Optimizing Routing and Fleet Sizing for Flash Delivery Operations

Kronmüller, M.

DOI
10.4233/uuid:5f831793-2dbc-4b24-9d92-1441f2d8ba16

Publication date
2024

Document Version
Final published version

Citation (APA)
Kronmüller, M. (2024). *Optimizing Routing and Fleet Sizing for Flash Delivery Operations*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:5f831793-2dbc-4b24-9d92-1441f2d8ba16

Optimizing Routing and Fleet Sizing for Flash Delivery Operations

Optimizing Routing and Fleet Sizing for Flash Delivery Operations

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus prof. dr. ir. T.H.J.J. van der Hagen
chair of the Board for Doctorates
to be defended publicly on
Wednesday 31 January 2024 at 12:30 o'clock

by

Maximilian KRONMÜLLER

Master of Science in Applied and Engineering Physics,
Technical University Munich, Germany
born in Stuttgart, Germany

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus	chairperson
Prof. dr. R. Babuska	Delft University of Technology, promotor
Dr. J. Alonso-Mora	Delft University of Technology, promotor
Independent members:	
Prof. dr. ir. L.A. Tavasszy	Delft University of Technology
Prof. dr. ir. K.I. Aardal	Delft University of Technology
Prof. dr. ir. M.B.M. de Koster	Erasmus University Rotterdam
Prof. dr.-ing. K. Bogenberger	Technical University of Munich, Germany
Dr. A.S. Fielbaum Schnitzler	University of Sydney, Australia

Printed by: Ridderprint

Cover by: Maximilian Kronmüller

ISBN 978-94-6384-533-5

An electronic version of this dissertation is available at
http://repository.tudelft.nl/.

Mit dem Wissen wächst der Zweifel.
Johann Wolfgang von Goethe

Doubt grows with knowledge.
Johann Wolfgang von Goethe

Contents

Summary

In recent years, *Flash Delivery* services have gained great popularity. Flash Delivery is a service where goods of daily need can be ordered on-demand and subsequently are delivered to the customer within a short time window, for example, in the next ten minutes. Operational efficiency and cost management are vital for sustainability in this competitive landscape, especially in the long term. To this end, this thesis aims to improve operational planning for Flash Delivery Operations. It focuses on two fundamental questions critical for the success of Flash Deliveries: the associated Vehicle Routing Problem and the associated Fleet Sizing Problem. The Vehicle Routing Problem aims to determine how to best utilize a given fleet of vehicles to deliver the requested orders efficiently, while the Fleet Sizing Problem involves finding the optimal number of vehicles required to serve the given demand. The primary objective of this dissertation is to provide algorithmic contributions, specifically focusing on optimizing vehicle routing and fleet sizing for Flash Delivery services.

First, the Flash Delivery Problem is formally defined and modeled as a Markov Decision Process. This serves as the basis for the dissertation's research and subsequent investigations. The thesis then proposes a novel routing algorithm for Flash Deliveries from multiple depots, which effectively handles multiple depots for order pick-up and dynamically determines the optimal depot for each order. The depots are distributed within the city, for example, using existing stores, this differs from other logistical processes using large warehouses outside of the city. Additionally, this approach allows vehicles to visit depots to load additional orders before distributing their loaded ones, resulting in more agile planning. The scalability of this method is demonstrated in scenarios involving thousands of orders and tens of vehicles.

The proposed routing method is then extended to accommodate heterogeneous vehicles and heterogeneous modes of transportation. Experiments using a fleet featuring trucks and drones demonstrate that this approach serves more orders while requiring less total traveled distance compared to a state-of-the-art method for heterogeneous vehicles. The effects of fleet size and fleet composition between drones and trucks are also examined. More drones were able to deliver more requests at the cost of an increase in traveled distance.

The Fleet Sizing Problem represents the second major challenge addressed in this dissertation. The balance between having too many vehicles, which can be very expensive, and having too few, which leads to unmet promises and undelivered orders, is crucial for operational success. Typically, the Fleet Sizing Problem involves a fixed set of tasks with no flexibility in their execution. However, this thesis introduces a

novel problem, adding flexibility in time through the allowance of slight delays in individual transportation tasks. We propose modeling and solving the novel problem as a Mixed Integer Linear Program. By incorporating this flexibility, the problem opens up a broader trade-off space between the required number of agents, traffic, and added delays. As a result, fleet sizes can be significantly decreased. To illustrate the practical application of this algorithm, a case study involving taxi rides in Manhattan is presented.

To conclude this thesis, fleet sizing is combined with the previously proposed routing methods for Flash Delivery, resulting in a novel approach. Our method groups individual delivery requests and generates optimized operational plans using a variation of the earlier proposed routing techniques. These plans are then used for fleet sizing. To assess the effectiveness of our approach, we compare it against applying routing and fleet sizing separately. The results clearly demonstrate the value of our proposed method. Our experimental analysis is based on a real-world dataset provided by a Dutch retailer, allowing us to gain valuable insights into the design of Flash Delivery operations.

In summary, this thesis makes significant contributions to the operational optimization of Flash Delivery services by addressing key challenges in vehicle routing and fleet sizing. We propose novel methods to improve efficiency and effectiveness in planning Flash Delivery operations.

Samenvatting

In de afgelopen jaren hebben *Flash Delivery* diensten, of in het Nederlands *Flitz Bezorging* diensten, enorm aan populariteit gewonnen. Flash Delivery is een dienst waarbij goederen voor dagelijks gebruik op aanvraag kunnen worden besteld en vervolgens binnen een kort tijdsbestek, bijvoorbeeld in de volgende tien minuten, aan de klant worden bezorgd. Operationele efficiëntie en kostenbeheer zijn van vitaal belang voor duurzaamheid in dit competitieve landschap, vooral op de lange termijn. Met dat doel voor ogen heeft deze dissertatie tot doel de operationele planning voor Flash Delivery operaties te verbeteren. Het richt zich op twee fundamentele vragen die cruciaal zijn voor het succes van Flash Delivery leveringen: het bijbehorende Vehicle Routing Problem (Voertuigrouteringsprobleem) en het bijbehorende Fleet Sizing Problem (Probleem van de Vlootomvang). Het Vehicle Routing Problem heeft als doel te bepalen hoe een gegeven vloot van voertuigen het meest efficiënt kan worden ingezet om de gevraagde bestellingen efficiënt te bezorgen. Het Fleet Sizing Problem zich bezighoudt met het vinden van het optimale aantal voertuigen dat nodig is om aan de gegeven vraag te voldoen. Het primaire doel van deze dissertatie is het ontwikkelen van algoritmes, met specifieke nadruk op het optimaliseren van voertuigrouting en vlootomvang voor Flash Delivery diensten.

Ten eerste wordt het Flash Delivery Probleem formeel gedefinieerd en gemodelleerd als een Markov Beslissingsproces. Dit dient als basis voor het onderzoek van de dissertatie en de daaropvolgende onderzoeken. De dissertatie stelt vervolgens een nieuw routeringsalgoritme voor Flash Delivery Leveringen voor vanuit meerdere depots, dat op doeltreffende wijze meerdere depots voor het ophalen van bestellingen beheert en dynamisch het optimale depot voor elke bestelling bepaalt. De depots zijn verspreid binnen de stad, bijvoorbeeld gebruikmakend van bestaande winkels, in tegenstelling tot andere logistieke processen met grote magazijnen buiten de stad. Daarnaast stelt deze aanpak voertuigen in staat om depots te bezoeken om extra bestellingen op te halen voordat ze hun geladen bestellingen distribueren, wat resulteert in meer flexibele planning. De schaalbaarheid van deze methode wordt gedemonstreerd in scenario's met duizenden bestellingen en tientallen voertuigen.

De voorgestelde routeringsmethode wordt vervolgens uitgebreid om rekening te houden met heterogene voertuigen en vervoerswijzen. accommoderen. Experimenten met een vloot bestaande uit vrachtwagens en drones tonen aan dat deze aanpak meer bestellingen kan verwerken terwijl er minder totale afgelegde afstand nodig is in vergelijking met een state-of-the-art methode voor heterogene voertuigen. De effecten van de grootte van de vloot en de verhouding tussen drones en vrachtwagens worden ook onderzocht. Meer drones konden meer verzoeken afhandelen ten koste van een toename in afgelegde afstand.

Het Fleet Sizing Problem vertegenwoordigt de tweede grote uitdaging die in deze dissertatie wordt aangepakt. Het evenwicht tussen te veel voertuigen hebben, wat erg duur kan zijn, en te weinig voertuigen hebben, wat leidt tot niet nagekomen beloften en niet-afgeleverde bestellingen, is cruciaal voor operationeel succes. Typisch betreft het Fleet Sizing Problem een vaststaande reeks taken zonder flexibiliteit in hun uitvoering. Deze dissertatie introduceert echter een nieuw probleem door flexibiliteit in de tijd toe te staan met lichte vertragingen in individuele transporttaken. We stellen voor om dit nieuwe probleem te modelleren en op te lossen als een Gemengd Geheel Lineair Programmeringsprobleem. Door deze flexibiliteit mee te nemen, opent het probleem een breder afwegingsgebied tussen het benodigde aantal agenten, verkeer en toegevoegde vertragingen. Als gevolg daarvan kunnen vlootgroottes aanzienlijk worden verkleind. Om de praktische toepassing van dit algoritme te illustreren, wordt een casestudy gepresenteerd met taxiritten in Manhattan.

Ter afronding van deze dissertatie wordt de dimensionering van de vloot gecombineerd met de eerder voorgestelde routeringsmethoden voor Flash Delivery, wat resulteert in een nieuwe aanpak. Onze methode groepeert individuele leveringsaanvragen en genereert geoptimaliseerde operationele plannen met behulp van een variatie van de eerder voorgestelde routeringstechnieken. Deze plannen worden vervolgens gebruikt voor de dimensionering van de vloot. Om de effectiviteit van onze aanpak te beoordelen, vergelijken we deze met het afzonderlijk toepassen van routering en dimensionering van de vloot. De resultaten tonen duidelijk de waarde van onze voorgestelde methode aan. Onze experimentele analyse is gebaseerd op een dataset uit de praktijk die is verstrekt door een Nederlandse retailer, wat ons waardevolle inzichten geeft in het ontwerp van Flash Delivery-operaties.

Samenvattend levert deze dissertatie aanzienlijke bijdragen aan de operationele optimalisatie van Flash Delivery diensten door belangrijke uitdagingen in voertuigrouting en vlootomvang aan te pakken. We stellen nieuwe methoden voor om efficiëntie en effectiviteit te verbeteren bij het plannen van Flash Delivery operaties.

Zusammenfassung

In den letzten Jahren haben sich Angebote von *Flash Delivery*, oder zu Deutsch *Blitz-Lieferungen*, großer Beliebtheit erfreut. Flash Delivery ist ein Service, bei dem Güter des täglichen Bedarfs auf Abruf bestellt werden können und anschließend innerhalb eines kurzen Zeitraums an den Kunden geliefert werden, beispielsweise innerhalb der nächsten zehn Minuten. Operative Effizienz und Kostenmanagement sind in dieser wettbewerbsintensiven Landschaft, insbesondere langfristig, von entscheidender Bedeutung für die Nachhaltigkeit. Zu diesem Zweck zielt diese Dissertation darauf ab, die operative Planung für Flash Delivery Operationen zu verbessern. Wir konzentrieren uns auf zwei grundlegende Fragen, die für den Erfolg von Flash Deliveries entscheidend sind: das zugehörige Vehicle Routing Problem und das zugehörige Fleet Sizing Problem. Das Vehicle Routing Problem zielt darauf ab, wie eine gegebene Fahrzeugflotte am effizientesten genutzt werden kann, um die angeforderten Bestellungen effizient zuzustellen, während das Fleet Sizing Problem die optimale Anzahl der benötigten Fahrzeuge für die gegebene Nachfrage ermittelt. Das Hauptziel dieser Dissertation besteht darin, algorithmische Beiträge zu liefern, die sich speziell auf die Optimierung der Fahrzeugrouten und die Größenanpassung der Flotte für Flash Delivery Dienste konzentrieren.

Zunächst wird das Flash Delivery Problem formell definiert und als Markow Entscheidungsprozess modelliert. Dies dient als Grundlage für die anschließenden Untersuchungen in dieser Dissertation. Die Dissertation schlägt einen neuen Routenalgorithmus für Flash Deliveries von mehreren Depots vor, der effektiv mit mehreren Depots für die Abholung von Bestellungen umgeht und dynamisch das optimale Depot für jede Bestellung ermittelt. Die Depots befinden sich innerhalb der Stadt, beispielsweise können bestehende Geschäfte genutzt werden, dies unterscheidet sich von anderen logistischen Abläufen, die große Lagerhäuser außerhalb der Stadt nutzen. Darüber hinaus ermöglicht dieser Ansatz den Fahrzeugen, Depots zu besuchen, um zusätzliche Bestellungen abzuholen, bevor sie ihre geladenen Bestellungen verteilen, was zu einer flexibleren Planung führt. Die Skalierbarkeit dieser Methode wird in Szenarien mit Tausenden von Bestellungen und Dutzenden von Fahrzeugen demonstriert.

Die vorgeschlagene Routenmethode wird dann erweitert, um heterogene Fahrzeuge und unterschiedliche Transportarten handhaben zu können. Experimente mit einer Flotte von Lastwagen und Drohnen zeigen, dass dieser Ansatz mehr Bestellungen bedient und gleichzeitig weniger Gesamtfahrstrecke erfordert im Vergleich zu einer State-of-the-Art-Methode für heterogene Fahrzeuge. Die Auswirkungen der Flottengröße und der Flottenzusammensetzung zwischen Drohnen und Lastwagen werden ebenfalls untersucht. Mehr Drohnen konnten mehr Bestellungen bedienen, was mit

einer Zunahme der zurückgelegten Entfernung einherging.

