and also do not include any predictions about them. The orders are fully known once placed, and we do not consider any demand uncertainties such as (Bernardo, Du, and Pannek 2021). Figure 3 depicts a schematic visualisation of the transition between subsequent states.

We formulate the **objective** of the FDP to minimize overall costs at the end of the operation. The overall objective function at $t = T_{end}$ is represented by Equation 2,

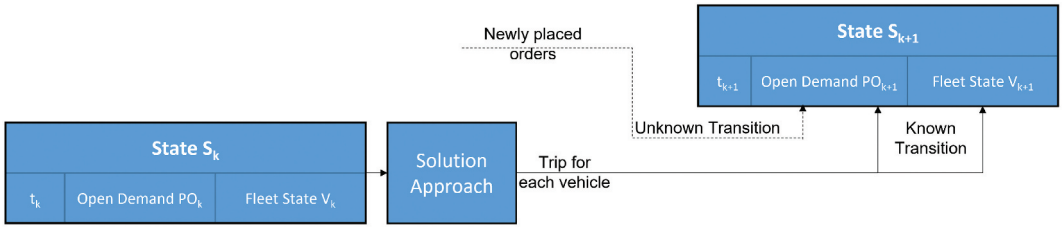


Figure 3. Visualization of the transition between two consecutive states. The transition of the vehicle fleet is known. In contrast, the transition of the open demand is partly unknown due to customers placing new orders.

$$\mathcal{J}_{T_{end}} = \left[(1-\beta) \cdot \sum_{o \in \mathcal{DO}_{T_{end}}} \theta_o + \beta \cdot \sum_{v \in \mathcal{V}} \tau_v + \sum_{o \in \mathcal{TO}_{T_{end}}} \alpha \right] \quad (2)$$

Two details worth highlighting. First, the sum of individual rewards of all decisions and the overall objective at the end of the day are not identical. This is because trips of vehicles T_v can span greater times than Δt and that they are subject to change. Further, not considered orders in a decision are not identical to the finally ignored ones. Second, the method we propose in Section 4 does not depend on this specific cost function (and reward); in other words, a different cost function could be used, and the proposed method still applies.

Method

This section gives a short overview of the proposed method and subsequently explains each method's component in detail.

Method overview

The set of decision points ψ is constructed by dividing the full operation into steps. We do so by a fixed step size of Δt. This results in $\mathcal{K} = T_{end}/\Delta t$ decisions from start to end of the operation. A fixed time step Δt means that our approach is 'batch-based,' in which a number of requests are accumulated before deciding how to assign, as opposed to 'event-based' approaches, where each request is assigned as soon as it appears. The extra information allows to make better decisions, as has already been acknowledged by the industry (Uber 2022).

Our approach is myopic, i.e. it does not explicitly consider future states. We regard this assumption as reasonable (not optimal), as Δt, the time between two consecutive decisions, is rather short (100 seconds in our experiments) and trips T span longer times. Thus, new information is included to the problem fast and previous solutions are updated frequently. Further, myopic approaches are usual in the scientific literature, although anticipatory techniques can be used to improve the solutions. For a discussion on this topic, see (Bent and Van Hentenryck 2004; Fielbaum, Kronmüller, and Alonso-Mora 2021; Hyland et al. 2020; Ulmer et al. 2019).

To take a decision a_k given a state $\mathcal{S}_k$ we propose a method divided into four steps: First, potential pick-up locations for each order are found. Second, orders with associated pick-up locations are grouped into potential trips, taking the current location of each vehicle into account. With enough computational time, we calculate all possible trips for each vehicle. Third, we decide which of these potential trips are being executed. Last, vehicles follow their assigned trips as time is propagated forward until the next decision is taken. These steps are explained in the next sections. An overview of the approach is depicted in Figure 4.

Finding pick-up locations

Each individual order $o \in \mathcal{O}$ needs to be assigned to a specific pick-up location $p_o \in \Xi$. A depot $\xi \in \Xi$ is a feasible option for an order o if a vehicle can pick up the goods at ξ and delivery them in time. Each order might have more than one feasible depot. To select one of these options, we first define the term candidate c of an order $o \in \mathcal{O}$ as follows.

Definition: A candidate c is a tuple containing an order $o_c \in \mathcal{O}$ and an associated pick-up location $p_c \in \Xi$. Thus, a candidate is described as $c = (o_c, p_c)$.

A candidate c is unique, but one order $o \in \mathcal{O}$ can have multiple candidates, each having a different pick-up location $p_c \in \Xi$. $\mathcal{J}_o^{\mathcal{C}}$ denotes the set of candidates that belong to order o. The set of all candidates is denoted by $\mathcal{C}$. $\mathcal{C}_k$ is the set of candidates at time t_k corresponding to all placed orders $o \in \mathcal{PO}_k$.

We introduce a tuneable heuristic to select a subset of pick-up locations. We do so to control the number of candidates per order and, thus, the number of potential trips for each vehicle, which is directly correlated to the required computational effort. For each order, we consider the x depots closest to the order's destination in terms of travel time. The parameter x can be tuned. This results in maximally x candidates per placed order. If $x = H$, all feasible depots are considered, and if $x = 1$, only the closest depot is considered for each order. For $x = 1$, the approach resembles a decomposition of the full problem into multiple single-depot problems. In decomposition approaches, vehicles are fixed to one depot, which is more restrictive than our approach even if we use $x = 1$.

Trip generation

In the trip generation step at t_k we calculate the set of feasible trips $\mathcal{T}_k$. This set describes potential trips that vehicles can follow. Recall, we define a trip T_v of a vehicle v as an ordered set of locations $n \in N$, each assigned one of the following activities. At each location, the vehicle either picks up an order, delivers an order to a customer or waits for further instructions. Between locations, the vehicle follows the shortest path. As such, a trip delivers a set of orders which, for simplicity, are denoted as o_T. In the same fashion, a trip delivers candidates which, equivalently, are denoted as c_T.

The trip generation process is done iteratively, it starts by calculating small trips. We do so to leverage the idea that a trip can only be feasible if all its sub-parts are feasible as well. A trip's size l, measured as the number of considered candidates, is thereby step-

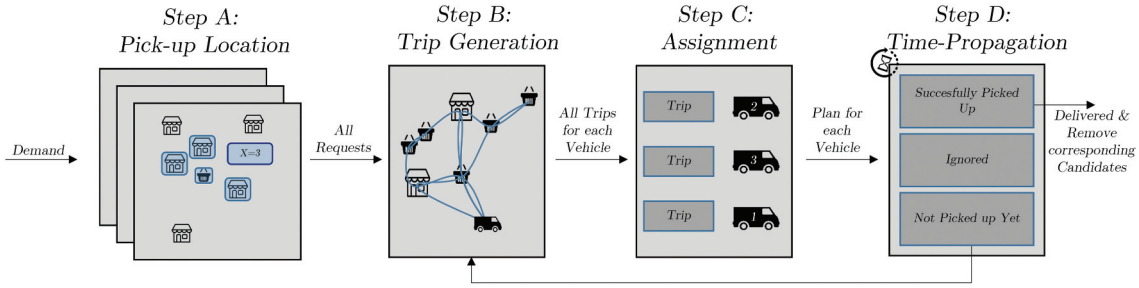


Figure 4. Schematic overview of our solution approach. Step a assigns several potential pick-up locations to each order. During step B, individual candidates c (combinations of orders and specific pick-up locations) are combined to feasible trips. In step C, trips to be executed and corresponding vehicles are selected. Within step D, we propagate time and vehicles follow their assigned trips.

wise increased starting at a size of one until a maximum size η is reached. The operator sets η. Additionally, huge trips are prevented as each order has a latest drop-off time $t_{drop,o,max}$. The result of this step is a set of potential trips for each vehicle.

The algorithm to calculate the set of all feasible trips $\mathcal{T}_k$ at time t_k is shown in Algorithm 1. In Algorithm 1 we use four functions: *CandidateVehicle*(), *TwoCandidates*(), *FeasibleTrip*() and *BestTripSequence*(), each explained in detail in the following:

- The binary logic function **Candidate Vehicle**(v, c) is valid if vehicle v can feasibly serve candidate c.
- The binary logic function **Two Candidates**(c_i, c_j) checks whether the two candidates c_i and c_j are combinable, i.e. if they can both be served by a hypothetical vehicle located at the corresponding depot satisfying all the constraints. As multiple candidates per order exist, we add a constraint to the existing time and capacity constraints: For two candidates to be combinable into one trip, we require them to share their pick-up location.
- The binary logic function **Feasible Trip**(v, T) checks whether all orders of a trip T can be feasibly served by the vehicle v.
- If a trip T is feasible, we determine the sequence in which to deliver all its candidates using the function **Best Trip Sequence**(T).

The cost of visiting a sequence of locations in trip T by vehicle v is given by $\gamma_{T,v}$, which is derived from Equation 2, and calculates as follows:

$$\gamma_{T,v} := (1-\beta) \cdot \sum_{o_T} \theta_o + \beta \cdot \tau_T, \tag{3}$$

where τ_T represents the total travel time to complete trip T. For vehicles that already contain load, the sequence includes those loaded orders. The sequence in which the prior loaded and new orders are served is not fixed. Herein the possibility of pre-empty depot returns occurs. We only keep the trip that minimizes the costs (Equation 3) for a specific vehicle and a set of candidates. Taking the minimal cost trip is included in the subsequent notation of a trip T. Calculations for one vehicle are stopped if a predefined time, ρ_{max}, has passed. In this case, the trips generated up to this point are considered.