Das Fleet Sizing Problem stellt die zweite große Herausforderung dar, die in dieser Dissertation behandelt wird. Das Gleichgewicht zwischen zu vielen Fahrzeugen, was sehr teuer sein kann, und zu wenigen, was zu nicht erfüllten Versprechen und nicht zugestellten Bestellungen führt, ist entscheidend für den operationellen Erfolg. Normalerweise betrachtet das Fleet Sizing Problem eine feste Anzahl von Aufgaben ohne Flexibilität in ihrer Ausführung. Diese Dissertation führt jedoch ein neues Problem ein, indem sie Flexibilität in der Zeit durch die Zulassung geringfügiger Verzögerungen in einzelnen Transportaufgaben ermöglicht. Wir schlagen vor, dieses neue Problem als gemischt-ganzzahliges lineares Programm zu modellieren und zu lösen. Durch die Integration dieser Flexibilität eröffnet das Problem einen breiteren Abwägungsraum zwischen der erforderlichen Anzahl von Fahrzeugen, dem Verkehr und zusätzlichen Verzögerungen. Infolgedessen können die Flottengrößen erheblich reduziert werden. Zur Veranschaulichung der praktischen Anwendung dieses Algorithmus wird eine Fallstudie mit Taxifahrten in Manhattan präsentiert.

Um diese Dissertation abzuschließen, wird die Größenanpassung der Flotte mit den zuvor vorgeschlagenen Routenmethoden für Flash Delivery kombiniert, was zu einem neuen Ansatz führt. Unsere Methode gruppiert einzelne Lieferanfragen und generiert optimierte operative Pläne unter Verwendung einer Variation der zuvor vorgeschlagenen Routentechniken. Diese Pläne werden dann zur Größenanpassung der Flotte verwendet. Um die Wirksamkeit unseres Ansatzes zu bewerten, vergleichen wir ihn mit der getrennten Anwendung von Routenplanung und Größenanpassung der Flotte. Die Ergebnisse zeigen deutlich den Wert unserer vorgeschlagenen Methode auf. Unsere experimentelle Analyse basiert auf einem realen Datensatz, der von einem niederländischen Einzelhändler bereitgestellt wurde, was uns wertvolle Einblicke in die Gestaltung von Flash Delivery Operationen ermöglicht.

Zusammenfassend leistet diese Dissertation erhebliche Beiträge zur operationellen Optimierung von Flash Delivery Diensten, indem sie Schlüsselherausforderungen bei der Fahrzeugroutenplanung und der Größenanpassung der Flotte angeht. Wir schlagen neue Methoden vor, um die Effizienz und Effektivität bei der Planung von Flash Delivery Operationen zu verbessern.

Acronyms

DMDVRP Dynamic Multi-Depot Vehicle Routing Problem

DVRP Dynamic Vehicle Routing Problem

FDP Flash Delivery Problem

FSD Fleet Sizing with Delays

FSP Fleet Sizing Problem

HVGA Heterogeneous Vehicle-Group Assignment

ILP Integer Linear Program

KPIs Key Performance Indicators

MDP Markov Decision Process

MILP Mixed Integer Linear Program

SDD Same-Day Delivery

SDDP Same-Day Delivery Problem

VGA Vehicle-Group Assignment

VRP Vehicle Routing Problem

1 Introduction

1

1.1. Motivation

Flash Delivery services have gained considerable popularity in recent years. These services cater to the immediate needs of customers by swiftly fulfilling their requests for daily essential products. With Flash Delivery services, customers can now order online and receive their desired products at their front door in a matter of minutes. Everyday inconveniences, such as missing an ingredient for a planned meal or running out of snacks and drinks, are commonplace in today's society. Flash Delivery services have become a luxury that addresses these challenges, allowing people to have their groceries and products of daily need delivered to their doorstep within minutes, eliminating the need for a trip to the nearest supermarket. As a result, these services' convenience and time-saving aspects have garnered widespread appreciation. The increasing significance and positive reception of Flash Delivery services are evident from the emergence of numerous young startups offering such solutions and the substantial turnover these companies have achieved [1]. Startups like Gorillas, Flink, and Getir are prime examples of enterprises that have entered this market and gained considerable traction. In 2021 alone, the Dutch population spent a staggering 40 million euros per month on Flash Deliveries [2]. On the other side, during 2023, this trend slowed down, leaving the future of Flash Delivery services uncertain.

The significance of Flash Delivery services is further strengthened by potential synergies with other technological advancements. Autonomous delivery robots and autonomous driving technologies are emerging as game-changers in the logistics and transportation industries. These developments have the potential to accelerate existing processes in the Flash Delivery domain. One notable example is Starship Technologies, whose robot solutions have successfully completed millions of autonomous deliveries, showcasing the feasibility and efficiency of such systems [3]. As these autonomous robots become more sophisticated and widespread, they can significantly enhance last-mile delivery capabilities, offering even faster and more cost-effective solutions. Similarly, the advancement of autonomous driving technologies could eliminate the need for human drivers and associated risks [4, 5, 6, 7]. Furthermore, the concept of fast individual deliveries by air, often explored through drone technologies, could open up new avenues for Flash Delivery services, particularly for high-value items and in urban areas with complex traffic patterns.

At the same time, Flash Delivery services pose a significant challenge to traditional retailers operating brick-and-mortar stores, as they offer customers a more convenient and time-saving alternative that eliminates the need to visit physical stores and carry products home. Established retailers have recognized the potential of this sector and are actively exploring opportunities to expand their service portfolio. A notable example is Albert Heijn, a prominent Dutch supermarket brand, which has partnered with Thuisbezorgd, a Dutch food delivery company, to provide grocery delivery services within Amsterdam [8]. Similarly, the strategic collaboration between Cornershop and Uber exemplifies this change [9]. However, traditional retailers also have certain potential advantages in the face of this disruption. For instance, in Amsterdam, there is a ban on opening new dark stores [10]. Dark stores are dedicated pick-up locations for online orders. This limitation on dark stores

presents a unique opportunity for brick-and-mortar retailers to capitalize on their existing physical infrastructure and utilize their stores as depots for Flash Delivery operations.

From a more technical perspective, Flash Deliveries are often described with other terms like logistics, last-mile, on-demand, or personal. The term *Logistics* refers to the efficient flow of goods within a value chain, with a specific focus on the movement of goods between different pick-up and drop-off locations in the context of Flash Delivery. The term *Last-mile* highlights the goal of executed deliveries, which involves reaching individual customers at their desired destinations. This term emphasizes the critical final segment of the delivery process. Additionally, Flash Delivery services are characterized as *On-demand*, meaning there is a short time gap between placing an order and the final delivery of goods. During the operation, the orders to fulfill change with time, and as such, the situation to plan for. This changing real-time nature of the service is often referred to as a dynamic problem. Flash Delivery services are *personal*, meaning each customer individually specifies a relatively small assortment of ordered goods.

Flash Delivery services hold great potential from a scientific perspective as they have not been extensively researched because they just emerged in the last years. Their growth is driven by high demand and intense competition for market share. Operational efficiency and cost management are vital for sustainability in this competitive landscape, especially in the long term. The delivery process and the necessary resources to support it are major cost drivers within the Flash Delivery industry. The limited exploration of routing and fleet design for Flash Delivery services, specifically including the option of batching orders together, is the identified research gap that this thesis addresses. These critical components have not been extensively studied, providing promising opportunities for further research and optimization. Despite presenting fascinating research opportunities, the practical implications of resolving these problems go far beyond academic interests. By optimizing routing and fleet design, Flash Delivery companies can achieve higher operational efficiency, cost-effectiveness, and, ultimately, customer satisfaction. Consequently, investing efforts in understanding and solving these challenges would be highly valuable, also leading to more sustainable and successful Flash Delivery services in the future.

This thesis specifically addresses two fundamental questions crucial for the execution of Flash Deliveries: the corresponding Vehicle Routing Problem (VRP) and the corresponding Fleet Sizing Problem (FSP). In general, the VRP asks the question of how a given fleet of vehicles is used best to deliver demand according to the constraints of the operation at hand. The FSP raises the question of how many vehicles are needed to serve some given demand best, given a rule on how vehicles are used. As such, a dependency of the two problems emerges, which is illustrated in Figure 1.1. Assuming a fleet of vehicles, for example, seven autonomous trucks, the VRP for Flash Deliveries is to decide on their deployment to efficiently meet the existing demand. This optimization leads to operational plans for each of these seven vehicles. Achieving this requires a routing methodology. Using such a method as a starting point, the FSP can be asked: “How many vehicles are needed to serve the demand?”. Resulting in a number of required vehicles, which then again was

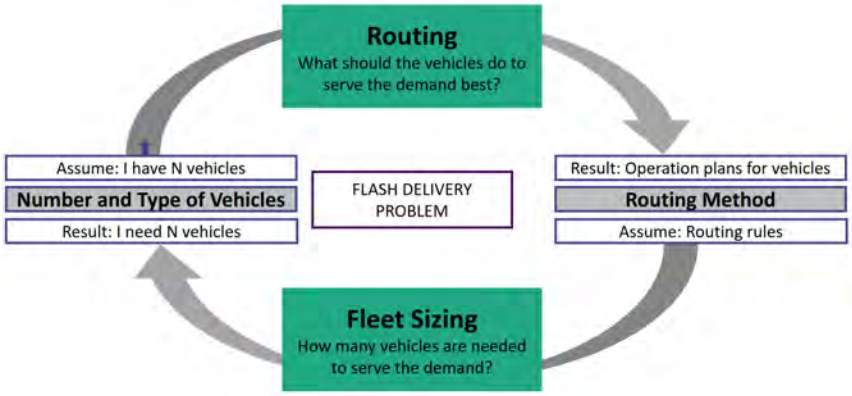


Figure 1.1: The VRP and the FSP are interrelated. VRP optimizes the usage of a given fleet, while the FSP determines the optimal number of vehicles under specific rules.

the starting point for the according VRP. This interplay is why both problems are tackled within this dissertation and shapes the structure of the here presented work.

Finding a feasible solution, no matter the costs, for both of these questions is a simple task. Simple approaches may not be the most efficient, such as dispatching riders sequentially to new orders, according to the motto first in, first out, or sizing the fleet based solely on past experience. More sophisticated approaches could involve grouping orders for simultaneous delivery when their locations are close, improving efficiency. Additionally, striking the right balance in fleet size is essential. Operational inefficiencies and unnecessary costs may arise if the number of riders is excessive. Conversely, too few riders can lead to an inability to meet promised service levels. Addressing these complexities effectively is crucial for achieving high-quality, cost-effective, and reliable Flash Delivery services. Other effects of being efficient means fewer vehicles crowding cities and less traffic causing pollution and, at the same time, being able to offer the operations sustainably. By developing advanced algorithms and strategies, this thesis aims to optimize on-demand last-mile logistics, providing valuable insights for the Flash Delivery industry and contributing to the growth and success of this rapidly expanding sector.

For the remainder of this introduction chapter, the goals of this dissertation are specified and divided into manageable sub-questions in Section 1.2. To address these objectives, we propose several algorithms from the field of combinatorial optimization, specifically modeling the problems as Integer Linear Program (ILP) or Mixed Integer Linear Program (MILP). These innovative algorithms draw inspiration from ride pooling and ridesharing concepts and apply them to the logistics domain. Further details on the proposed methodology are presented in Section 1.3. The main scientific contributions of this work are summarized in Section 1.4, highlighting the

novel insights and advancements made in the area. Section 1.5 introduces collaborative works beyond the work presented in this thesis. Lastly, the structure of the remaining dissertation is outlined in Section 1.6, providing readers with a clear roadmap of the upcoming chapters and their content.

1.2. Research Questions

To tackle the above-motivated aim of this thesis, we formulate the main research question and divide it into smaller sub-questions. These sub-questions are tackled individually, each contributing to answering the main question, which we pose as follows:

> **Main-Research Question:**
> How can the planning for Flash Delivery operations regarding vehicle routing and fleet sizing be accomplished efficiently and effectively?

We decided to tackle vehicle routing first because assuming a method for how to use each vehicle is a harder and stronger assumption than assuming a number of vehicles to be used. For the routing problem for Flash Delivery services, we assume as input a predefined fleet and an operation to be carried out. What each of the agents should do, in which order, and when needs to be determined. To emulate the situation of a traditional retailer, we consider multiple depots as pick-up locations, which leads to the first sub-question:

Sub-Question 1: How can we efficiently plan vehicle routes for Flash Delivery services from multiple stores?

A limitation of the proposed approach is the usage of a single vehicle type with identical characteristics. To generalize the found results from a single type of vehicle to multiple ones, even including different modes of transportation, like trucks following the road network and drones flying in the air, we ask the following sub-question:

Sub-Question 2: How can we generalize the previously developed routing approach to heterogeneous modes of transportation?

Answering the previous two questions provides a method to determine what each vehicle of the fleet does, given a concrete situation. Nevertheless, how many vehicles the fleet should consist of can not yet be tackled other than by assuming different fleets and testing using the developed tools. Under the assumption that vehicles follow plans as determined by the previously proposed method, we ask the question of fleet sizing as follows:

Sub-Question 3: How to optimize required fleet sizes for on-demand last-mile delivery operations?

Last, we want to bring the insights from a theoretical realm closer to an applied operation. To do so, we apply the proposed methods to a dataset resembling a Flash Delivery operation extrapolated from traditional shopping behavior in supermarkets. We ask:

Sub-Question 4: To what extent is Flash Delivery applicable to a traditional retailer?

To study this question, we apply the previously proposed methods together to a real-life dataset provided by a traditional retailer.

1.3. Approach

At a high level, the algorithms developed in this thesis's scope belong to the combinatorial optimization field. Combinatorial optimization involves searching for optimal solutions within a finite and discrete set of possibilities. However, these problems can quickly become computationally challenging, making exhaustive searches impractical. More specifically, we model our problems as ILP or MILP. These problems are linear in their objective function and constraints. For ILP, the variables are discrete. The problems are classified as MILP if some variables are continuous.

We build on existing methods that focus on the transportation of people rather than goods, like dial-a-ride services or taxi rides. While Flash Deliveries or on-demand last-mile logistics and on-demand transportation of people share similarities, there are three significant differences. First, taxi rides specify the pick-up and drop-off location rather than a goal location only. For Flash Delivery, customers do not mind at which location their goods are picked up. Second, people care about the transportation process itself, which is not the case for goods. For example, the time till pick-up that is considered for pooled taxi rides is negligible for logistics. Third, in Flash Deliveries, the delay of individual orders is less important as long as they are delivered within the promised times. In contrast, passengers are highly delay sensitive.