Assignment of trips to vehicles

After calculating the set of potential feasible trips $\mathcal{T}_k$ in the previous step, we need to decide which of them should be carried out. We call this step the Assignment of Trips to Vehicles. The assignment is formulated as an integer linear program (ILP). The ILP is presented in Equations 4-8.

$$argmin_\chi \sum_{T,r \in \in_{\mathcal{TV}}} (\gamma_{T,v} - \gamma_{loaded,v}) \in_{\mathcal{T},v} + \sum_{o \in \{1,\ldots,|\mathcal{PO}_t|\}} \alpha \chi_o \tag{4}$$

Algorithm 1: Trip Generation for decision a_k at t_k

input : $S_k, \mathcal{C}_k, \eta$
output: All feasible trips $\mathcal{T}_k$
begin
  $\mathcal{T}_k = \emptyset$;
  foreach $v \in \mathcal{V}$ **do**
    $\mathcal{T}_\ell = \emptyset \quad \forall \ell \in \{1, ..., \eta\}$ (Set of all trips of size ℓ);
    [add trips of size 1]
    foreach $c \in \mathcal{C}_k$ **do**
      if $CandidateVehicle(v, c)$ *valid* **then**
        $\mathcal{T}_1 \leftarrow \mathcal{T}_1 \cup (c)$ (Add trip to set of trips)
      end
    end
    [add trips of size 2]
    foreach (c_i), $(c_j) \in \mathcal{T}_1$ **do**
      if $TwoCandidates(c_i, c_j)$ *valid and* $FeasibleTrip(v, c_i, c_j)$ *valid* **then**
        $\mathcal{T}_2 \leftarrow \mathcal{T}_2 \cup BestTripSequence(v, c_i, c_j)$
      end
    end
    [add trips of size ℓ]
    for $\ell \in \{3, ..., \eta\}$ **do**
      foreach $T_i, T_j \in \mathcal{T}_{\ell-1}$ *with* $|T_i \cup T_j| = \ell$
      (The two combined trips contain ℓ candidates together) **do**
        if $\forall h \in \{1, ..., \ell\}, \{c_1, ..., c_\ell\} \setminus c_h \in \mathcal{T}_{\ell-1}$
        (Each subset of this trip is a feasible smaller trip) **then**
          if $FeasibleTrip(v, T_i \cup T_j)$ *valid* **then**
            $\mathcal{T}_\ell \leftarrow \mathcal{T}_\ell \cup BestTripSequence(T_i \cup T_j)$;
          end
        end
      end
    end
  end
  return $\mathcal{T}_k \leftarrow \cup_{\ell \in \{1,...,\eta\}} \mathcal{T}_\ell$
end

$$\sum_{T \in \mathcal{J}_v^T} \in_{\mathcal{T},v} \leq 1 \qquad v \in \mathcal{V} \tag{5}$$

$$\sum_{c \in \mathcal{J}_o^C} \sum_{T \in \mathcal{J}_{c_o}^T} \sum_{v \in \mathcal{J}_T^V} \in_{\mathcal{T},v} + \chi_o = 1 \qquad o \in \mathcal{PO}_t \tag{6}$$

$$\chi_o \in \{0,1\} \tag{7}$$

$$\in_{\mathcal{T},v} \in \{0,1\} \tag{8}$$

Thereby, $\in_{\mathcal{TV}}$ denotes the set of all feasible trip vehicle combinations, and $\in_{\mathcal{T},v}$ is the corresponding binary variable, taking the value 1 if the combination is executed. Further, we define the following sets: $\mathcal{J}_v^T$, the set of trips that can be serviced by a fixed vehicle $v \in \mathcal{V}$; $\mathcal{J}_c^T$, the set of trips that contain candidate c; $\mathcal{J}_T^{\mathcal{V}}$, the set of vehicles that can service trip T; $\mathcal{J}_o^{\mathcal{C}}$, the set of candidates that belong to order o. Further, χ_o is a binary variable, taking the value of one if the corresponding order is ignored, and $\mathcal{X}$ is a set of all variables $\mathcal{X} = \{\in_{\mathcal{T},v}, \chi_o; \quad \in_{\mathcal{TV}} \text{ and } \quad o \in \mathcal{O}\}$.

Equation 4 describes the objective function. Note that the considered costs are relative. From the costs of a vehicle's trip $\gamma_{T,v}$ (see Equation 3), the costs for the considered vehicle to serve its already loaded orders are subtracted, $\gamma_{loaded,v}$. Thus, we only account for changes in the vehicle's trip. If a vehicle's trip is not changed by not assigning any new orders, the assignment poses no costs. Equation 5 ensures that each vehicle is at most assigned to one trip. Equation 6 ensures that each order is assigned to a single vehicle or is rejected in this decision and the penalty α is charged. Furthermore, it ensures that no more than one candidate belonging to the same order is chosen. Equations 7-8 ensure that the corresponding variables are binary. χ_o takes the value one if its associated order $o \in \mathcal{O}$ can not be served by any vehicle or is ignored. Equation 8 defines $\in_{\mathcal{T},v}$ as binary. As a result, each vehicle is assigned to a new trip or does not receive any new orders. If a vehicle receives no new orders, it will follow its current trip of delivering the currently loaded orders or be considered idle if it has none.

To fasten the time needed to solve the above-presented ILP, we initialize it by a greedy solution. The greedy solution is constructed by selecting the largest trip, measured by the number of served candidates l, first. If multiple trips serve the same amount of candidates, the trip with the lowest cost is selected. We remove all trips which include already assigned orders or vehicles. We iterate until there are either no more vehicles or no more orders to assign.

If a vehicle is considered idle after an assignment, we perform a rebalancing step. The corresponding vehicle's trip sends it to the closest depot from its current location. We do so to enable the vehicle to pick up orders quickly in the following steps. Nevertheless, it may still be assigned otherwise in a future time step before reaching that depot.

Time-propagation

In this step, we propagate time and update all elements affected by it, until the next decision $k+1$ is triggered, $t_{k+1} = t_k + \Delta t$. Each vehicle follows its trip determined in the decision a_k. As time is propagated, each order can be in one of the following five states: First, an order is **picked up** by a vehicle at a depot ($o \rightarrow \mathcal{LO}_{k+1}$). As soon as an order is picked up its vehicle allocated cannot be changed. Multiple candidates belonging to one order are available, but only one of them is selected, and so all other candidates of the order are removed. gets served, the other candidates belonging to this order are removed. Second, an order is **delivered** to its destination ($o \rightarrow \mathcal{DO}_{k+1}$). Third, an order is assigned to a trip, and the planned pick-up time is later than t_{k+1}, the time of the next decision. Thus, we consider the order as **not picked up**, yet. All not picked up orders, more precisely the associated candidates, are reinserted into the trip generation step for the next decision, thus allowing for reassignment ($o \rightarrow \mathcal{PO}_{k+1}$).

Fourth, an order is assigned to no vehicle. This order (associated candidates) is reinserted into the trip generation step for the next decision ($o \rightarrow \mathcal{PO}_{k+1}$), unless it is no longer feasible to serve it as explained in the next bullet point. Last, an order is **ignored** ($o \rightarrow \mathcal{JO}_{k+1}$), i.e. it is not feasible to deliver it without violating a constraint. All candidates belonging to this order are removed.

Note that an order $o \in \mathcal{O}$ is ignored in the case it can't be delivered before the latest drop-off time $t_{drop,o,max} = t_{ideal,o} + \delta_{delay}$. Hereby, $t_{drop,o,max}$ is mainly influenced by the value of δ_{delay}. The smaller δ_{delay} is set, the harder it is to combine multiple candidates to be served by one vehicle. On the other hand, if δ_{delay} is set too large, the number of possible combinations becomes vast, which can hinder solving the problem in the first place due to increased combinatorial size. A good balance has to be found by the system operator. We distinguish between $\delta_{\text{delay,real}}$, defined by the service level and $\delta_{\text{delay,heuristic}}$, the maximum delay at which the method performs well. In case that $\delta_{\text{delay,heuristic}} < \delta_{\text{delay,real}}$, the former should be used. To adjust to $\delta_{\text{delay,real}}$ we allow a candidate to be reinserted into the problem after it has violated $\delta_{\text{delay,heuristic}}$, but not $\delta_{\text{delay,real}}$. The candidate gets reinserted with a new request time of t_k, the current time. Each candidate can be ignored up to a limit of ζ times, which is defined as:

$$\zeta = (\delta_{\text{delay,real}} - (\delta_{\text{delay,real}} \bmod \delta_{\text{delay,heuristic}}))/\delta_{\text{delay,heuristic}}. \tag{9}$$

When a candidate gets ignored ζ times, it is removed from the problem. Note that for feasibility calculations, the new request time has to be used. Nevertheless, the original request time is used to calculate the users' costs of a candidate on a trip.