More specifically, our proposed routing algorithm is inspired by a routing algorithm for pooled taxi rides [11]. In essence, [11] proposes an approach that assigns each vehicle a plan to follow in the near future based on the current problem state. This is achieved through two steps: first, potential plans are determined for each vehicle, and second, the plans to be executed are selected by solving an optimization problem.

In our work regarding fleet design, we build upon the concept of *chaining*, which was also originally introduced for taxi rides [12]. The core idea behind chaining is to relocate vehicles to the start location of new transportation tasks after completing a previous one. If this relocation can be accomplished within the required time constraints, the same vehicle can be reused, reducing the need for additional vehicles. By applying this method throughout an entire operation, chaining can determine the number of vehicles required for efficient service.

1.4. Contribution Statement

This dissertation aims to provide algorithmic contributions for Flash Delivery services, with a specific focus on optimizing vehicle routing and fleet sizing. The contributions of this thesis are as follows:

- **A formal problem formulation of the Flash Delivery Problem**. We formally define and model the Flash Delivery Problem as a Markov Decision Process.
- **An Algorithm for Online Flash Delivery from Multiple Depots**. The proposed method handles multiple depots for order pick-up and endogenously determines the optimal depot for each order. Additionally, vehicles have the flexibility to visit a depot to load additional orders before distributing their already loaded ones if beneficial. The algorithm's scalability is demonstrated through successful application in scenarios with thousands of orders and tens of vehicles.
- **A Flash Delivery Routing Method for Heterogeneous Fleets**: Building upon the previously proposed routing method, we propose an extension that allows for routing heterogeneous vehicles and accommodating various modes of transportation. This approach considers the specific characteristics of each vehicle and plans routes accordingly, optimizing the overall delivery process.
- **A Fleet Sizing Algorithm for On-Demand Operations Allowing for Delays**: We propose a novel problem and algorithm for fleet sizing, which allows delaying individual transportation tasks slightly. Thus, the trade-off space between the required number of vehicles, traffic, and newly introduced delay can be enlarged. This allows previous minimal fleets to be decreased. The proposed approach is based on chaining and models and solves the problem as an MILP.
- **A Fleet Sizing Algorithm including Sophisticated Routing**: We propose a novel combination of methods enabling fleet sizing, including vehicle routing for Flash Delivery operations from multiple stores. Combining a sophisticated routing method with fleet sizing enables better quality solutions than applying either of them alone.
- **Insights into Designing Flash Delivery Operations at Large Scale**: Through a comprehensive case study conducted in Amsterdam, we offer valuable insights into the design and implementation of Flash Delivery operations. By exploring a real-world scenario, we aim to shed light on the challenges and opportunities of large-scale Flash Delivery services.

Together, these contributions advance the field of on-demand last-mile logistics, providing efficient and effective solutions for vehicle routing and fleet sizing, ultimately contributing to the improvement of modern delivery services and transportation systems.

1.5. Research Beyond this Thesis

Within the scope of this Ph.D. project, several additional research projects were conducted in collaboration with colleagues and supervised students. Although not the primary focus of this dissertation, these projects significantly contributed to the dissertations related broader research landscape. In the following, we briefly provide an overview of these projects:

- **Group-Based Distributed Auction Algorithms for Multi-Robot Task Assignment**: We investigated dynamic distributed multi-robot task assignment, aiming to find efficient methods for allocating tasks to multiple robots [13]. We propose two group-based distributed auction algorithms guided by auction principles. We show that the designed algorithms can compete with a centralized ILP.

- **Request Anticipation in Dynamic Vehicle Routing Problems**: We conducted research on the integration of anticipation to enhance the limitations of fully myopic routing approaches [14, 15, 16]. In [14], a novel anticipatory order insertion technique is proposed, which predicts future requests based on clustered historical data. These predictions are then integrated into a dynamic vehicle routing solver using heuristics and an adaptive large neighborhood search. In contrast, [15] introduces alternative methods to modify individual trip costs based on the current problem state to incorporate the effect on future states. The adapted costs influence the calculation and selection of vehicle plans. Multiple ways to alter the cost terms are investigated. In the scope of his master thesis [16], Stavya Bhatia builds upon the insights gained from [15] and applies these approaches to the context of logistics and Flash Deliveries.

- **Fleet Sizing in On-Demand Delivery Services**: Fleet sizing was the focus of the master thesis of Cilia Claij [17], specifically examining the efficient selection of pick-up locations from multiple stores in the FSP. This research project aimed to optimize the fleet size while considering the most suitable pick-up points to enhance operational efficiency.

- **Learning-Based Methods for Operations Research**: To assess the viability of learning-based methods within this dissertation, we investigated the application of transformer networks in tackling the classical operations research problem known as the knapsack problem [18]. Through this investigation, we aimed to explore the possibilities and limitations of utilizing such methods in optimization tasks.

For detailed insights into each project, please refer to the corresponding publications.

1.6. Outline

A graphical outline of this thesis is visualized in Figure 1.2.

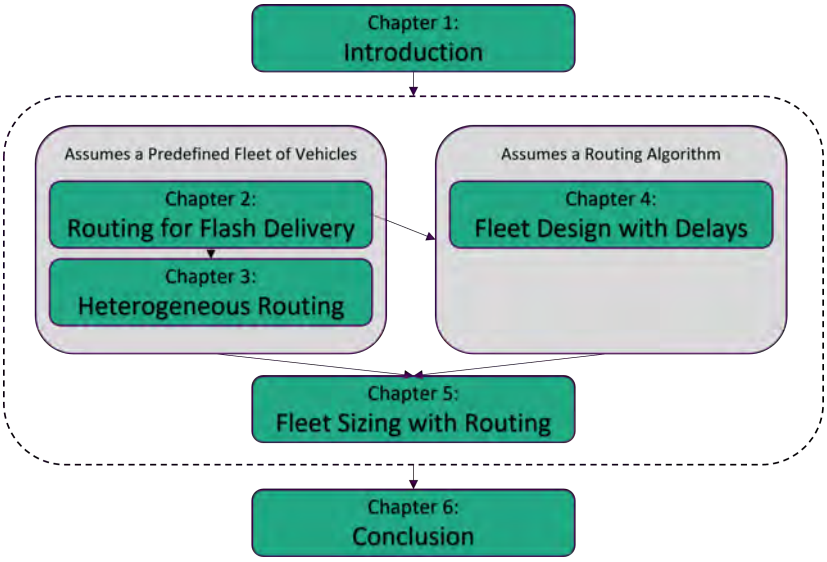


Figure 1.2: Graphical outline of the structure of this dissertation. Chapters 2 and 3 explore routing algorithms, while Chapter 4 concerns fleet design algorithms. A case study is presented in Chapter 5. Chapter 6 concludes this thesis.

This introductory chapter, **Chapter 1**, provides the motivation behind the research, presents the main and sub-research questions, and outlines the overall approach. It also highlights the contributions of the thesis and briefly introduces collaborative works.
In **Chapter 2**, a routing method for the Flash Delivery Problem (FDP) is proposed. The method considers multiple depots and adapts plans in real-time to optimize the delivery process.
Chapter 3 generalizes the routing method of Chapter 2 to accommodate multi-modal transportation, encompassing various vehicle types such as trucks and drones.
In **Chapter 4**, fleet design for on-demand operations, proposing a methodology that allows to delay individual tasks, is investigated.
Chapter 5 combines fleet sizing and routing for Flash Deliveries. A case study in Amsterdam is conducted, providing practical applications and insights into real-world Flash Delivery operations.
Finally, **Chapter 6** concludes the thesis, summarizing the key findings and offering potential directions for future research.

2

Online Flash Delivery from Multiple Depots

This chapter formally introduces the Flash Delivery Problem and presents a novel routing method for it. The proposed approach allows for the consideration of multiple pick-up locations per order and provides the flexibility to adapt the vehicle's route before being empty. This method serves as a central cornerstone of this thesis, forming the basis for further extensions and reuse in multiple contexts.

This chapter is based on:

- M. Kronmueller, A. Fielbaum, J. Alonso-Mora, *"On-Demand Grocery Delivery From Multiple Local Stores With Autonomous Robots"*, in Proceedings of the International Symposium on Multi-Robot and Multi-Agent Systems (MRS), pp.29-37, 2021 [19]
- M. Kronmueller, A. Fielbaum, J. Alonso-Mora, *"Online Flash Delivery from Multiple Depots"*, in Transportation Letters, pp.1-17, 2023 [20]

Abstract

In recent years, delivery companies that deliver orders minutes after being placed have emerged in many countries. In this chapter, we study the underlying routing problem, which we call Flash Delivery Problem. The Flash Delivery Problem is a variation of the Same-Day Delivery Problem requiring that orders are delivered minutes after being placed. The new problem is formally introduced, and a novel solution approach with two novel features is proposed: i) Multiple depots, optimizing where to pick up every order, ii) Allowing vehicles to perform depot returns prior to being empty, thus adapting their trips to include new orders online. Both features result in shorter distances and allow for more agile planning. The problem is solved online in discrete time steps. In each time step, trips for vehicles are computed. Vehicles then follow them till the next time step. In each time step, a large set of potential trips is calculated. Subsequently, which of these trips will be executed and by which vehicle is decided by solving an integer linear program. Trips can serve multiple orders together, define which depots to use, and can be updated in later steps. Results show an improvement of 20% over a greedy approach. Considering multiple depots also shows to be beneficial. Experiments with up to 10500 orders show the scalability of the approach.

2.1. 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 have 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 [2]. 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 [8]. Similarly, in several countries in South and North America, the delivery company Cornershop has recently merged with Uber with a similar purpose [9].[1]

This chapter 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 chapter is devoted to filling that research gap.

The 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,

[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 [3]. It might become feasible to operate large fleets with moderate costs and without the inherent risks that the human riders currently face [4, 5, 6, 7]. Nevertheless, this chapter does not exclusively assume autonomous vehicles. It applies the same to human-driven ones.

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 chapter formally defines the FDP 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 chapter 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 [21, 22], and the multi-depot vehicle routing problem [23]. Moreover, it requires dynamic optimization and can easily scale to large problem sizes.
To illustrate the concept that considering multiple depots and en-route adaptions can lead to shorter trips that deliver more orders quicker, we give an example. The example is illustrated in Figure 2.1. 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 purple. Subsequently, it needs to return to depot A and then drive to the new customer individually, the second part of the solid purple tour. 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 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.

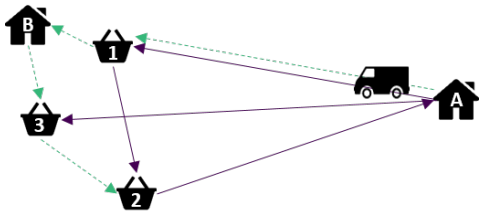


Figure 2.1: An exemplary tour of one vehicle serving two known orders (1 & 2) and one newly requested order (3) that gets placed after the vehicle has already left depot A. The solid purple arrows show the trip of the vehicle when there is only one depot and no pre-empty depot returns. The dashed green arrows show the trip when using multiple depots (A & B) and allowing for pre-empty depot returns.

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 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 chapter 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 preempty 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.

2.2. 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 [24]. To the best of our knowledge, there are no works tackling routing for the FDP up to now.[2] As such, in Section 2.2.1, we discuss the most relevant SDDP works. In Section 2.2.2, we have a look at other related works.

2.2.1. 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.

[2] One exception is [25], which tackles routing for the FDP with heterogeneous vehicles. We exclude this work here as it is part of this dissertation.

This related work section focuses on routing optimizations for the SDDP [26], [27] and [28], routing refers to actively deciding on the routes of vehicles. This excludes works on order assignment or the sole dispatching of vehicles [29, 30, 31, 32, 33, 34]. [26] use a multi-scenario sampling approach, first introduced by [35]. They are leveraging waiting strategies and test on scenarios with up to 800 orders and up to 13 vehicles. Similar to our work, [27] 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 [27] can plan for a single vehicle. [28] 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 [26]. Scenarios of up to ten vehicles were analyzed. A SDDP with micro-hubs was tackled by [36] using a two-stage stochastic programming approach. Also, [37] 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 [38]. The problem is divided into a series of static problems. Orders are inserted online into trajectories based on a similarity measure.

This chapter 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.

2.2.2. Other related Problems

In addition to the SDDP, other problems are related to the FDP. The meal delivery routing problem [39, 40, 41] 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 [42], 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 [11, 43] 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 [44], 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 [45, 46, 47].

The approach proposed in this chapter is based upon a routing method for a rideshar-

ing system [11] that transports people in metropolitan areas. This method is called Vehicle-Group Assignment (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 any-time optimal manner.

In regards to considering multiple depots for dynamic problems, the work in this chapter is connected to the Dynamic Multi-Depot Vehicle Routing Problem (DMD-VRP). Only a few works tackled this problem. It has been tackled by decomposing the problem into multiple single-depot Dynamic Vehicle Routing Problem (DVRP), where each order is assigned to one fixed depot, and each sub-problem is solved separately [48, 49]. In contrast, we include the decision of which depot should be used within the routing decision itself, and thus, this chapter is the first, up to our knowledge, to consider multiple depots simultaneously for a DVRP.

2.3. 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 $G = (\mathcal{N}, \mathcal{A})$ where $\mathcal{N}$ 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 $n_1, n_2 \in \mathcal{N}$ 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 $\xi \in \mathcal{N}$ 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 $\Xi \subset \mathcal{N}$. 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 $g_o \in \mathcal{N}$. We

[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 [50, 51, 52, 53], 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.

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 $\mathcal{IO}_t$ 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 $\mathcal{O} = \mathcal{UO}_t \cup \mathcal{PO}_t \cup \mathcal{LO}_t \cup \mathcal{DO}_t \cup \mathcal{IO}_t$. 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 $\mathcal{UO}_{T_{end}} = \mathcal{PO}_{T_{end}} = \mathcal{LO}_{T_{end}} = \emptyset$.

A major point of distinction between 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[7]. 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 $\theta_o \leq \delta_{delay} \quad \forall o \in \mathcal{O} \backslash \mathcal{IO}$. 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 2.2.

[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.

[7] Alternatively a maximally allowed lead time can be defined, which is a defined maximal time till delivery and independent of the ideal delivery time. This approach is often seen in practice. One benefit of defining a maximum delay is the equal urgency of all orders, in contrast to a dependency on the distance of the goal location to the depot.

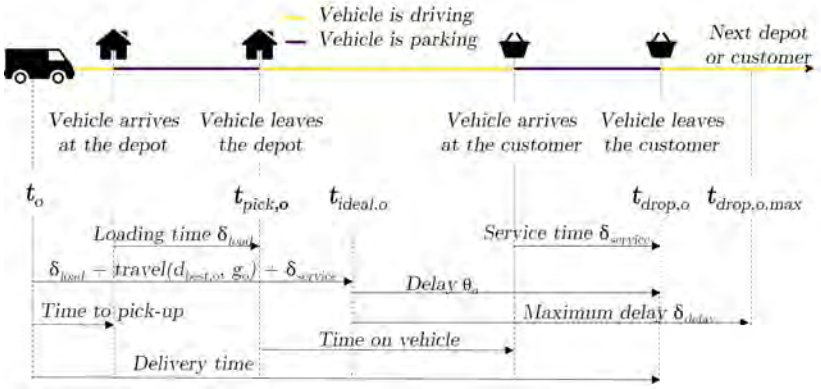


Figure 2.2: Visualization of the different times and time spans for one order.

Following [54] on modeling a MDP 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 $\mathcal{LO}_{v,T_{start}} = \emptyset \quad \forall v \in \mathcal{V}$.
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 plans as trips. A trip T_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}. A feasible action a_k is a set of feasible trips $a_k = (T_1, T_2, ...)$. The number of orders considered by all trips of 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, $\mathcal{LO}_{v,t} \leq C \quad \forall v \in \mathcal{V},\ t \in [0, T_{end}]$. 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 [54], 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 2.1.

$$J(a_k, t_k) = \left[(1-\beta) \cdot \sum_{o_{T_v} \forall T_v \in a_k} \theta_o + \beta \cdot \sum_{T_v \in a_k} \tau_{T_v} + \alpha \cdot (|\mathcal{PO}_k| - |o_{a_k}|) \right] \tag{2.1}$$

In this chapter, 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 [22]. Figure 2.3 depicts a schematic visualization 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 2.2.

$$\mathcal{J}_{T_{end}} = \left[(1-\beta) \cdot \sum_{o \in \mathcal{DO}_{T_{end}}} \theta_o + \beta \cdot \sum_{v \in \mathcal{V}} \tau_v + \sum_{o \in \mathcal{IO}_{T_{end}}} \alpha \right] \tag{2.2}$$

Two details worth highlighting. First, the sum of individual rewards of all decisions

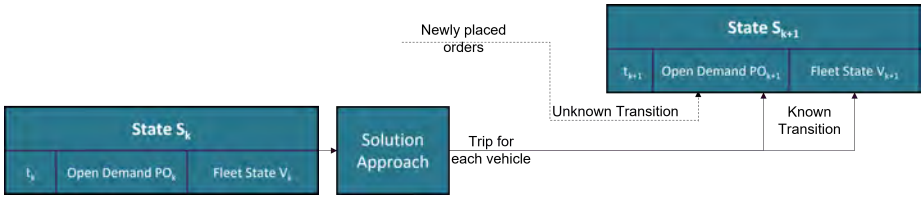


Figure 2.3: Visualization of the transition between two consecutive states. The transition of the vehicle fleet is known. In contrast, the transition of the open demand is partly unknown due to customers placing new orders.

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 2.4 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.

2.4. Method

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

2.4.1. 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 the start to the 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 [55].

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 [56, 35, 15, 57].

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, smaller Δt, 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 2.4.

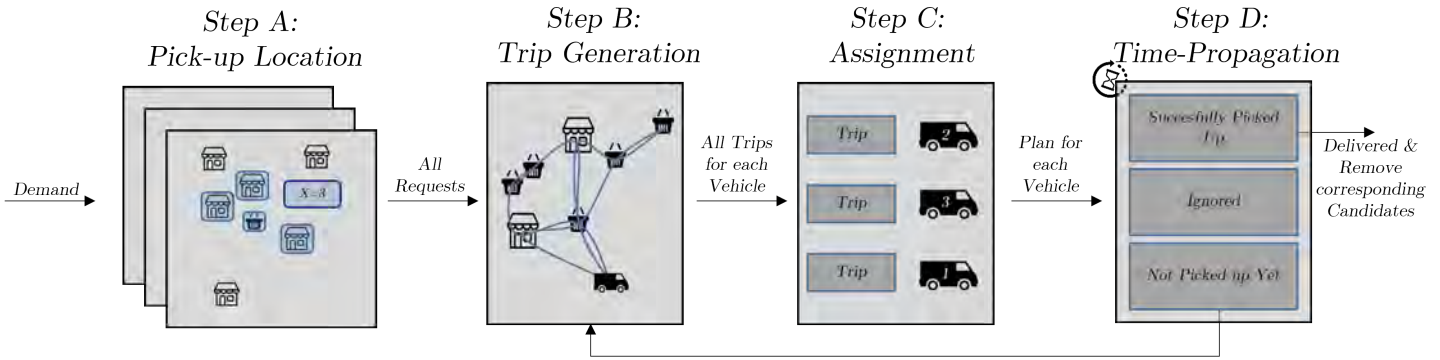


Figure 2.4: Schematic overview of our solution approach. Step A assigns several potential pick-up locations to each order. During step B, individual candidates c (combinations of orders and specific pick-up locations) are combined to feasible trips. In Step C, trips to be executed and corresponding vehicles are selected. Within step D, we propagate time and vehicles follow their assigned trips.

2.4.2. 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 deliver 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 $p_c \in \Xi$. $\mathcal{I}_o^{\mathcal{C}}$ 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 $x = 1$.

2.4.3. 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 that 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-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:

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

- The binary logic function **CandidateVehicle**(v, c) is valid if vehicle v can feasibly serve candidate c.
- The binary logic function **TwoCandidates**(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 **FeasibleTrip**(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 **BestTripSequence**(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 2.2, and calculates as follows:

$$\gamma_{T,v} := (1-\beta) \cdot \sum_{o_T} \theta_o + \beta \cdot \tau_T \tag{2.3}$$

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 2.3) 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.

2.4.4. 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 ILP. The ILP is presented in Equations 2.4-2.8.

$$argmin_{\chi} \sum_{T, r \in \epsilon_{\mathcal{T}\mathcal{V}}} (\gamma_{T,v} - \gamma_{loaded,v}) \epsilon_{\mathcal{T},v} + \sum_{o \in \{1,\dots,|\mathcal{PO}_t|\}} \alpha \chi_o \tag{2.4}$$

$$\sum_{T \in \mathcal{I}_v^{\mathcal{T}}} \epsilon_{\mathcal{T},v} \leq 1 \qquad \forall v \in \mathcal{V} \tag{2.5}$$

$$\sum_{c \in \mathcal{I}_o^{\mathcal{C}}} \sum_{T \in \mathcal{I}_{c_o}^{\mathcal{T}}} \sum_{v \in \mathcal{I}_T^{\mathcal{V}}} \epsilon_{\mathcal{T},v} + \chi_o = 1 \quad \forall o \in \mathcal{PO}_t \tag{2.6}$$

$$\chi_o \in \{0,1\} \tag{2.7}$$

$$\epsilon_{\mathcal{T},v} \in \{0,1\} \tag{2.8}$$

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

Equation 2.4 describes the objective function. Note that the considered costs are relative. From the costs of a vehicle's trip $\gamma_{T,v}$ (see Equation 2.3), 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 2.5 ensures that each vehicle is at most assigned to one trip. Equation 2.6 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 2.7-2.8 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 2.8 defines $\epsilon_{\mathcal{T},v}$ 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.

2.4.5. 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** ($o \rightarrow \mathcal{IO}_{k+1}$), 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:

$$\zeta = (\delta_{\text{delay,real}} - (\delta_{\text{delay,real}} \mod \delta_{\text{delay,heuristic}}))/\delta_{\text{delay,heuristic}} \tag{2.9}$$

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.

2.4.6. Complexity and Optimality Analysis

Complexity: Our approach divides the full-day problem (Section 2.3) into multiple sub-problems at specific times t_k. Each sub-problem deals with its associated state S_k. The trip generation step (Section 2.4.3) is the most complex and, thus, the bottleneck of the proposed approach. The ILP (Section 2.4.4) 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 becomes 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 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.

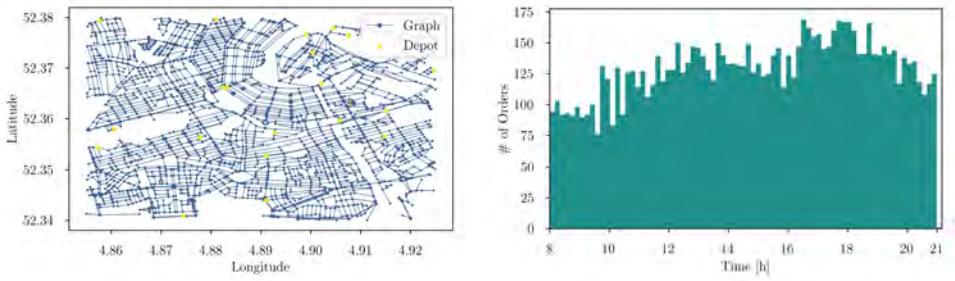
2.5. Experiments and Results

In this section, we present the computational experiments. First, Section 2.5.1 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. Secondly, in Section 2.5.2, 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 2.5.3 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 2.1 in the Appendix contains all results of all analyzed scenarios.

2.5.1. 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 2.5(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 $\mu = 36\frac{\text{km}}{\text{h}}$.[8] 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 2.5(b). Each bar shows the number of newly placed orders within ten 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 ten. The maximum delay $\delta_{\text{delay,real}}$ is set as eight 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

[8] If more accurate data is available, like historical data per street or time-dependent data, this information should be used instead.



(a) Graph representing the city centre of Amsterdam.

(b) Temporal distribution of placed orders.

Figure 2.5: A visual representation of the underlying graph $G = (\mathcal{N}, \mathcal{A})$ 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 $\forall o \in \mathcal{O}$ is depicted. Each bar shows the number of newly placed orders within ten minutes.

two different objectives with $\beta = 1/3$. These values have been chosen to create a scenario that is serving most orders but can't serve everything. To solve the ILP described in Equations 2.4-2.8, 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.

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 2.6 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 Key Performance Indicators (KPIs)) involved in the delivery process of each order, see Figure 2.2. 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 2.7. These times can be compared to the average distance of all nodes to their closest depot, which is 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, 8min, the maximum allowed delay.

Let us analyze the delay in more detail. We distinguish two time windows of two hours, one in the morning (09:00 to 11:00) with a low workload and one in the evening (17:00 to 19:00) with a high workload. Figure 2.8 shows the delay distribution for all orders placed in the corresponding time windows. For a low workload (Figure 2.8a), the average delay is significantly lower and the overall shape of the distribution is less pushed towards the maximum delay. With a lower workload, additional resources become available, thereby allowing the improvement of the service

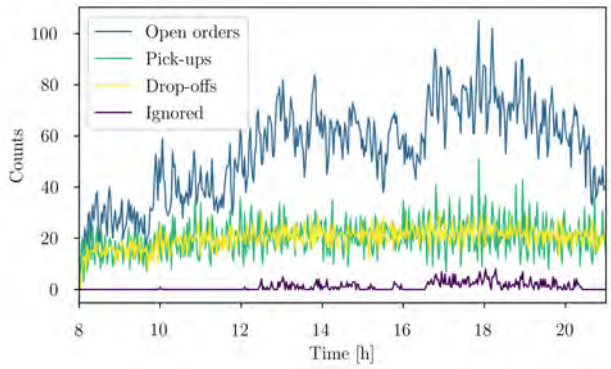


Figure 2.6: The number of open orders, the number of picked up and dropped off orders, as well as ignored orders per time step are visualized.

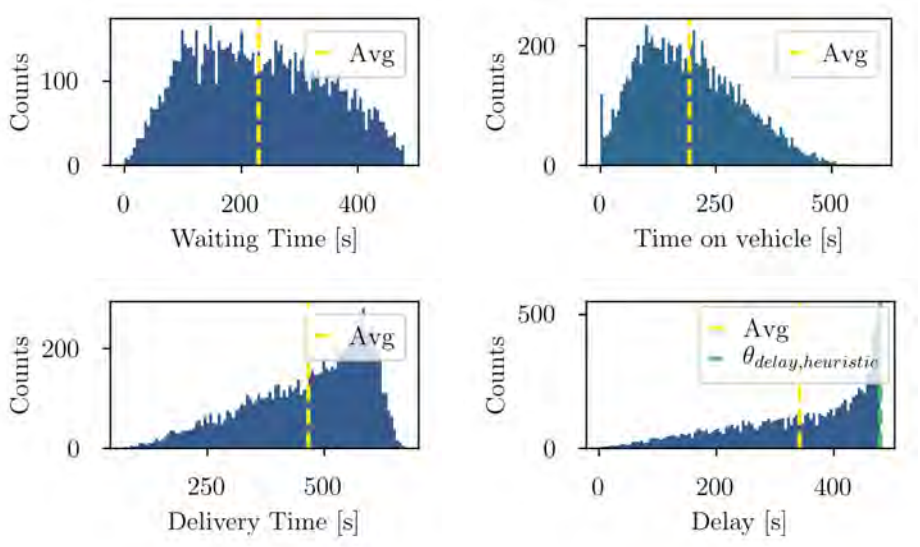
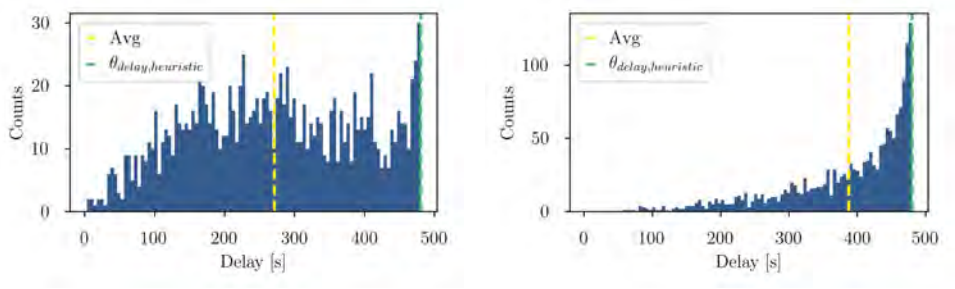


Figure 2.7: Distributions of time to pick-up, time of orders spend loaded to a vehicle, the total delivery time, and delay of the base scenario.

level without the necessity of serving additional orders initially. This is also reflected in the number of rejected orders, as shown in Figure 2.6. In contrast, during high workload (Figure 2.8b) most orders are served with a high delay.