Complexity and Optimality Analysis

Complexity

Our approach divides the full-day problem (Section 3) into multiple sub-problems at specific times t_k. Each sub-problem deals with it's associated state S_k. The trip generation step (Section 4.3) is the most complex and thus the bottleneck of the proposed approach. The ILP (Section 4.4) can become large but stays solvable in a reasonable time by state-of-the-art solvers. Thus we analyze the trip generation step in more detail.

Let us do a worst-case scenario analysis, where all the orders are associated with the same x depots, the corresponding candidates are all combinable, and all sets of candidates can be served by any vehicle. Recall that the maximum trip size is η. This leads to a complexity of:

$$\mathbb{O}(|\mathcal{V}| \cdot |\mathcal{O}|^{\eta} \cdot x)$$

If the trips' size become large, limited by η, the complexity can increase rapidly. In practice, the trip size is further influenced by two other factors: First, the density of orders, i.e. the relation of the spatial size of the graph and the size of the set of orders, which affects how orders can be combined. The lower the density of orders is, the harder it becomes to serve them together. As a result, the maximum trip size decreases. Second, a short maximum delivery time also decreases the maximum length of potential trips and also their number.

Optimality

The proposed approach is able to solve a sub-problem, regarding a single state, to optimality. To achieve this, all depots have to be considered ($x = \mathcal{H}$), enough computational time has to be given, and the maximum trip length has to be unconstrained. Note that even if each sub-problem is solved exactly, this does not imply an optimal solution to the full-day problem, due to the myopic approach employed.

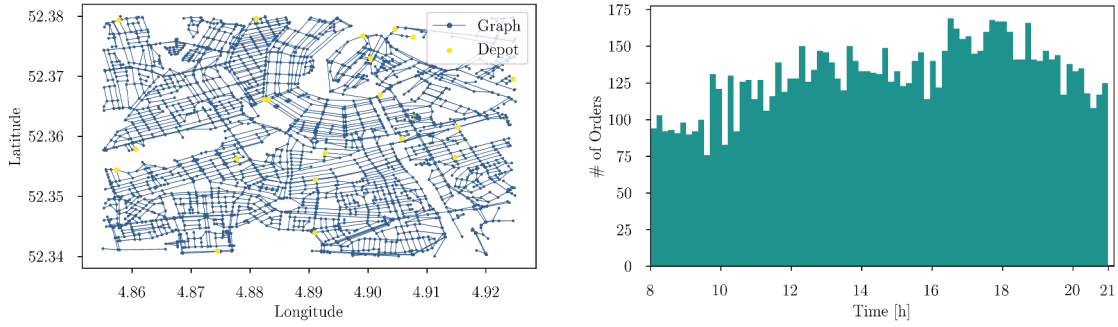
Experiments

In this section, we present the computational experiments. First, Section 5.1 analyzes one run in detail, representing a day of on-demand grocery delivery in Amsterdam, where we are able to deal with thousands of requests. Second, in Section 5.2, we assess the performance of our solution approach by comparing it with a greedy approach, a scenario that considers a single depot per order, and a scenario that does not allow for pre-empty depot returns. Finally, in Section 5.3 we present the results of a sensitivity analysis of the main parameters, including the number of considered stores, the number of vehicles and the used cost functions. Table A1 in the Appendix contains all results of all analyzed scenarios.

Base scenario

To analyze the proposed algorithm, we simulate a potential day in the city center of Amsterdam. We represent the street network as a directed graph containing 2717 nodes and 5632 edges, shown in Figure 5(a). Over the whole service area, there are 20 pick-up depots which have been distributed by a k-center algorithm. The travel times between nodes are calculated as their distance divided by the constant vehicle's speed of $\mu = 36\frac{\text{km}}{\text{h}}$. We simulated a demand of 10,000 orders, homogeneously distributed in space. Time-wise they cover a period from $T_{start} = 08:00$ to $T_{end} = 21:10$, including two peaks: at noon and in the evening. The temporal demand distribution is shown in Figure 5(b). Each bar shows the number of newly placed orders within 10 minutes. In the last 10 minutes, before the end of the day T_{end}, no more orders are placed, $\delta_T = 10$.

The vehicle fleet V has 30 vehicles ($M = 30$) of capacity $C = 6$. The maximum trip size η is set to 10. The maximum delay $\delta_{\text{delay,real}}$ is set as 8 minutes and equal to $\delta_{\text{delay,heuristic}}$, resulting in a ζ of one. Per order, the three closest depots to the final destination ($x = 3$) are considered. To load and service an order, we assume $\delta_{load} = 15\,sec$, implying that all orders are prepared in advance and only need to be loaded, and $\delta_{service} = 30\,sec$, assuming that all customers are ready to grab their groceries at the front door. The algorithm runs in time spans $\Delta t = 100\,sec$. The penalty for ignoring an order is set to equal 10^4 seconds. We weighted the two different objectives with $\beta = 1/3$. These values have been chosen to create a scenario that is serving most orders but cannot serve everything. To solve the ILP described in Equations 4-8, we use the software Mosek 7.1 with a time budget of $50\,sec$. This time budget is enough to find the optimal solutions in about 85% of the cases. Otherwise, the best-obtained solution at that point is used.



(a) Graph representing the city centre of Amsterdam.

(b) Temporal distribution of placed orders.

Figure 5. A visual representation of the underlying graph $G = (N, \mathcal{A})$ is shown on the left. The locations of all 20 depots are highlighted in yellow. On the right side, the temporal distribution of all order's request times $t_o \quad o \in \mathcal{O}$ is depicted. Each bar shows the number of newly placed orders within 10 minutes.

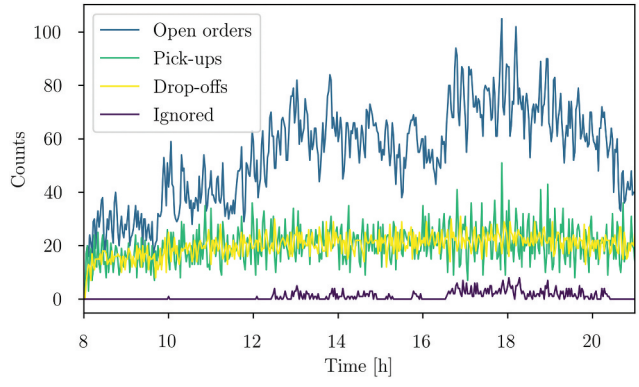


Figure 6. The number of open orders, the number of picked up and dropped off orders, as well as ignored orders per time step are visualized.

First, we evaluate the service rate, which is defined by the percentage of served orders. A service rate of 95.19 % is achieved, which equals 481 ignored orders. Figure 6 shows the number of open orders, pick-ups and drop-offs, and finally, ignored orders per decision. Most ignored orders happen during peak times. Peak times are characterized by large numbers of placed orders. The number of pick-ups shows occasional spikes. These appear as vehicles can load a high number of orders consecutively without driving when they visit a depot.

Second, we analyze different time spans (time KPIs) involved in the delivery process of each order, see Figure 2. The distributions of the time until pick-up (mean: 3 min 50 s), the time a order is loaded onto a vehicle (mean: 3 min 13 s), the delivery time (mean: 7 min 47 s), and the associated delay (mean: 5 min 43 s) are illustrated in Figure 7. These times can be compared to the average distance of all nodes to their closest depot, which is 1 min 20 s. Note that the total delivery time is always greater than 45s, the sum of the loading and service time ($\delta_{load} + \delta_{service}$). The delay distribution increases strongly towards a sharp cut-off at 480s, 8 min, the maximum allowed delay.

Let us analyze the delay in more detail. We distinguish two time windows of 2 hours, one in the morning (09:00 to 11:00) with low workload and one in the evening (17:00 to 19:00) with a high workload. Figure 8 shows the delay distribution for all orders placed in the corresponding time windows. For low workload Figure 8(a), the average delay is significantly lower and the overall shape of the distribution is less pushed toward the maximum delay. With a lower workload, additional resources become available, thereby allowing the improvement of the service level without the necessity of serving additional orders initially. This is also reflected in the number of rejected orders, as shown in Figure 6. In contrast, during high workload Figure 8(b) most orders are served with a high delay.

Third, we analyze how the proposed method utilizes each vehicle. The occupancy of all vehicles is depicted in Figure 9(a). The evening peak of the demand can also be identified through the brighter colors that appear there, meaning that many vehicles have more loaded orders. Idle vehicles only occur at the beginning and end of the day. During the rest of the day, vehicles are immediately used while or after returning to a depot. Figure 9(b) displays the mean number of loaded orders of all vehicles over time. The average load per vehicle over the day is 1.49 orders.

Fourth, we analyze the total traveled distance. In the base scenario, a distance of 8,973.8 km is traveled by all 30 vehicles. All vehicles are used similarly. Driven distance per vehicle ranges from 267.86 km to 311.86 km.

Comparison

We now assess the performance of our method by comparing it to three approaches. First, in Section 5.2.1, we compare the results obtained with our approach with those obtained by using a greedy assignment strategy. Second, in Section 5.2.2, we compare considering multiple depots to picking up each order at its closest depot. Third, we analyze the benefits of pre-empty depot returns in Section 5.2.3. For simplicity, we refer to the scenario analyzed in Section 5.1 as the base scenario. The parameters' values are set as in the base scenario.