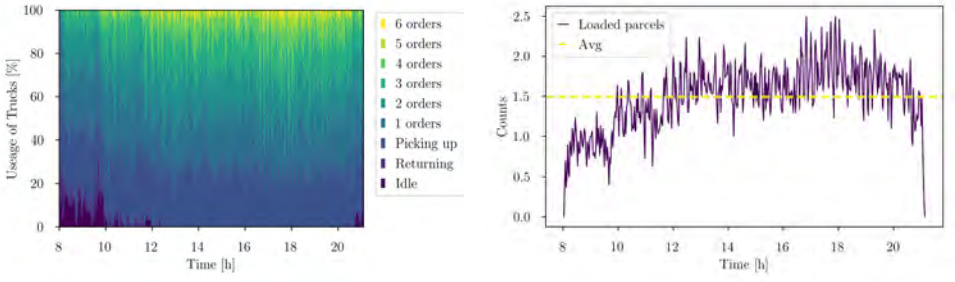


(a) Delay distribution 9a.m. - 11a.m.

(b) Delay distribution 5p.m. - 7p.m.

Figure 2.8: Distribution of delay for two different time windows differing in the workload of the base scenario.

Third, we analyze how the proposed method utilizes each vehicle. The occupancy of all vehicles is depicted in Figure 2.9(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, best seen by the further reaching down parts in bright green. 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 2.9(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, captured in the KPI: mean-loaded orders.



(a) Occupancy.

(b) Mean-loaded orders.

Figure 2.9: Shown on the left, occupancy of all vehicles over the entire operation duration. Shown on the right is the mean number of loaded orders of all vehicles as the day progresses.

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.

2.5.2. Comparison

We now assess the performance of our method by comparing it to three approaches. First, in Section 2.5.2, we compare the results obtained with our approach with those obtained by using a greedy assignment strategy. Second, in Section 2.5.2, 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 2.5.2. For simplicity, we refer to the scenario analyzed in Section 2.5.1 as the base scenario. The parameters' values are set as in the base scenario.

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 an 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 2.10. 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.

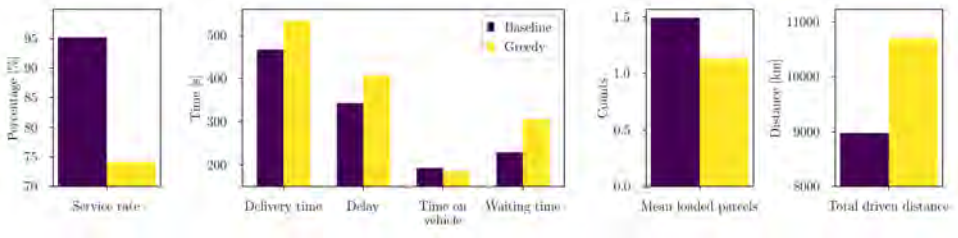


Figure 2.10: Service rate, time KPIs, mean-loaded orders and total driven distance of the base scenario and the greedy assignment strategy.

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 2.4.2 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 2.11. 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 of 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 2.5.3 for the analysis of additional values of x.

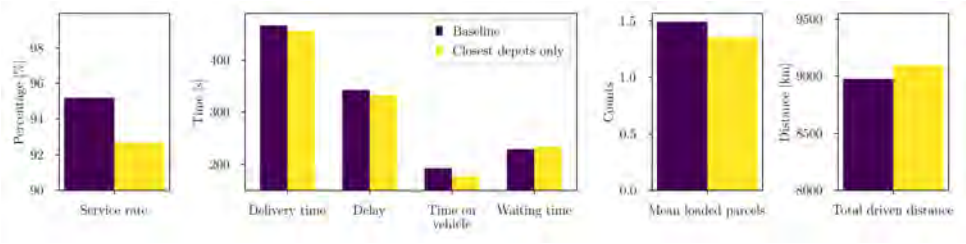


Figure 2.11: Service rate, time KPIs, mean-loaded orders and total driven distance of the base scenario and the same scenario considering the closest depot only.

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 2.12. 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.

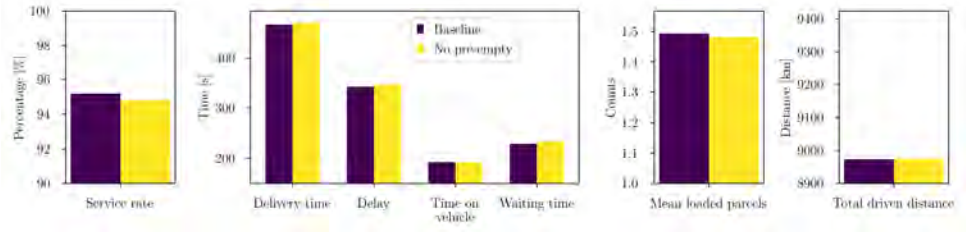


Figure 2.12: Service rate, time KPIs, mean-loaded orders and total driven distance of the base scenario and the same scenario with prohibiting pre-empty depot returns.

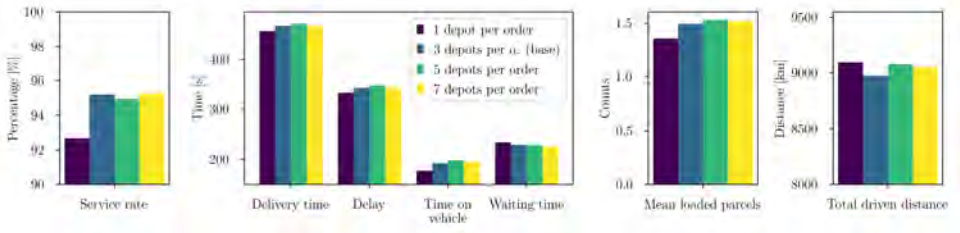


Figure 2.13: The main performance indices for different numbers of considered depots per order are visualized.

2.5.3. Sensitivity Analysis

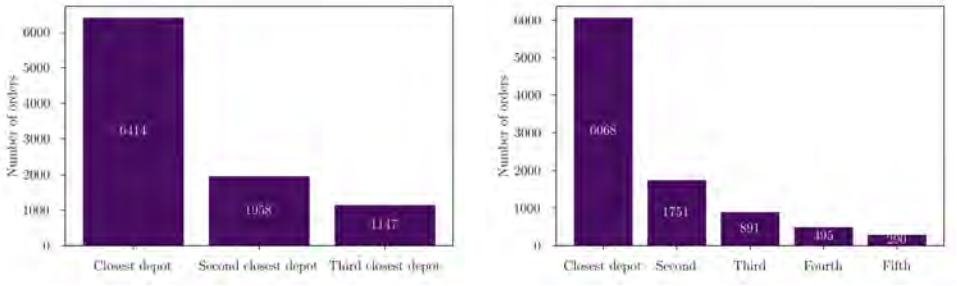
In this Section, we analyze the effects of five parameters, namely: the number of considered stores, the total number of stores, the effect of allowing to reinsert orders into the problem to enable longer delivery times, the number of used vehicles, and the used cost function. We perform a sensitivity analysis for each one of them. We present the analyses in the following sections. All parameters are identical to the base scenario (Section 2.5.1) unless mentioned otherwise.

Number of considered depots per order x

Our heuristic, introduced in Section 2.4.2, 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 of potential candidates and, consequently, the higher the computational load.

Figure 2.13 shows the results if one (see Section 2.5.2), 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 2.14 additionally shows the usage of depots, if they are ranked according to the distance to their order's destination, for the baseline (Figure 2.14a) and the scenario considering five depots per order (Figure 2.14b). 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 ap-



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

Figure 2.14: Depots are categorized 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.

pear 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 oder, as [48] and [49], 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 chapter 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 2.15. 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 [58].

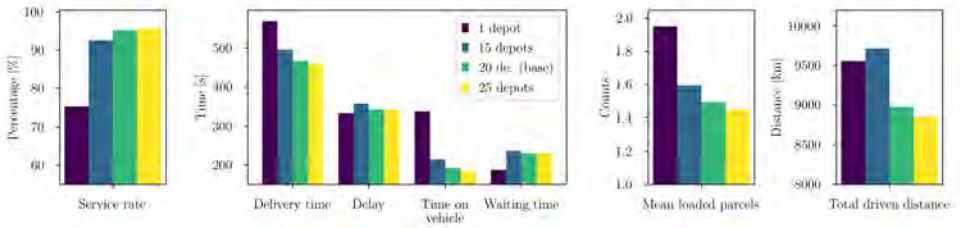


Figure 2.15: Service rate, time KPIs, mean-loaded orders, and total driven distance of the four different runs, featuring a different number of total depots distributed over the service area.

Allowing for reinsertion of orders

In Section 2.4.5, 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 order would be considered ignored. But, by allowing reinsertions, see Section 2.4.5, 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 2.16 shows a comparison with the of $\delta_{\text{delay,real}} = \delta_{\text{delay,heuristic}} = 8$, as in Section 2.5.1. 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 2.17(a), where we observe a repetitive nature. The number of reinsertions is shown in Figure 2.17(b), showing a significant number of orders that are reinserted at least once.

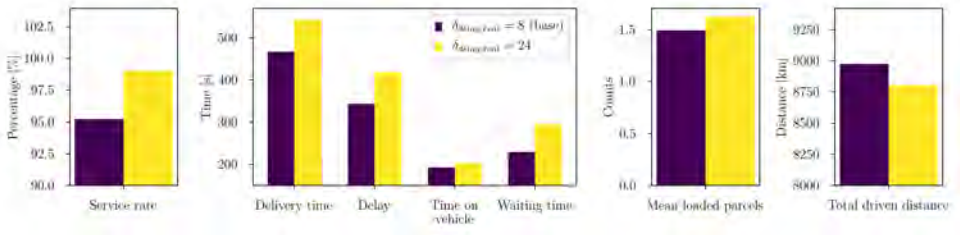
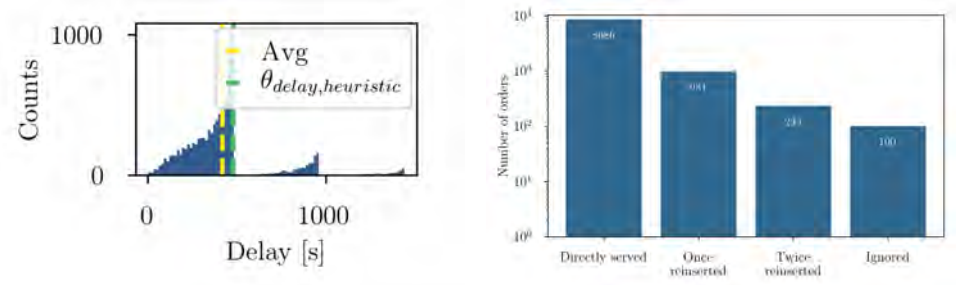


Figure 2.16: Service rate, time KPIs, mean-loaded orders, and total driven distance of the base scenario and the same scenario having a $\delta_{\text{delay,real}}$ of 24 minutes, thus allowing reinsertion.

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 2.18 depicts the corresponding results. The more orders are placed, the lower the service rate because available resources are kept constant. Nevertheless, the absolute number of



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

Figure 2.17: 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.

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.

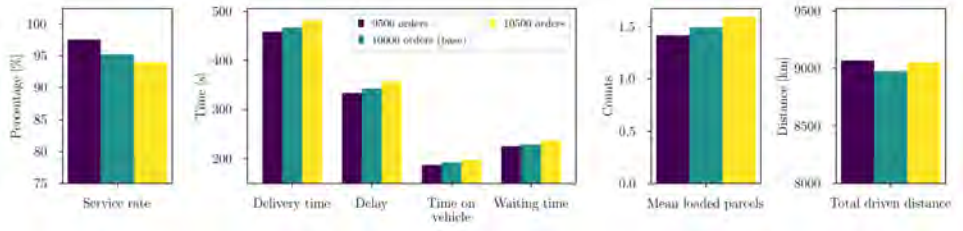


Figure 2.18: Service rate, time KPIs, mean-loaded orders, and total driven distance of the three different runs, featuring different demand patterns.

Number of used vehicles:

We varied the number of used vehicles to 25 and 35. Figure 2.19 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 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

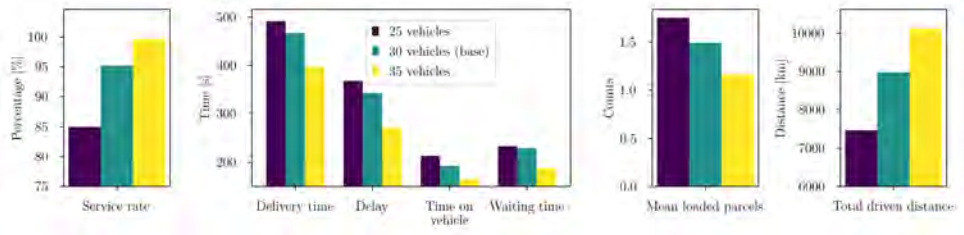


Figure 2.19: Service rate, time KPIs, mean-loaded orders, and total driven distance of the three different runs, featuring a different number of vehicles.

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 2.1-2.4 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 2.20 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 unfavorable states to serve future demand if the goal is to serve as many orders as possible.

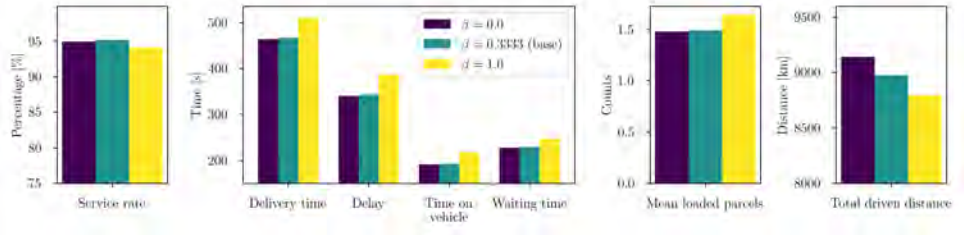


Figure 2.20: Service rate, time KPIs, mean-loaded orders and total driven distance of three runs, featuring different cost weights β.