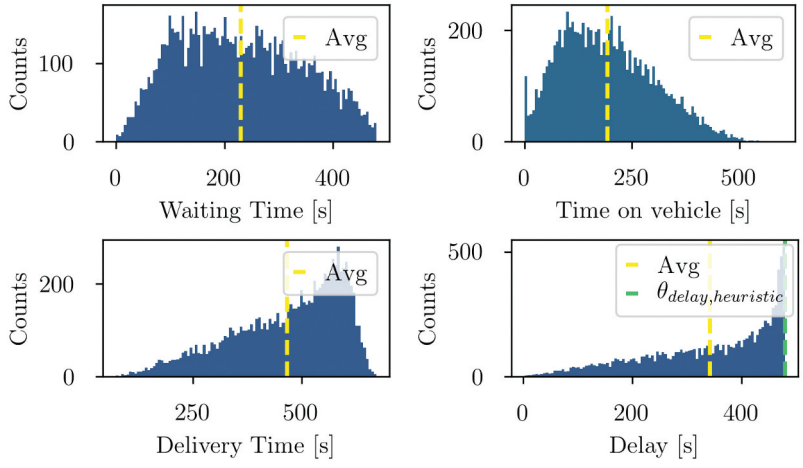
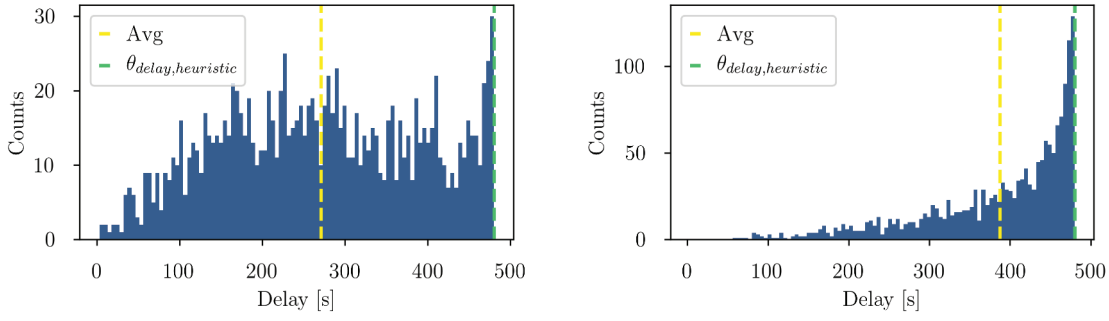
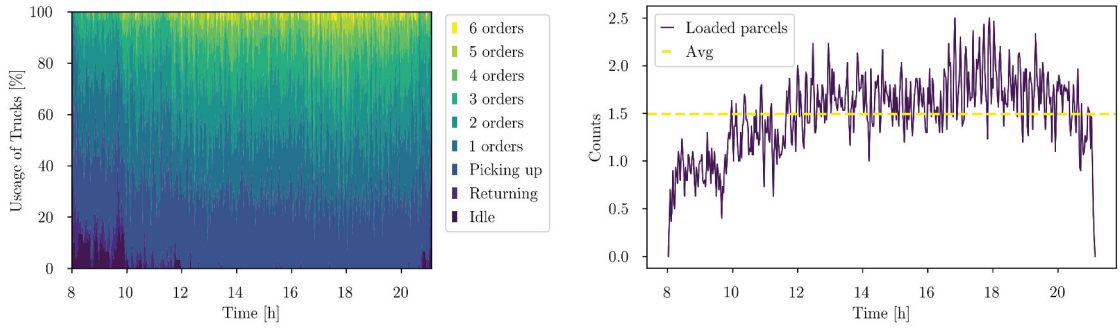


Figure 7. Distributions of time to pick-up, time of orders spend loaded to a vehicle, the total delivery time, and delay of the base scenario.



(a) Delay distribution 9a.m. - 11a.m. (b) Delay distribution 5p.m. - 7p.m.

Figure 8. Distribution of delay for two different time windows differing in workload of the base scenario.



(a) Occupancy. (b) Mean loaded orders.

Figure 9. On the left, occupancy of all vehicles over the entire operation duration, the evening peak can be identified clearly. On the right, the mean number of loaded orders of all vehicles as the day progresses.

Greedy assignment strategy

This section compares the results of the base scenario to those obtained by a greedy assignment strategy. We explain this strategy as follows. Every time an order is placed, the algorithm immediately checks which is the best way to serve it. For that, the strategy checks how the new order could be inserted into each vehicle. Then, the new order is assigned to the best vehicle, i.e. the one that would achieve the minimum added cost if such order is allocated to this vehicle. Thereby, all x pick-up options are considered. If an order can not be added to any vehicle's trip without violating some constraint, then the order is rejected immediately.

The obtained results are visualized in Figure 10. Comparing the two approaches shows a decrease in service rate from 95.19% to 74.18%, while delay and total driven distance increase significantly. This means that with the greedy algorithm fewer orders are delivered, those that are delivered require longer travel times, and total operators costs become larger.

Considering the closest depot only

We compare the base scenario to the case in which each order is picked up at the depot that is closest to its destination (i.e. applying our heuristic, introduced in Section 4.2 with $x = 1$). This is similar to comparing to the case in which the problem is decomposed into several single-depot problems, although this approach is still more flexible as vehicles are not fixed to some specific depot. In the case of comparing to a decomposition approach, we expect larger differences, than the ones, discussed in the following.

For $x = 1$, worse results in terms of service rate and total driven distance are obtained, as shown in Figure 11. The service rate drops from 95.19% to 92.68%, i.e. 251 additionally ignored orders. Even having fewer orders delivered overall, the total driven distance increases by 121.14 km, for $x = 1$. These improvements to the baseline come at the cost of increased delay in the order of seconds, it increases from 5 min 33 s to 5 min 43 s. See Section 5.3.1 for the analysis of additional values of x.

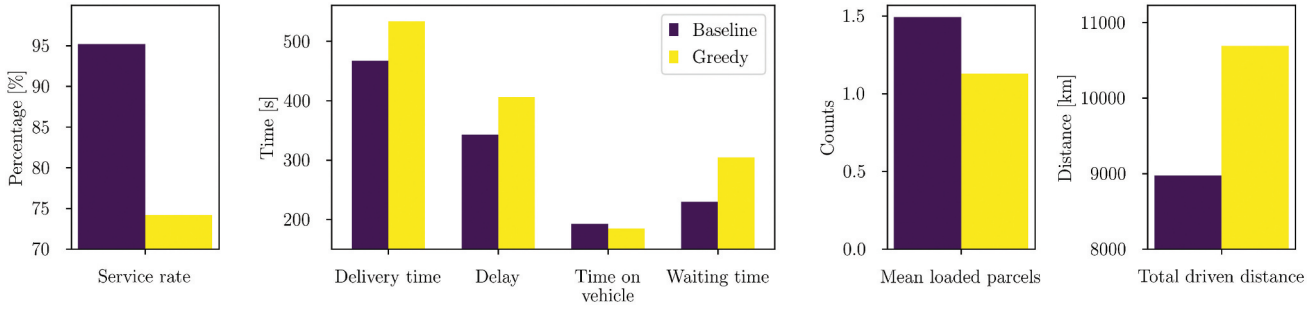


Figure 10. Service rate, time KPIs, mean loaded orders and total driven distance of the base scenario and the greedy assignment strategy.

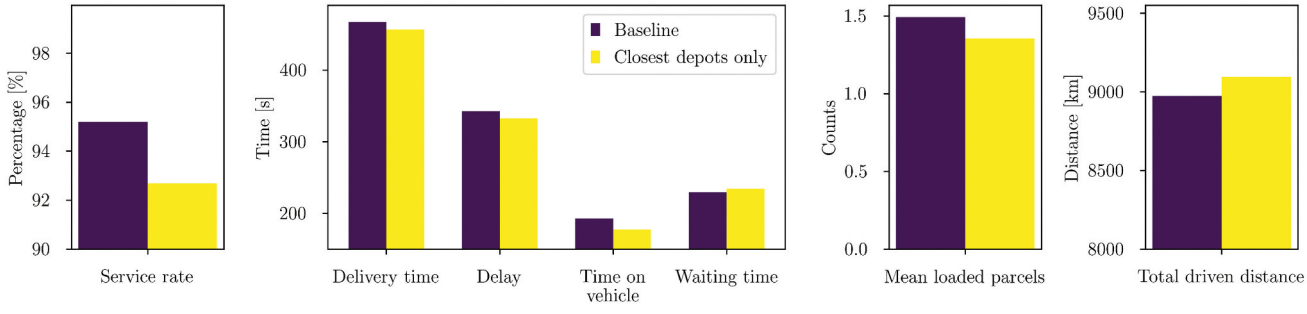


Figure 11. Service rate, time KPIs, mean loaded orders and total driven distance of the base scenario and the same scenario considering the closest depot only.

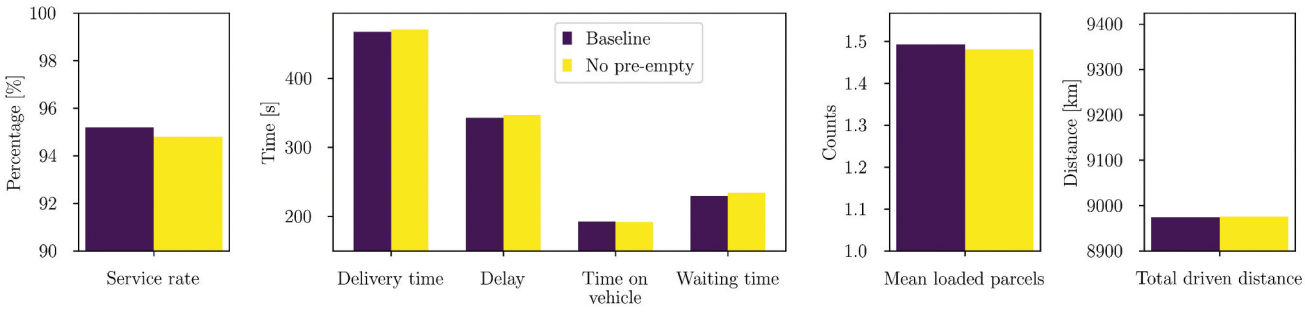


Figure 12. Service rate, time KPIs, mean loaded orders and total driven distance of the base scenario and the same scenario with prohibiting pre-empty depot returns.

No pre-empty depot returns

One of the advantages of our proposed approach is that it allows for pre-empty depot returns. Here, we compare our approach to the case where we prohibit those depot returns. This means that only empty vehicles can load new orders. Results of the comparison are depicted in Figure 12. They show a decrease in service rate (95.19% → 94.81%) and a slight increase for all time KPIs (average delay: 5 min 43 s → 5 min 46 s) and also for the total driven distance (8,973.8 km → 8,975.5 km). Generally, the more depots are used for the operation, the less impactful allowing for pre-empty depot returns is. The difference in results for a similar comparison would be more significant if fewer depots were used, for example, a total of 5 or 10 depots. This is due to a smaller average distance of vehicles to the next depot. We remark that these improvements are fully achieved by modifying the trips without the need for additional infrastructure.

Sensitivity analysis

In this Section, we analyze the effects of five parameters, namely: the number of considered stores, the total number of stores, the effect of allowing to reinsert orders into the problem to enable longer delivery times, the number of used vehicles and the used cost function. We perform a sensitivity analysis for each one of them. We present the analyses in the following sections. All parameters are identical to the base scenario (Section 5.1), unless mentioned otherwise.

Number of considered depots per order, x

Our heuristic, introduced in Section 4.2, considers the x closest depots to an order's destination as potential pick-up locations. Therefore, the more depots are considered, the higher the number

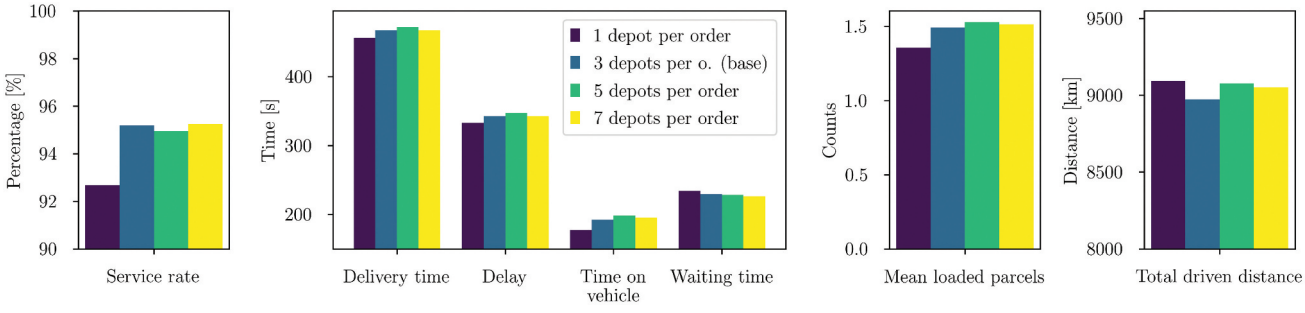
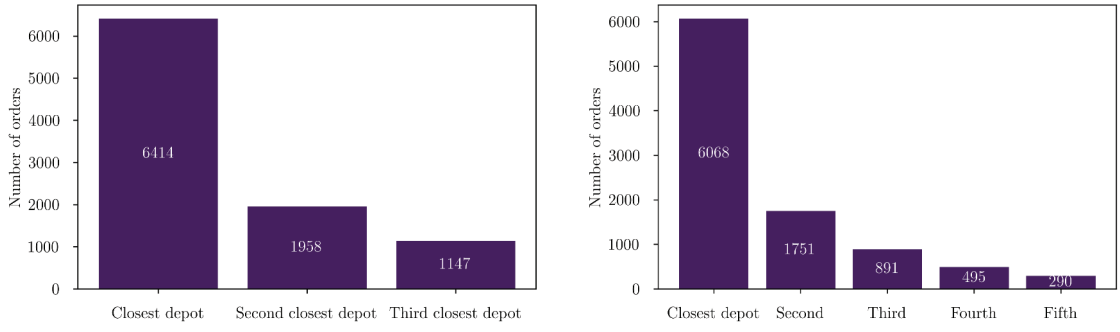


Figure 13. The main performance indices for different numbers of considered depots per order are visualized.



(a) x = 3 (Base scenario). (b) x = 5.

Figure 14. Depots are categorised based on their distance to the goal location of the corresponding order. This figure illustrates how often the depots are used. The left figure shows the case in which three depots are considered per order (baseline). The right figure depicts the same case for five considered depots per order.

of potential candidates, and consequently the higher the computational load.

Figure 13 shows the results if one (see Section 5.2.2), three, five, or seven depots per order are considered.

Service rate rises while total traveled distance decreases when three instead of one depot per order are considered. Both get worse if $x = 5$ and then improve slightly if $x = 7$. Delay increases the more depots are considered with a slight drop if x changes from five to seven. Figure 14 additionally shows the usage of depots, if they are ranked according to the distance to their order's destination, for the baseline Figure 14(a) and the scenario considering five depots per order Figure 14(b). In both cases, the closer the depot, the more frequently it is used, with the most significant change from the closest to the second closest depot. The closest depot was used in both cases over 6000 times, leaving over 30% to be distributed over the others, showing again that it is beneficial to consider them.

On the one hand, considering multiple depots is beneficial, as shown by the improvements compared with considering the closest depot only. On the other hand, it is not always better to consider more depots per order, as revealed by the worse results obtained with $x = 5$. This can be explained by the myopic nature of our approach, which can lead the system into unfavorable states to serve future demand. For a single decision, using more depots is better as it enlarges the set of feasible solutions, but the overall problem's solution can worsen due to chaining multiple states dynamically with each other. For example, a truck might be sent to a depot that is far away when considering only current information, but if some orders appear nearby soon after, it would have been better to go to a closer depot in the first place. We conclude:

- Considering only one depot per order, as (Yu et al. 2013) and (Xu, Pu, and Duan 2018), can be inferior to considering multiple ones.
- To consider as many depots as possible can be inefficient for dynamic problems having imperfect anticipation.

The question 'How many depots should be considered?' emerges. The answer can depend on various factors, including the problem at hand or even the current state. This is outside of the scope of this work and is left for future work.

Total number of depots H

We varied the number of placed depots within the service area. We simulated scenarios featuring 1, 15, and 25 depots in total. Results are depicted in Figure 15. The service rate improves at decreasing rates as more depots are available. Delay shows non-monotonic behavior, which is related to the corresponding service rates. For a single depot, fewer orders are served, i.e. more orders are ignored. When some orders are ignored, the most complicated ones are ignored first, meaning that they would have had a high delay if delivered. The number of mean loaded orders decreases as more depots are available. The total driven distance generally decreases for more depots, except for 15 depots, compared to the single depot case, which the substantial increase in service rate can explain. Generally, the magnitude of changes in performance varies as the number of depots is increased linearly in steps, each of size five. These results raise the question: 'How many depots are optimal, taking the cost to open and operate them into account?.' We consider this question interesting and relevant for future research related to the multi-facility location problem (Farahani and Hekmatfar 2009).

Allowing for reinsertion of orders

In Section 4.5, we distinguished between $\delta_{\text{delay,real}}$, the maximum delay allowed by the operator and $\delta_{\text{delay,heuristic}}$, the maximum delay used for running the proposed algorithm. Recall that this distinction is made to reduce the computational complexity and the time needed to solve the problem. If an order violates $\delta_{\text{delay,heuristic}}$, the orderwould be

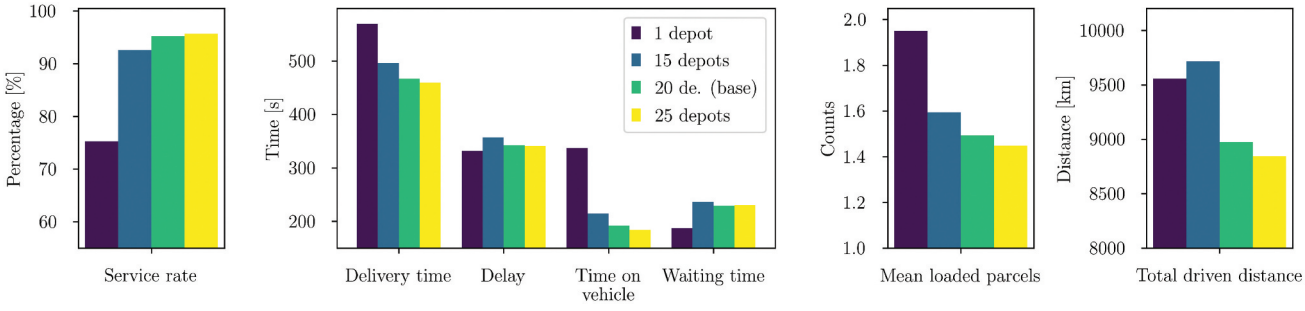


Figure 15. Service rate, time KPIs, mean loaded orders, and total driven distance of the four different runs, featuring a different number of total depots distributed over the service area.