2.6. Conclusion

In this chapter, we introduce and formalize the FDP. The FDP is a variation of the SDDP, aiming to deliver orders in minutes. The proposed approach considers picking 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.

2.7. Chapter Appendix

Results

The precise values of all performance indices of all executed runs are shown in Table 2.1.

	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

Table 2.1: Precise results of all performance indices of all experiments of Chapter 2.

3

Routing of Heterogeneous Fleets for Flash Deliveries via Vehicle Group Assignment

The routing method proposed in the *previous chapter* will serve as the starting point of this chapter. So far, the proposed approach considers vehicles that are homogeneous and are bound to the street network of a city. Using a single type of vehicle, whether small trucks, bikes or drones, can be restrictive and cause inefficient operations.
In this chapter, we generalize the approach to multiple modes of transportation. We investigate how multiple types of transportation can be combined to serve Flash Delivery operations and propose an adapted method to do so. The new main challenge is the decision on which vehicle, including its type, to use for which orders depending on the vehicle's characteristics. We study an example scenario featuring trucks and drones.

This chapter is based on:

- M. Kronmueller, A. Fielbaum and J. Alonso-Mora, *"Routing of Heterogeneous Fleets for Flash Deliveries via Vehicle Group Assignment"*, in IEEE 25th International Conference on Intelligent Transportation Systems (ITSC), pp. 2286-2291, 2022 [25]

Abstract

This chapter presents a novel approach to route heterogeneous fleets for Flash Delivery operations. Flash deliveries offer to serve customers' wishes in minutes. We investigate a scenario that allows to pick-up orders at multiple depots with a heterogeneous vehicle fleet leveraging different modes of transportation. We propose the Heterogeneous Vehicle-Group Assignment (HVGA) method, which, given a problem state, identifies potential pick-up locations, calculates potential trips for all modes of transportation and last chooses from the set of potential trips. Experiments to analyze the proposed method are executed using a fleet featuring two modes of transportation, trucks and drones. We compare to a state-of-the-art method. Results show that HVGA is able to serve more orders while requiring less total traveled distance. Further, the effects of the fleet size and fleet composition between drones and trucks are examined by simulating three hours of a Flash Delivery operation in the city center of Amsterdam.

3.1. Introduction

Using heterogeneous fleets for last-mile delivery operations allows leveraging the strengths of different modes of transportation. For example, a drone can maneuver independently of roads and traffic and thus deliver quickly in hard-to-reach areas and a truck can load many parcels simultaneously and deliver multiple orders in one neighborhood. In addition, grocery deliveries within minutes have established themselves within many cities. In the Netherlands alone, consumers spent around 40 million euros per month on Flash Deliveries at the end of 2021 [2]. Combining these two aspects poses a highly relevant and interesting question: How do we route heterogeneous fleets for on-demand last-mile deliveries?

This chapter proposes a novel optimization-based approach to route heterogeneous fleets for on-demand last-mile deliveries, considering multiple depots and short delivery times. The proposed method is able to combine many types and forms of transportation while staying scalable. We specifically investigate the use case of supporting ground-based vehicles with drones.

The main virtues are threefold. First, we generalize a well-known method for ride-sharing, called Vehicle Group Assignment, to be able to handle heterogeneous fleets, including various modes of transportation. Second, this chapter combines dynamic multi-depot vehicle routing with heterogeneous fleets. Third, to the best of our knowledge, this chapter is first in investigating the use of drones for Flash Delivery operations.

3.2. Related Work

The problem we face can be classified as a dynamic and heterogeneous vehicle routing problem (pick-up and delivery problem). Thus it is related to the two broad fields of Dynamic Vehicle Routing Problems (DVRP) and Vehicle Routing Problems with heterogeneous fleets. For an overview of dynamic routing problems, see [59, 60, 61], and for an overview of routing for heterogeneous fleets, see [62]. Heterogeneous routing problems are further divided by the type of vehicles they are

considering and by the way these can interact with each other. For a more detailed analysis, we exclusively focus on dynamic problems, as approaches deployed between static and dynamic vehicle routing problems differ strongly.
Proposed methods to tackle the dynamic vehicle routing problem with a heterogeneous fleet are mainly based on heuristics. Large Neighbourhood Search based strategies were applied by [63] considering a fleet of trucks with different capabilities and by [64] tackling the technician routing problem (technicians differ in skills and repair parts carried). [65] used a Tabu Search to improve routes while dynamically integrating new incoming orders. They consider different vehicle speeds and capacities. The problem of dynamically refueling airplanes by trucks with different speeds, capacities, and fit planes was studied by [66] using a genetic ant colony algorithm.

More specifically, the problem studied in this chapter is a specific case of a SDDP [30, 26, 67, 33, 68, 69]. Works that combine an SDDP and a heterogeneous fleet are sparse. We identified three works, analyzed in the following.
[67] proposes a method routing a heterogeneous fleet of trucks and drones for a SDDP, splitting it into two. For each new order, it is decided if it is served by drones or trucks or is ignored. Subsequently, each mode of transportation is routed independently. To determine if a drone or a truck should be used, they apply policy function approximation based on geographical districting. Orders which have a travel time longer than a given threshold are preferably served by drones. [68] builds upon [67] by adapting the way orders are assigned to a mode of transportation. They leverage Q-learning to do so and improve on previous results. Improvements are most substantial for small fleets but vanish fully if more resources are provided. [70] investigates a SDDP, in which drones are used to resupply trucks, which exclusively deliver to customers. They examine two approaches for resupplying: first, only empty trucks can receive a resupply and second, resupplies are possible anytime. They optimize on finding the best resupply locations, serving as many orders as possible.
In contrast, this chapter investigates a specific variation of Same-Day Delivery (SDD), a Flash Delivery operation featuring multiple depots. HVGA further differs from the works proposed for heterogeneous SDD by not splitting into assigning orders to vehicles and routing but doing so jointly.

The proposed approach in this chapter is building on a ride-sharing method called VGA [11]. VGA is an anytime optimal approach given enough computational time. It shows great scalability. A retail context variation considering multiple depots was investigated by [19]. This chapter extends VGA to heterogeneous fleets.

As we test our method considering scenarios in which drones and trucks are utilized, we want to point interested readers to an overview on routing problems featuring the usage of drones (mostly static), see [71].

3.3. Problem Formulation

This section presents the problem statement covering the whole operation and introduces the used notation (Section 3.3.1). As our problem is dynamic, we model it as a MDP, explicitly capturing the problem's dynamics (Section 3.3.2).

3.3.1. Notation and Problem Statement

We consider a heterogeneous fleet of vehicles $\mathcal{V}$ consisting of multiple sub-fleets, each having a different mean of transport, noted by the subscript m. Each vehicle, independent of its mode, is denoted by $v \in \mathcal{V}$. For the sake of simplicity, we describe the method assuming a truck fleet $\mathcal{L}$ of H identical trucks and a drone fleet $\mathcal{D}$ of G identical drones. The generalization towards more than two sub-fleets is straightforward. They move on a weighted and directed graph $G_m = (N, A_m)$, thereby the arcs and arcs' weights are mode-dependent. Apart from the network they travel on, the fleets differ in their maximum capacity of each vehicle C_m and their traveling speed s_m. A set of depots De, locations where orders can be picked up, is placed within the graph at specific nodes $\{de_1, ..., de_{|De|}\}$[1].

Each customer is defined as an order o. Customers can request the delivery of goods to a location of their desire. We denote the goal location by g_o, which is assumed to be a node. The time a customer orders is called the request time t_o. All orders are summarized within the demand $\mathcal{O}$. Each order is unique and has a size of one. We additionally introduce time-dependent demand subsets: First, the set of loaded orders $\mathcal{LO}_t$ consisting of all orders currently loaded to a truck or drone. Further, $\mathcal{DO}_t$ denotes the set of delivered orders and $\mathcal{IO}_t$ the set of ignored orders (i.e., orders that were not delivered within the given constraints). Last, the set of known open orders $\mathcal{OO}_t$ consisting of all orders currently known and unloaded nor delivered nor ignored.

The operation starts at $t = 0$ and ends at $t = T$, with t being the current time. During a short time span of duration δ_T before T, no new orders are accepted. Further, δ_{pick} and δ_{drop} are fixed time spans needed to load or to deliver an order.

General Problem Statement: *Consider a heterogeneous fleet $\mathcal{V}$ consisting of two sub-fleets, a truck fleet $\mathcal{L}$ of H identical trucks and a drone fleet $\mathcal{D}$ of G identical drones. Each mode of transportation operates on a weighted and directed graph $G_m = (N, A_m)$. Multiple depots $de \in De$ are placed within the graph. Each truck and drone is in a known initial state (starting locations and no prior load). Customers continuously place orders, summarized as the demand $\mathcal{O}$, where each order is specified by a request time $t_o \in [0, T - \delta_T]$ and a goal location g_o. The operation starts at $t = 0$ and ends at $t = T$. Find a set of routes for each truck and drone to pick up, thus including the choice of a depot, and to deliver the orders such that a given cost function $\mathcal{J}$ is minimized subject to a set of given constraints.*

We consider two constraints: First, each vehicle's maximum capacity C_m may not be exceeded. Second, each order has a maximum delivery time $t_{o,max,m}$. We calculate the maximum delivery time as the sum of the request time t_o, the optimal delivery time $t_{o,opt,m}$ and a fixed time interval, called maximum delay θ_{max}, i.e., $t_{o,max,m} = t_o + t_{o,opt,m} + \theta_{max}$. Thereby, the optimal delivery time $t_{o,opt,m}$ is calculated as the time a vehicle needs to travel form the closest depot to the goal location of the order plus the time to pick-up δ_{pick} and to deliver δ_{drop} the order. Note that the optimal

[1] Also shared between trucks and drones.

delivery time is dependent on the used mode of transportation.[2] An order might not be delivered within the given constraints and is thus considered ignored.

3.3.2. Markov Decision Process

An MDP captures the dynamics of a problem by modeling subsequent states $\mathcal{S}_t$ connected by a decision taken and a transition between them.

Decision Points: Decisions are taken at specific points in time, summarized in the set[3] ψ. Individual decisions and corresponding states are enumerated by k. We make decisions in fixed time steps of Δt, i.e., $t_{k+1} = t_k + \Delta t$.

Problem State: The problem state $\mathcal{S}_k$ at time t_k is fully characterized by the time itself t_k, the state of both sub-fleets and the set of open orders $\mathcal{OO}_t$. To describe the fleet's states, each individual vehicle v is described by its location $l_{v,t}$, its set of loaded orders $\mathcal{LO}_{v,t}$ and the plan it follows currently $\pi_{v,t}$. For example, the truck fleet's state is described as $\mathcal{L}_t = ((l_{l,t}, \mathcal{LO}_{l,t}, \pi_{l,t}) \forall l \in \mathcal{L})$. This results in the overall state definition as

$$\mathcal{S}_t = (t, \mathcal{L}_t, \mathcal{D}_t, \mathcal{OO}_t)$$

Note that a plan $\pi_{v,t}$ consists of an ordered sequence of actions (pick-up and drop-off of orders or rebalancing) with associated locations. Between locations, the vehicle follows the shortest path. As soon as an action is executed, it is removed from the plan. Further, a plan can be updated at a later point in time.

Decision: The decision at decision point k is to update the plan $\pi_{v,t}$ of each vehicle v, which it will follow until the next decision point t_{k+1}. Note that a vehicle's plan can change if a subsequent decision updates it, including newly obtained information.

Transition: The transition can be split into a deterministic part and an unknown part. In the deterministic part, we update the truck and drone fleet's status. Each vehicle follows its plan $\pi_{v,t}$ determined within the taken decision. Doing so, a vehicle's location and its loaded orders change (orders can get delivered and new orders can be loaded). As time propagates between subsequent states, customers may place new orders (which is the unknown aspect). As a result, new orders are added to the set of open orders $\mathcal{OO}_{k+1}$. If an order can not be delivered within its constraints anymore, we consider it ignored. The order is removed from the set of open orders[4] $\mathcal{OO}_{k+1}$.

Objective: The goal of the posed problem is to minimize a combination of costs, considering the total driven distance, service quality measured as the delay θ, and a penalty term α for orders that are not delivered (Equation 3.1).

$$\mathcal{J}_T = \left[(1-\beta) \cdot \sum_{o \in \mathcal{DO}_T} \theta_o + \beta \cdot \sum_{v \in \mathcal{V}} \eta_{v,T} + \sum_{o \in \mathcal{IO}_T} \alpha \right] \tag{3.1}$$

[2] Alternatively, a maximally allowed lead time can be defined, which is a defined maximal time till delivery and independent of the ideal delivery time and delivery mode. This approach is often seen in practice.

[3] ψ can be determined during operation or beforehand.

[4] Note that when vehicles load orders following their plan, they get removed from the set of open orders.

Thereby, the delay θ_o is defined per order as the difference between the actual delivery time and the optimal delivery time. The total driven distance of a vehicle $v \in \mathcal{V}$ at time t is denoted by $\eta_{v,t}$. α is a constant predefined penalty for ignoring an order. Note that if α is set to a large constant, the objective function puts the highest priority on serving as many orders as possible. β is a tuneable weighting parameter between operator cost and service quality. Note that we can not directly map a decision taken at a specific state S_t to Equation 3.1. Each vehicle will follow its plan $\pi_{v,t}$ until the next decision state, then each plan may be altered due to new information. Thus, a current plan can not be directly correlated with Equation 3.1.

3

3.4. Method

Here, we describe the proposed method, called HVGA. Following the MDP (Section 3.3.2), each time a decision point is hit, a decision given the current state S_k, determining a set of routes $\pi_{v,t}$ for every vehicle $v \in \mathcal{V}$, needs to be taken.
To do so, we follow a sequence of steps explained in the following. First, we determine multiple potential pick-up locations for each order (Section 3.4.1). Second, we calculate sets of potential trips each truck and drone could take (Section 3.4.2). Last, we pick from those sets of potential trips, which each vehicle should carry out by solving an ILP (Section 3.4.3). An illustrative overview is depicted in Figure 3.1. The proposed method builds on top of previous work for Flash Deliveries for homogeneous fleets, introduced in [19].[5]

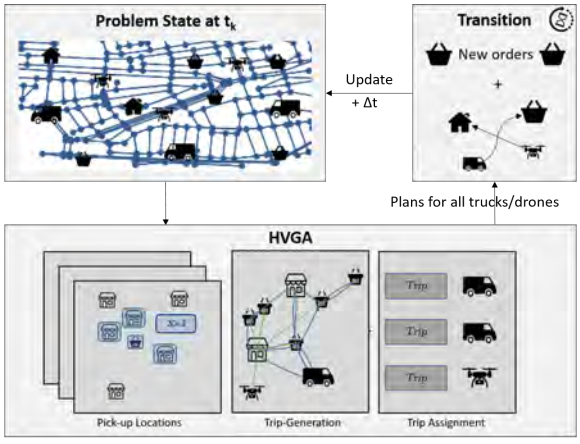


Figure 3.1: Given a problem state, a decision is taken on how to update the plans of all trucks and drones. HVGA creates plans for all vehicles by first identifying potential pick-up locations for each order. Subsequently, a large set of potential trips is calculated for each vehicle. Which of these trips are executed is determined in the last step, called trip assignment. During a transition phase, all trucks and drones follow their determined plan, and new orders are received until a new decision state is met.

[5] [19] also provides a more detailed analysis of their proposed method. Interested readers are referred to their work for further details.

3.4.1. Selecting Potential Pick-up Locations

As orders only specify their delivery location g_o, the decision of where to pick up the corresponding goods is raised. To acknowledge this, we introduce a concept called candidate. A candidate combines an order with a potential pick-up location. We define it as follows: A candidate c is a tuple containing an order $o_c \in \mathcal{O}$ and an associated pick-up location $p_c \in De$. Thus, a candidate is described as $c = (o_c, p_c)$. Each order can have multiple candidates associated with it. Further, we introduce a heuristic that reduces the number of candidates by only considering a subset of depots. We consider the x depots closest in delivery time by truck to the order's goal location. x is a predetermined tuneable parameter.

3.4.2. Finding Potential Trips

A trip, denoted as Γ, is defined as a set of candidates $c_1, ..., c_j$, a vehicle, and a plan, that serves all candidates of the trip,

$$\Gamma = (c_1, ..., c_j, v, \pi_{v,t}) = \big((o_{c_1}, p_{c_1}), ..., (o_{c_j}, p_{c_j}), v, \pi_{v,t}\big)$$

The goal of the trip generation step is to find all potential trips for each truck and drone, summarized as the set ZZ. This set is formed by combining the sets of all potential trips Z_v for each vehicle, i.e. $ZZ = \bigcup_{v \in \mathcal{V}} Z_v$. We calculate the sets Z_v separately for each truck and drone and thus also separately for each mode of transportation. Each trip is further tagged with the cost to execute it.
The general workflow for a truck and drone are identical but differ in the used network G_m, speed s_m and vehicle's capacity C_m. If a trip Γ is feasible for a specific vehicle v, a plan $\pi_{v,t}$ can be found, picking up and delivering all candidates of this trip without violating any constraint. This includes orders that are already on board of the considered vehicle. To generate the complete set of all feasible trips Z_v, the method builds onto the idea that a trip can only be feasible if all its sub-trips are feasible as well. As a result, we start searching for trips of size one. Subsequently, we combine obtained trips to form larger trips, successively increasing in size. The cost of a trip Γ is given by γ_Γ and is derived from Equation 3.1:

$$\gamma_\Gamma = (1 - \beta) \cdot \sum_{o \in \Gamma} \theta_o(\Gamma) + \beta \cdot travel(\Gamma)/s_m \tag{3.2}$$

$travel(\Gamma)$ determines the required distance traveled to serve the according trip Γ. Note that the delay θ_o of an order, as well as the time needed to complete a trip, depends on the trip and the used mode of transportation. The way to navigate between different trip stops is determined by the used graph G_m, and the time needed to traverse a given arc is shaped by the speed of the vehicle s_m. To determine the cost and plan $\pi_{v,t}$ of a trip Γ, we perform an exhaustive search on all possible sequences and continue solely with the cheapest option.

3.4.3. Assigning Trips

Given the set ZZ of all potential trips each truck and drone can take, this step decides which of them are executed. Thereby, we want to coordinate the individual

decisions to maximize performance and minimize cost. This problem is formalized and solved as an ILP, see Equations 3.3-3.6. Equation 3.3 presents the cost function. Note it differs slightly from Equation 3.2 as we only account for changes in vehicles' plans by subtracting the costs needed to serve the already loaded orders of the corresponding vehicle $\gamma_{loaded,v}$. We introduce the constraints that each truck and drone is maximally used once (Equation 3.4). Also, each order is maximally served once or ignored for now (Equation 3.5). Equation 3.6 introduces the binary variable ϵ_Γ, which takes the value of one if a trip Γ is executed; and the binary variable χ_o taking the value one if an order o is not served by the chosen trips.[6]

As a result, each truck and drone are either assigned a new trip, which they execute until the next decision is taken, or they follow their plan as previously determined. If a vehicle becomes idle (i.e., it has no orders to serve after the trips have been assigned), we assign it a special plan, we call rebalancing. The vehicle is routed to its closest depot, such that is in a promising position for future decisions.

$$\text{argmin}_\chi \sum_{\Gamma \in ZZ} (\gamma_\Gamma - \gamma_{loaded,v})\epsilon_\Gamma + \sum_{o \in \mathcal{O}\mathcal{O}_t} \alpha\chi_o \tag{3.3}$$

$$\sum_{\Gamma \in Z_v} \epsilon_\Gamma \leq 1 \qquad \forall v \in \mathcal{V} \tag{3.4}$$

$$\sum_{\Gamma \in ZZ | o_c \in \Gamma} \epsilon_\Gamma + \chi_o = 1 \quad \forall o \in \mathcal{O}_t \tag{3.5}$$

$$\epsilon_\Gamma \in \{0,1\} \quad \text{and} \quad \chi_o \in \{0,1\} \tag{3.6}$$

Remark: Note that HVGA has a large potential to unify different types of vehicles and modes of transportation due to assigning a cost to each trip and using it to make decisions. Trips and their associated costs can thereby be calculated using entirely different approaches. If approaches are similar, this ensures more straightforward comparability. For example, one can only adapt the graph vehicles operate on, from a road network to a water canal system or the speed can be adjusted to the capabilities of an individual vehicle.

3.5. Experiments and Results

First, we compare HVGA to a state-of-the-art method to evaluate its performance in Section 3.5.1. Section 3.5.2 investigates the effect of the size and composition of a heterogeneous fleet of trucks and drones.

All experiments presented in this section use a time window of 3 hours and 10 minutes, whereby no new orders are placed within the last 10 minutes. During this time, 1828 orders are placed randomly within the operation area.[7] As an operation area, we simulate the city center of Amsterdam, with trucks driving along the street network

[6] To solve this problem, standard software, like Mosek or Gurobi, can be used. We used Mosek [72].

[7] We keep the used demand distribution and number constant to allow for better comparison between all analyzed scenarios.

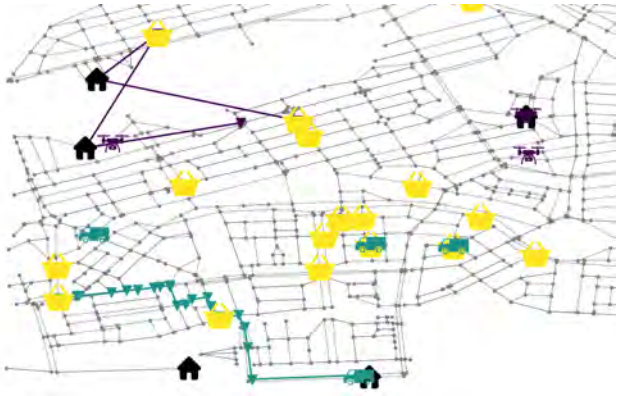


Figure 3.2: A cutout of one particular state of the simulation is illustrated. The road network graph, on which trucks move is shown in grey. The plans of one drone and one truck are highlighted in purple and green. Goal locations of known and non-delivered orders are shown in yellow.

with a speed of 10 meters per second and assuming that drones can fly directly to their goal location with a speed of 15 meters per second. Trucks can load up to six orders simultaneously, whereas drones have a maximum capacity of one. Each order is allowed a maximum delay of 8 minutes. To load an order 15 seconds are needed and 30 seconds to deliver it. HVGA was performed every 100 seconds, i.e., $\Delta t = 100$. The penalty for ignoring an order was chosen as a high constant $\alpha = 10000$ [sec] and the cost weighting parameter as $\beta = 0.3333$.
As a first impression, a snapshot of an area of a particular state is depicted in Figure 3.2. The plans for one truck and one drone are highlighted.

3.5.1. Comparison to Other Approaches

Next to HVGA, we implemented a second approach resembling the work by Ulmer et. al. [67]. This method first decides which mode of transportation is going to be used for which order, followed by a separate routing step. To assign the mode of transportation, they use a policy function approximation based on geographical districting, preferably serving orders with long travel times by drones. To allow for a fair comparison, we adapted the approach [67] to use the routing method as introduced in this chapter. Details on this adapted approach can be found in the Appendix (Section 3.7.1). In the following, we call this approach Split&Route. Here we investigate a scenario in which a single depot placed in the center of the graph is used, as Split&Route is not specifically designed to work with multiple depots. Obtained results are depicted in Figure 3.3. HVGA improves on Split&Route by increasing the service rate by about 17 %. Despite serving more orders, the total driven distance reduces by 108 km. On the other hand, the average delay increases by 24 seconds. Figure 3.4 illustrates the difference in the orders that get served in regard to their distance between the depot and drop-off location, based on the street network. A histogram of orders against travel distance is shown, where orders served by trucks are shown in purple and orders served by drones in yellow (HVGA:

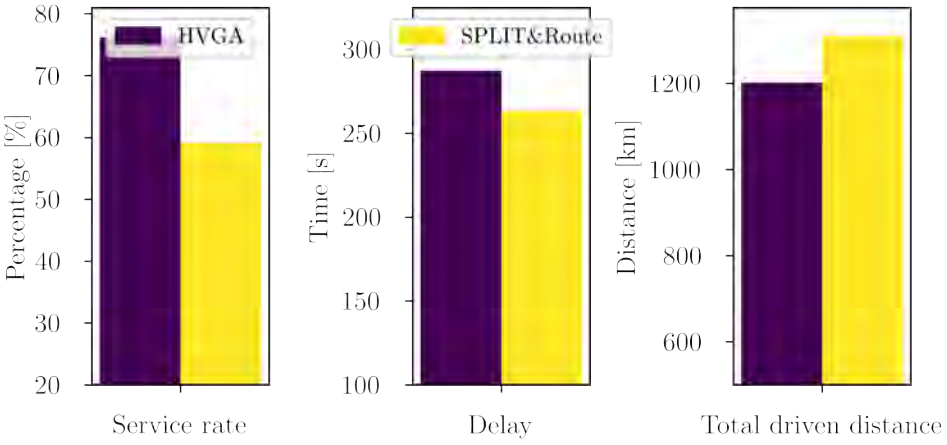
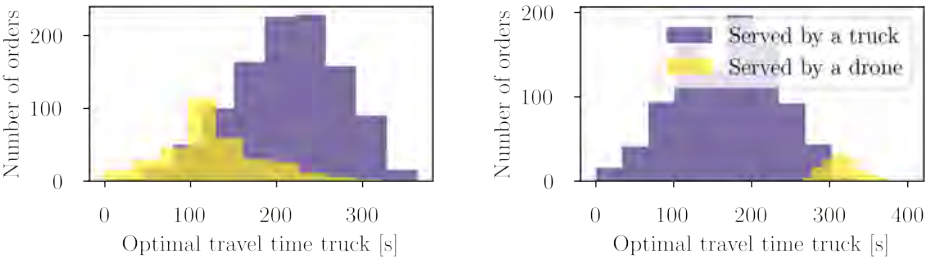


Figure 3.3: Comparison of HVGA and Split&Route

Figure 3.4a, Split&Route: Figure 3.4b). HVGA primarily utilizes drones to serve short-distance orders, standing in direct contrast to Split&Route. This results in a greater amount of orders being served by drones (HVGA: 341, Split&Route: 138). The results suggest that the assumption to serve long-distance orders using drones does not hold for Flash Delivery operations. Not doing so allows drones to serve a larger share of the requests.



(a) HVGA.

(b) Split&Route.

Figure 3.4: Illustration highlighting the difference in served orders regarding their distance between the depot and drop-off location between HVGA, left, and Split&Route, right.

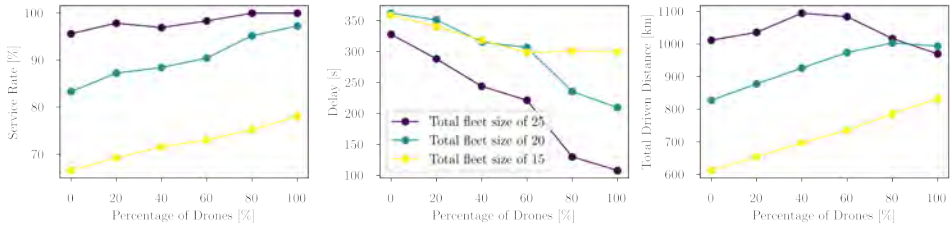


Figure 3.5: Service rate, delay and total travelled distance are depicted for various compositions of the total fleet. Further, the effect of the total fleet size is investigated by setting it to 15, 20, and 25.

3.5.2. Fleet Composition

As this chapter is the first (to the best of our knowledge) in investigating the effect of deploying drones within a Flash Delivery operation, we analyzed the fleet composition in more detail. We increase the number of pick-up locations in the service area to 20, to leverage that our method can handle multiple depots; three are considered per order ($x = 3$). Figure 3.5 shows the change in service rate, delay and total travelled distance for three fleets of total size 15, 20 and 25 and their composition in various ratios of trucks to drones.

Independent of the total fleet size, we see an increase in service rate the more drones are used. This comes with an increase in total traveled distance. We see one exception for a fleet size of 25, when the service rate comes close to 100%. Then resources become available, which the approach can use to stronger optimize on traveled distance and delay. Generally, delay decreases the higher the percentage of drones used.