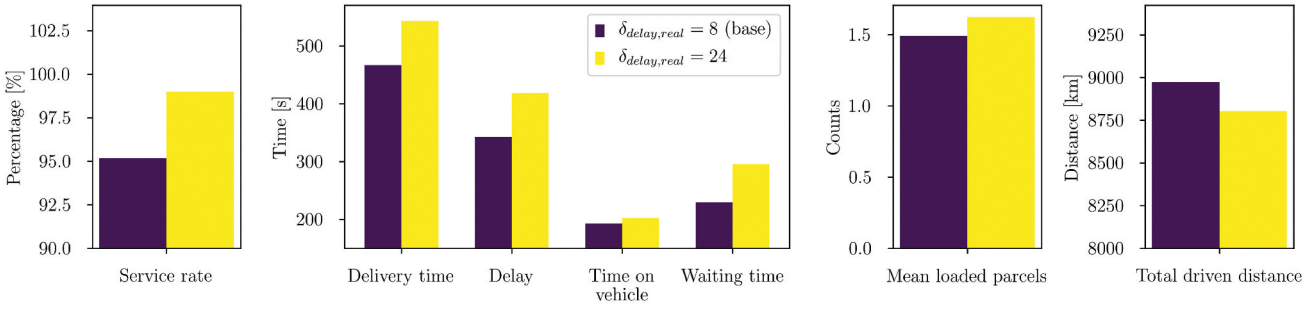
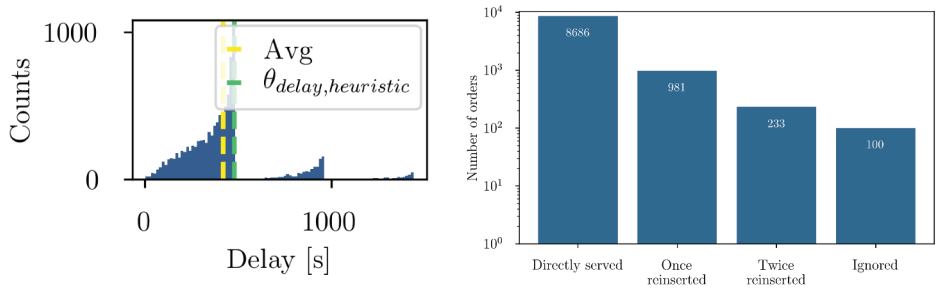


Figure 16. Service rate, time KPIs, mean loaded orders, and total driven distance of the base scenario and the same scenario having a $\delta_{\text{delay,real}}$ of 24 minutes, thus allowing reinsertion.



(a) Delay distribution. (b) Reinsertion count.

Figure 17. The delay distribution having a maximum delay $\delta_{\text{delay,real}}$ of 24 minutes, thus allowing reinsertion is shown on the left. On the right, the number of served orders per reinsert step is illustrated.

considered ignored. But, by allowing reinsertions, see Section 4.5, the order has the chance to be still served, while not violating $\delta_{\text{delay,real}}$.

Here we set $\delta_{\text{delay,real}} = 24$ minutes keeping $\delta_{\text{delay,heuristic}} = 8$, which results in $\zeta = 3$ maximum reinsertions per order. Figure 16 shows a comparison with the $\delta_{\text{delay,real}} = \delta_{\text{delay,heuristic}} = 8$, as in Section 5.1. We see an increase in service rate and a decrease in the total driven distance. All time KPIs increase. Increasing $\delta_{\text{delay,real}}$ allows for a higher delay per order, which leads to an overall higher average delay. The corresponding delay distribution is shown in Figure 17(a), where we observe a repetitive nature. The number of reinsertions is shown in Figure 17(b), showing a significant number of orders that are reinserted at least once.

Number of orders U

We created two alternative demand scenarios, featuring different numbers of customer orders $U = 9,500$ and $U = 10,500$, for the entire day. Both scenarios resemble the distribution of the 10,000 order case in time and space. Figure 18 depicts the corresponding results. The more orders are placed, the lower the service rate because available resources were kept constant. Nevertheless, the absolute number of delivered orders increases (from 9,268 to 9,519, and then to 9,867). Although more orders have been delivered, this does not necessarily increase the total driven distance. Compared with the 9,500 orders case, the driven distance decreases for both 10,000 and 10,500 orders. In general, if there are more orders to serve, it will be easier to find customer destinations close to each other, so the average distance between them decreases. However, those improvements do not hold for the time KPIs as all of them worsen with more placed and served orders. The number of mean loaded orders increases in the same manner.

Number of used vehicles

We varied the number of used vehicles to 25 and 35. Figure 19 shows the effect of the number of used vehicles on the obtained solutions. The service rate increases as more vehicles are used, and similarly, the total driven distance increases. Time KPIs decrease the more vehicles are used as well as the mean loaded orders. The maximum traveled distance by one vehicle stays about the same, as

Figure 18. Service rate, time KPIs, mean loaded orders, and total driven distance of the three different runs, featuring different demand patterns.

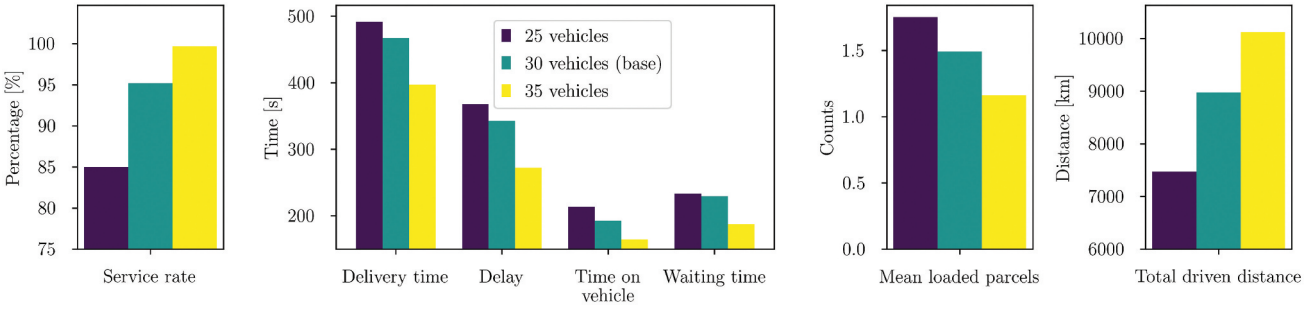


Figure 19. Service rate, time KPIs, mean loaded orders, and total driven distance of the three different runs, featuring a different number of vehicles.

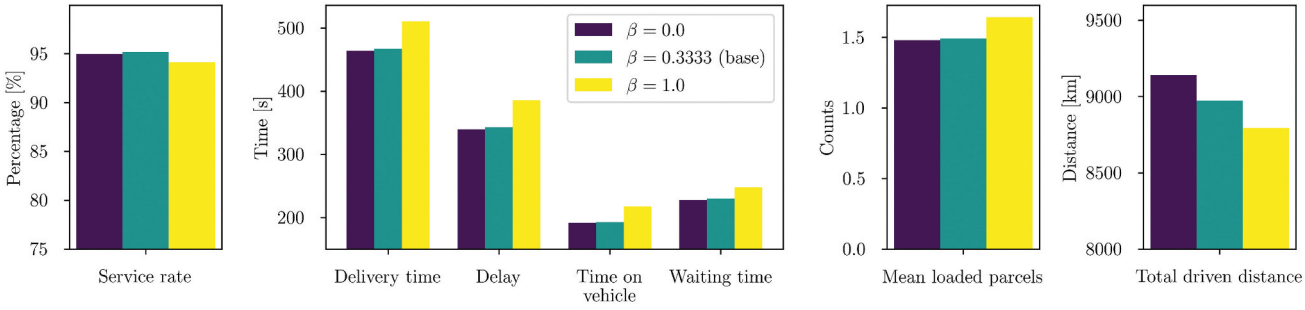


Figure 20. Service rate, time KPIs, mean loaded orders and total driven distance of three runs, featuring different cost weights β.

those vehicles are utilized continuously throughout the whole day. For 25 vehicles, the maximal distance is 308 km, for 30 vehicles is 311 km, and for 35 vehicles is 309 km. On the other hand, the minimal distance traveled by one vehicle varies strongly. For 25 vehicles, the minimal distance is 290 km, for 30 vehicles is 267 km, and for 35 vehicles is just 228 km. When considering 35 vehicles some vehicles are used sparsely, especially at the start of the operation, where the overall workload is still low compared to later in the day.

Cost function weight β

The cost weight β, shaping the cost terms in Equations 1-4 is varied in this section. It weighs service level costs and operational costs. $\beta = 0$ leads to neglecting operational costs, and the method fully tries to minimize delay, whereas the opposite happens for $\beta = 1$ (only optimizing on operational costs). We run simulation for $\beta \in \{0, \frac{1}{3}, 1\}$, the baseline scenario as well as both extreme cases. Figure 20 displays the obtained results. For $\beta = 0$ time KPIs are the lowest, as expected. For $\beta = 1$, the total driven distance is minimized, as expected. The service rate varies slightly between the different runs. The highest service rate is achieved for $\beta = 1/3$, which suggests that fully optimizing to minimize driven distance or delay leads the system into unfavourable states to serve future demand if the goal is to serve as many orders as possible.

Conclusion

In this paper, we introduce and formalize the FDP. The FDP is a variation of the SDDP, aiming to deliver orders in minutes. The proposed approach considers to pick up goods at multiple depots per order and allows vehicles to perform pre-empty depot returns, if beneficial. This enables a reduced average distance to customers' homes and more agile planning. In each step, orders are assigned to potential pick-up locations, followed by checking how they could be combined into potential trips. As many potential trips as possible are calculated for each vehicle, limited by predefined constraints on the total delivery times and vehicle capacity. Vehicles are assigned to the potential trips via solving an integer linear program.

The proposed method can handle large problem sizes. Extensive computational experiments simulating one day of service have been carried out. Looking at one scenario in detail, in which 10,000 orders are placed and 30 vehicles are available to serve those, a service rate of 95.19% was achieved, which represents an improvement of 20% over a greedy approach. The average delay accounts for 5 min 43 s and 8,973.8 km needed to be driven. Further, simulations showed the value of using and considering multiple depots and the value of performing pre-empty depot returns. A sensitivity study analyzed the varying influence of individual parameters on the obtained solution.

Future research could extend the proposed method to look ahead or to actively anticipate, such that the risk that the system gets into unfavorable states is reduced. Further, the possibility to plan for heterogeneous fleets of vehicles could be added. Additionally, the proposed approach is designed for on-demand deliveries exclusively. How to integrate already known orders is another future research question, such as the incorporation of stochastic information about the presence of potential orders. How many depots should be operated within a given area and how many of them should be considered per order are two further interesting questions for the future. Additionally, representing the operational environment as realistically as possible can strongly increase the expressiveness of found results. This includes but is not limited to modeling realistic traffic conditions, considering dynamic travel times and congestion, or accurately representing parking options in cities.

Notes

1. The rapid development in the area of autonomous delivery robots and autonomous driving increases the relevance of such routing algorithms. For example, Starship Technologies already completed their fourth million autonomous delivery using their developed robot solution [(Technologies 2023)]. It might become feasible to operate large fleets with moderate costs and without the inherent risks that the human riders currently face [(Amiri, Ferguson, and Razavi 2022; Christie and Ward 2019; Fielbaum et al. 2023; Zheng et al. 2019)]. Nevertheless, this work does not exclusively assume autonomous vehicles. It applies the same to human-driven ones.
2. One exception is [(Kronmueller, Fielbaum, and Alonso-Mora 2022)], which tackles routing for the FDP with heterogeneous vehicles. We exclude this work here as it is an extension of the work presented here.
3. The travel times are assumed to be static. They are determined as the real-world distance between the two locations divided by the constant speed of the vehicles. This assumption can be replaced by a more sophisticated approach, like live data from a tool like Google Maps or similar. Unfortunately, this is beyond the scope of the here presented work. Interested readers are pointed to works such as [(Musolino et al. 2013; Musolino, Polimeni, and Vitetta 2018; Polimeni and Vitetta 2013; Vitetta 2023)], which tackle problems with changing travel times.
4. We take this assumption for the sake of simplicity. However, it is straightforward how to extend our method if this isn't the case.
5. Size is only used to see if the maximum capacity of a vehicle is violated. Therefore this can be either the weight, the volume of the order or a combination of both.
6. Customers do not mind where their goods are picked up from.

Acknowledgments

This research was supported by Ahold Delhaize. All content represents the opinion of the author(s), which is not necessarily shared or endorsed by their respective employers and/or sponsors.

Disclosure statement

No potential conflict of interest was reported by the author(s).

ORCID

Maximilian Kronmüller http://orcid.org/0000-0002-5199-7942

References

Ackva, C., and M. Ulmer 2022. "Consistent Routing for Local Same-Day Delivery via Micro-Hubs." *Working Paper Series*.

Agatz, N., A. Erera, M. Savelsbergh, and X. Wang. 2012. "Optimization for Dynamic Ride-Sharing: A Review." *European Journal of Operational Research* 223 (2): 295–303. https://doi.org/10.1016/j.ejor.2012.05.028.

Albert Heijn Nieuws. 2022. "Albert heijn breidt samenwerking met deliveroo en thuisbezorgd.nl uit."

Alonso-Mora, J., S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus. 2017. "On-Demand High-Capacity Ride-Sharing via Dynamic Trip-Vehicle Assignment." *Proceedings of the National Academy of Sciences* 114 (3): 462–467. https://doi.org/10.1073/pnas.1611675114.

Amiri, A. M., M. R. Ferguson, and S. Razavi. 2022. "Adoption patterns of autonomous technologies in logistics: evidence for niagara region." *Transportation Letters* 14 (7): 685–696. https://doi.org/10.1080/19427867.2021.1923305.

Azi, N., M. Gendreau, and J.-Y. Potvin. 2012. "A Dynamic Vehicle Routing Problem with Multiple Delivery Routes." *Annals of Operations Research* 199 (1): 103–112. https://doi.org/10.1007/s10479-011-0991-3.

Bent, R., and P. Van Hentenryck. 2004. "Scenario-Based Planning for Partially Dynamic Vehicle Routing with Stochastic Customers." *Operations Research* 52 (6): 977–987. https://doi.org/10.1287/opre.1040.0124.

Bernardo, M., B. Du, and A. B. Matias. 2023. "Achieving Robustness in the Capacitated Vehicle Routing Problem with Stochastic Demands." *Transportation Letters* 15 (3): 254–268. https://doi.org/10.1080/19427867.2022.2049547.

Bernardo, M., B. Du, and J. Pannek. 2021. "A Simulation-Based Solution Approach for the Robust Capacitated Vehicle Routing Problem with Uncertain Demands." *Transportation Letters* 13 (9): 664–673. https://doi.org/10.1080/19427867.2020.1752448.

Cbinsights Research Briefs. 2021. "uber acquires cornershop".

Christie, N., and H. Ward. 2019. "The Health and Safety Risks for People Who Drive for Work in the Gig Economy." *Journal of Transport Health* 13:115–127. https://doi.org/10.1016/j.jth.2019.02.007.

Cordeau, J.-F., and G. Laporte. 2007. "The Dial-A-Ride Problem: Models and Algorithms." *Annals of Operations Research* 153 (1): 29–46. https://doi.org/10.1007/s10479-007-0170-8.

Côté, J.-F., T. A. de Queiroz, F. Gallesi, and M. Iori 2021. "Dynamic Optimization Algorithms for Same-Day Delivery Problems."

Farahani, R. Z., and M. Hekmatfar. 2009. *Facility Location: Concepts, Models, Algorithms and Case Studies*. Springer Science & Business Media. https://link.springer.com/book/10.1007/978-3-7908-2151-2.

Fielbaum, A., X. Bai, and J. Alonso-Mora. 2021. "On-Demand Ridesharing with Optimized Pick-Up and Drop-Off Walking Locations." *Transportation Research Part C: Emerging Technologies* 126:103061. https://doi.org/10.1016/j.trc.2021.103061.

Fielbaum, A., M. Kronmüller, and J. Alonso-Mora. 2021. "Anticipatory Routing Methods for an On-Demand Ridepooling Mobility System." *Transportation* 49 (6): 1921–1962. https://doi.org/10.1007/s11116-021-10232-1.

Fielbaum, A., F. Ruiz, G. Boccardo, D. Rubio, A. Tirachini, and J. Rosales-Salas. 2023. "The Job of Public Transport, Ride-Hailing and Delivery Drivers: Conditions During the COVID-19 Pandemic and Implications for a Post-Pandemic Future." *Travel Behaviour and Society* 31:63–77. https://doi.org/10.1016/j.tbs.2022.11.004.

Ghiani, G., E. Manni, A. Quaranta, and C. Triki. 2009. "Anticipatory Algorithms for Same-Day Courier Dispatching." *Transportation Research Part E: Logistics & Transportation Review* 45 (1): 96–106. https://doi.org/10.1016/j.tre.2008.08.003.

Hyland, M., F. Dandl, K. Bogenberger, and H. Mahmassani. 2020. "Integrating Demand Forecasts into the Operational Strategies of Shared Automated Vehicle Mobility Services: Spatial Resolution Impacts." *Transportation Letters* 12 (10): 671–676. https://doi.org/10.1080/19427867.2019.1691297.

Kantar. 2022. "Markt van flitsbezorging groeit onstuimig in nederland."

Khamis, A., A. Hussein, and A. Elmogy. (2015). Multi-robot Task Allocation: A Review of the State-of-the-Art. In A. Koubâa and J. Martínez-de Dios (edited by) *Cooperative Robots and Sensor Networks 2015. Studies in Computational Intelligence*, Vol. 604. Cham: Springer. https://doi.org/10.1007/978-3-319-18299-5_2.

Klapp, M. A., A. L. Erera, and A. Toriello. 2016. "The One-Dimensional Dynamic Dispatch Waves Problem." *Transportation Science* 52 (2): 402–415. https://doi.org/10.1287/trsc.2016.0682.

Klapp, M. A., A. L. Erera, and A. Toriello. 2018. "The Dynamic Dispatch Waves Problem for Same-Day Delivery." *European Journal of Operational Research* 271 (2): 519–534. https://doi.org/10.1016/j.ejor.2018.05.032.

Klapp, M. A., A. L. Erera, and A. Toriello. 2020. "Request acceptance in same-day delivery." *Transportation Research Part E: Logistics & Transportation Review* 143:102083. https://doi.org/10.1016/j.tre.2020.102083.

Kronmueller, M., A. Fielbaum, and J. Alonso-Mora 2021. "On-Demand Grocery Delivery from Multiple Local Stores with Autonomous Robots." *Proceedings of the International Symposium on Multi-Robot and Multi-Agent Systems, MRS 2021*, Cambridge, 29–37.

Kronmueller, M., A. Fielbaum, and J. Alonso-Mora 2022. "Routing of heterogeneous fleets for flash deliveries via vehicle group assignment." In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, Macau, 2286–2291.

Montoya-Torres, J. R., J. López Franco, S. Nieto Isaza, H. Felizzola Jiménez, and N. Herazo-Padilla. 2015. "A Literature Review on the Vehicle Routing Problem with Multiple Depots." *Computers Industrial Engineering* 79:115–129. https://doi.org/10.1016/j.cie.2014.10.029.

Mourad, A., J. Puchinger, and C. Chu. 2019. "A Survey of Models and Algorithms for Optimizing Shared Mobility." *Transportation Research Part B: Methodological* 123:323–346. https://doi.org/10.1016/j.trb.2019.02.003.

Musolino, G., A. Polimeni, C. Rindone, and A. Vitetta. 2013. "Travel Time Forecasting and Dynamic Routes Design for Emergency Vehicles." *Procedia - Social & Behavioral Sciences*.87:193–202. SIDT Scientific Seminar 2012. https://doi.org/10.1016/j.sbspro.2013.10.603.

Musolino, G., A. Polimeni, and A. Vitetta. 2018. "Freight Vehicle Routing with Reliable Link Travel Times: A Method Based on Network Fundamental Diagram." *Transportation Letters* 10 (3): 159–171. https://doi.org/10.1080/19427867.2016.1241040.

Narayanan, S., E. Chaniotakis, and C. Antoniou. 2020. "Shared Autonomous Vehicle Services: A Comprehensive Review." *Transportation Research Part C: Emerging Technologies* 111:255–293. https://doi.org/10.1016/j.trc.2019.12.008.

Polimeni, A., and A. Vitetta. 2013. "Optimising Waiting at Nodes in Time-Dependent Networks: Cost Functions and Applications." *Journal of Optimization Theory and Applications* 156 (3): 805–818. https://doi.org/10.1007/s10957-012-0121-7.

Ralphs, T. K., L. Kopman, W. R. Pulleyblank, and L. E. T. Trotter. 2003. "On the Capacitated Vehicle Routing Problem." *Mathematical Programming* 94 (2–3): 343–359. https://doi.org/10.1007/s10107-002-0323-0.

Reyes, D., A. Erera, M. Savelsbergh, S. Sahasrabudhe, and R. O'Neil. 2018. "The Meal Delivery Routing Problem." *Optimization Online*. https://optimization-online.org/2018/04/6571/.

Technologies, S. 2023. "Comapny Website of Starship Technologies."

Uber. 2022. "Uber Marketplace Matching." https://www.uber.com/us/en/marketplace/matching/.

Ulmer, M. W., J. C. Goodson, D. C. Mattfeld, and M. Hennig. 2019. "Offline–Online Approximate Dynamic Programming for Dynamic Vehicle Routing with Stochastic Requests." *Transportation Science* 53 (1): 185–202. https://doi.org/10.1287/trsc.2017.0767.

Ulmer, M. W., J. C. Goodson, D. C. Mattfeld, and B. W. Thomas. 2020. "On Modeling Stochastic Dynamic Vehicle Routing Problems." *EURO Journal on Transportation and Logistics* 9 (2): 100008. https://doi.org/10.1016/j.ejtl.2020.100008.

Ulmer, M. W., and S. Streng. 2019. "Same-Day Delivery with Pickup Stations and Autonomous Vehicles." *Computers Operations Research* 108:1–19. https://doi.org/10.1016/j.cor.2019.03.017.

Ulmer, M. W., B. W. Thomas, A. M. Campbell, and N. Woyak. 2021. "The Restaurant Meal Delivery Problem: Dynamic Pickup and Delivery with Deadlines and Random Ready Times." *Transportation Science* 55 (1): 75–100. https://doi.org/10.1287/trsc.2020.1000.

Ulmer, M. W., B. W. Thomas, and D. C. Mattfeld. 2019. "Preemptive Depot Returns for Dynamic Same-Day Delivery." *EURO Journal on Transportation and Logistics* 8 (4): 327–361. https://doi.org/10.1007/s13676-018-0124-0.

Vitetta, A. 2023. "The Importance of Modeling Path Choice Behavior in the Vehicle Routing Problem." *Algorithms* 16 (1): 47. https://doi.org/10.3390/a16010047.

Voccia, S. A., A. M. Campbell, and B. W. Thomas. 2017. "The Same-Day Delivery Problem for Online Purchases." *Transportation Science* 53 (1): 167–184. https://doi.org/10.1287/trsc.2016.0732.

Xue, G., and Z. Wang. 2023. "Order Acceptance and Scheduling in the Instant Delivery System." *Computers Industrial Engineering* 182:109395. https://doi.org/10.1016/j.cie.2023.109395.

Xu, H., P. Pu, and F. Duan. 2018. "A hybrid ant colony optimization for dynamic multidepot vehicle routing problem." *Discrete Dynamics in Nature & Society* 2018:1–10. https://doi.org/10.1155/2018/3624728.

Yildiz, B., and M. Savelsbergh. 2019. "Provably High-Quality Solutions for the Meal Delivery Routing Problem." *Transportation Science* 53 (5): 1372–1388. https://doi.org/10.1287/trsc.2018.0887.

Yu, B., N. Ma, W. Cai, T. Li, X. Yuan, and B. Yao. 2013. "Improved Ant Colony Optimisation for the Dynamic Multi-Depot Vehicle Routing Problem." *International Journal of Logistics: Research & Applications* 16 (2): 144–157. https://doi.org/10.1080/13675567.2013.810712.

Zheng, Y., Y. Ma, L. Guo, J. Cheng, and Y. Zhang. 2019. "Crash Involvement and Risky Riding Behaviors Among Delivery Riders in China: The Role of Working Conditions." *Transportation Research Record* 2673 (4): 1011–1022. https://doi.org/10.1177/0361198119841028.

Zhen, L., J. Wu, G. Laporte, and Z. Tan. 2023. "Heterogeneous instant delivery orders scheduling and routing problem." *Computers Operations Research* 157:106246. https://doi.org/10.1016/j.cor.2023.106246.

Appendix

A. Results

The precise numbers of all performance indices of all executed runs are shown in Table A1.

Table A1. Precise results of all performance indices of all experiments.

	Service rate [%]	Delivery time [s]	Delay [s]	Time on vehicle [s]	Waiting time [s]	Mean loaded orders	Total distance [km]
Base scenario	95.19	467.07	342.61	192.53	229.54	1.49	8973.8
Comparison							
Greedy	74.18	533.93	406.30	184.81	304.12	1.13	10693.17
1 depot per ord.	92.68	456.74	332.82	177.23	234.51	1.36	9094.94
No pre-empty	94.81	470.89	346.43	191.73	234.16	1.48	8975.48
Considered depots							
5 depots per ord.	94.95	472.08	347.54	198.28	228.80	1.53	9077.29
7 depots per ord.	95.25	467.12	342.66	195.60	226.52	1.52	9051.34
Total depots							
1 depot	75.22	569.656	332.055	337.055	187.602	1.95	9555.17
15 depots	92.59	495.94	357.07	214.50	236.44	1.59	9714.92
25 depots	95.67	459.92	341.06	184.55	230.37	1.45	8844.39
Reinserts							
3 reinserts	99.0	542.99	418.26	202.44	295.55	1.63	8803.1
Demand patterns							
9500 orders	97.56	458.64	333.60	187.37	226.27	1.42	9067.55
10500 orders	93.97	481.85	357.19	198.39	238.46	1.59	9050.93
Number of used vehicles							
25 vehicles	84.98	491.69	367.70	213.30	233.39	1.75	7469.91
35 vehicles	99.67	396.95	272.14	164.58	187.37	1.16	10121.33
Cost weight							
$\beta = 0$	94.96	464.15	339.74	191.38	227.76	1.48	9140.61
$\beta = 1$	94.13	510.35	385.76	217.56	247.79	1.65	8793.51