For Flash Delivery operations drones hold great potential. Short times between ordering and delivery and short distances between drop-off and pick-up location (strengthened by considering multiple depots) fit well with drones' benefits. High capacities are less important if comparing Flash Deliveries to traditional next-day operations, where times between leaving and returning to a depot can cover many hours.

3.6. Conclusion

This chapter presented an optimization-based approach to route a heterogeneous fleet of vehicles for an on-demand last-mile delivery operation. Orders are served within minutes after ordering, posing a special variant of a Same-Day Delivery problem. To analyze the results, a fleet of trucks and drones was studied and compared to a method inspired by [67]. HVGA was able to serve more orders while driving fewer kilometers in total. Further, the size and composition of various fleets have been studied. A larger amount of drones increases obtained service rates at the cost of increased traveled distance.

Future work involves a more detailed representation of used vehicle types, enabling a more accurate study of their use for Flash Deliveries. Further, the scope of ex-

perimental analysis can be broadened such that the most critical drivers for Flash Delivery operations can be identified and studied.

3.7. Chapter Appendix

3.7.1. Details on Split&Route

Split&Route selects a set of orders to be potentially served by drones and routes the drones exclusively for this set of orders. We attempt to serve as many orders as possible by drones. To do so, we take an iterative approach. The individual steps are outlined in the following.

1. We select y orders to be potentially served by drones, further called the drone order set. The y longest orders, based on their traveling distance on the road network, are selected. Note that y can be larger than the number of drones, as it is possible to assign multiple orders to one drone and deliver them subsequently, even though the maximum capacity is set to one. y is a predefined tuneable parameter.
2. We calculate a solution using HVGA (potential pick-up locations, trip generation and trip assignment) for all drones and the drone order set exclusively.
3. We check if all orders of the drone order set are assigned to a drone's plan and none is ignored. If yes, we update the drone plans $\pi_{v,t}$ and route all trucks using HVGA for the set of remaining open orders. If not, we reduce the drone order set by excluding the order with the shortest optimal travel distance and repeat starting from step 2.

4

Reducing the Minimal Fleet Size by Delaying Individual Tasks

The two *previous chapters* proposed methods for routing a predefined fleet of vehicles. Thus answering the question, "What should the existing vehicles do given a concrete situation or problem?". However, the question of the initial number of vehicles required remains unresolved, necessitating an assumed value.
Within *this chapter*, we present an innovative approach to determine the optimal fleet size. This approach expands upon existing research by incorporating the option to delay individual trips. Allowing delays introduces the opportunity for new trade-offs, namely reducing the fleet size for increasing delay. The degree of this trade-off is adjustable. To tackle the fleet sizing issue, we propose a novel mixed-integer linear program, establish its NP-hard complexity, and conduct a comprehensive analysis of the method's potential. We conduct experiments in an abstract environment and a case study of taxi rides in Manhattan.

This chapter is based on:

- M. Kronmueller, A. Fielbaum, J. Alonso-Mora, *"Reducing the Minimal Fleet Size by Delaying Individual Tasks"*, submitted to IEEE Transactions on Intelligent Transportation Systems, 2023

Abstract

This chapter formally defines the problem of fleet sizing with delays (FSD), where the option of delaying individual tasks within fleet sizing is considered. We prove that the Fleet Sizing with Delays (FSD) problem is NP-hard and solve a formulation of the FSD problem as a MILP. We then analyze the proposed method in detail in an abstract case and validate it in a case study of taxi rides in Manhattan. We show that fleet sizes can be decreased significantly and that the trade-off space of the number of required vehicles to execution time and added delay can be enlarged.

4.1. Introduction

Fleets of vehicles of various types are used to drive today's world. For example, taxi fleets offering mobility in cities, robot fleets operating entire warehouses or bike courier fleets delivering your pizza for dinner. Thereby, a wrongly sized fleet, regardless if it is too small or too large, causes inefficient operations regarding bad service or additional costs. Thus, the general question of fleet sizing, i.e., "How many vehicles does an operation optimally need?", is posed. Typically, this question involves a trade-off between the quality of the solution that can be provided and the posed costs. Previous works showed the great potential of fleet sizing, [12] showcase a great reduction in required taxis within Manhattan, or [73] optimizing the size of robot fleets in flexible manufacturing systems.

All works tackling the fleet sizing question assume that tasks have a fixed starting and ending time. In this chapter, we pose a new problem called FSD, which allows to actively delay individual tasks slightly, if beneficial. Let us exemplify through a situation of daily life: Within a family, Person A wants to go from home to sports, leaving at 18:00, and person B arrives at the same home at 18:01. If A cannot delay her trip, two vehicles are required to fulfill this demand. This changes if we allow person A to leave slightly delayed at 18:01. Then, the two persons could use the same vehicle sequentially. This can generalize to many passengers transported by one system.

This chapter introduces and formally defines the FSD problem in a general fashion. We give a general formulation that applies to operations, which can be described as a set of tasks with vehicles that can execute the given tasks on their own and move independently. Through such a general formulation, this chapter covers a wide range of operations. For example, in the case of an operation of shared taxi rides, a task can be a series of customers a single taxi fulfills, transporting a set of customers to their locations until it becomes empty again. In the case of autonomous robots in a greenhouse, a task can represent a set of crops to be harvested or pesticides to be applied. In a warehouse or factory, a task can be a single request of parts to an engineer. In the case of an on-demand delivery service, a task can represent multiple orders one bike rider can transport at the same time.

Contribution Statement: This chapter proposes a new problem, *fleet sizing with delays*. It introduces the option of delaying individual tasks within fleet sizing.

The problem is formally defined. We prove that the FSD problem is NP-hard. Further, we propose and solve a formulation of the FSD problem as a MILP. A large experimental analysis is conducted, analyzing an abstract case in detail and investigating a case study of taxi rides in Manhattan. We show that introducing the option to delay tasks has two effects on obtained results. Firstly, fleet sizes can be decreased significantly. Secondly, the trade-off space of the number of required vehicles to execution time and added delay is increased.

4.2. Related Work

4.2.1. Overview

The fleet sizing problem was originally introduced by [74] in 1984. It poses the question of the number of required vehicles (identical operating characteristics) to satisfy a given demand. Since then, the problem itself has only slightly evolved, mainly changing in the application it is asked for and the scale it can be tackled at. For example, looking at fleet sizing in the maritime context [75, 76, 77], considering the need to charge for electrical vehicles [78, 79], fleet sizing together with service region partitioning [80], or pooled taxi rides [81, 82, 83]. Extending on the fleet sizing question, the question of Fleet Design emerged [84, 85, 86]. Fleet Design goes beyond determining the number of vehicles of one predefined type and incorporates considerations about the types of vehicles to employ from a range of options and the corresponding quantities needed. To tackle the fleet sizing problem, various approaches in various areas have been developed and applied. We categorize these approaches into three categories: chaining-based approaches, simulation-based approaches, and others. As an overview, we highlight each class.

Chaining-based approaches build on the idea of sequencing tasks to build so-called chains, each chain representing the journey of a single vehicle. The obtained number of chains equals the required fleet size if done for all tasks. The proposed approach belongs to this class. As such, we have a closer look below, see Section 4.2.2.

Simulation-based approaches implement a simulation of an operation reflecting its real counterpart as well as possible, [87, 88] (on-demand mobility services), [89, 90] (shared rides) and [73] (flexible manufacturing systems). Once this is obtained, various input parameters are changed, and the resulting Key Performance Indicators are logged. By this means, fleets of different sizes and their influence can be analyzed. The downside to these methods is that no theoretical guarantees can be given and that it might require a lot of computational resources, as the simulation needs to be repeated for each set of parameters. The upside is the capability to represent the simulation environment in more detail, for example, traffic flow or traffic lights.

Other approaches found in the literature are analytical approaches, such as [86] looking at robots in a warehouse environment. They derive an optimal analytical solution for the case of infinite pick-up stations. Further, we see approaches doing a structured search, for example, [91] applies a binary search to find the best fleet size for a fleet of mobile robots. Further, genetic algorithms have been used

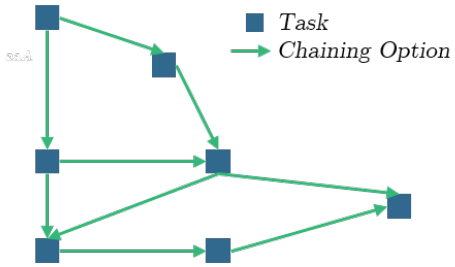


Figure 4.1: Illustration of a vehicle-shareability network [12], vertices are tasks or rides, and edges represent potential chaining options, i.e. a single vehicle can serve both tasks in succession.

in fleet sizing. A comparison of a genetic algorithm and a chaining-based approach (Hopcroft-Karp algorithm for the minimum path cover problem) was made by [92] revealing that the genetic algorithm was outperformed in regards to the quality of solution and the required computational time. [93] builds an optimization model to estimate required fleet sizes based on GPS data. [94] formulates a mixed integer linear problem to solve the joint fleet sizing and charging system planning problem.

This chapter is concerned with centrally controlled systems. That is, we do not review works dealing with cases in which vehicles decide themselves on their plan, such as [95] or [96], both papers modeling the behavior of ride-hailing drivers.

4.2.2. Chaining-based approaches

For the following of this literature review, we will focus on chaining-based approaches alone, as our method belongs in this category. The general idea of chaining-based approaches was first introduced by [12] to tackle the *minimum fleet problem* for taxi rides in Manhattan. [12] first create a so-called *Vehicle-shareability network*, which is a graph capturing which rides could be executed by a single vehicle in succession. Rides or tasks are the graph's vertices, and edges represent potential chaining options. A vertex i is connected to another vertex j, if a vehicle can serve ride j after ride i. An example is illustrated in Figure 4.1. They show how, through these networks, the minimum fleet sizing problem can be transformed into a minimum path cover problem. As the resulting graph is acyclic, the minimum path cover problem can be solved through a maximum-matching algorithm (Hopcroft-Karp algorithm [97]).

Since its introduction, chaining-based approaches have found great popularity, especially within the mobility of people community, and were adapted and extended. Next, we analyze adaption from a problem perspective and from a methodological one, subsequently.

Problem perspective: Finding the minimum fleet for non-shared taxi rides in the city of Manhattan by relocating taxis to serve subsequent customers is tackled by [12]. A group of papers extend chaining-based approaches to the application of ridesharing, i.e. the option for multiple customers to share one vehicle simul-

taneously. A minimum fleet sizing problem with ridesharing was studied by [81], [82] and [83], while the latter focused on the inclusion of demand predictions. [98] formulates the pooling and chaining questions in a combined fashion and shows the effect of pooling on obtained fleet sizes. Fleet sizing for On-Demand Multimodal Transit Systems, combining fixed routes with on-demand vehicles, was studied by [99].

From a **methodological** point of view, the most relevant question for chaining-based approaches is how the chains are found. As a starting point, the problem is often modeled as a graph, like the vehicle-shareability-network, capturing chaining options. Transforming the graph into a bipartite graph and applying a maximum-matching algorithm (like the Hopcroft-Karp algorithm) results in a minimal fleet, like [12, 83, 100]. Other works formulate an ILP and solve it as a minimum flow problem [99, 101]. If other objectives are considered beyond the fleet size, such as the costs to chain tasks, other techniques are required. With costs, the problem changes to a minimum-weight bipartite matching problem. It can be formalized as an ILP considering costs in the objective function and solved as such or as a bipartite matching problem with costs, approached by means of a Hungarian algorithm. Forming chains iteratively by means of solving an ILP repeatedly was done by [81]. They use the chains formed up to this point as input and prolong them by adding new tasks at each step. They start at the end of the day, and with each step, they go further back in time.

This chapter extends the chaining-based approaches by allowing to delay individual tasks. This changes the problem in both regards, from the tackled problem perspective and methodological. From an application-driven point of view, this enables to decrease minimal fleet sizes even further, by adding delay. From a methodological point of view, the newly posed problem can not be tackled by the existing approaches as the decision, which tasks are delayed and if, by how much, is added.

4.3. Problem Formulation

In this section, we formally introduce and define the FSD. Intuitively described, the FSD poses a problem in which a fleet of vehicles and their operational plans need to be found to fulfill a given set of tasks while allowing to delay individual tasks. Solutions are optimized based on a given objective.

4.3.1. Formal Problem Formulation

We first introduce the inputs, then the decision variables, and last, the cost function of the problem.

Inputs: The operational environment is represented by a weighted directed graph $G = (V, E)$. The set of vertices V represents all locations, and the set of weighted edges E represents connections between them. An edge exists if a vehicle can move from one vertex to another directly, and the edge's corresponding weight $c(e)$ is the

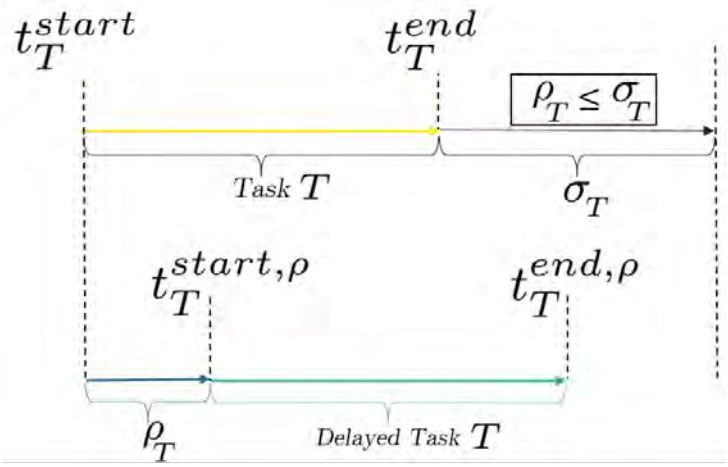


Figure 4.2: Illustration of a single task, its delayed variant, and associated points in time. For clarity, we use $t_T^{start,\rho}$ and $t_T^{end,\rho}$, as the notation for the new starting and ending time of a delayed task.

time needed to traverse it. All tasks are summarized in the set $\mathcal{T}$. Each individual task T is characterized as $T = (l_T^{start}, l_T^{end}, t_T^{start}, \alpha_T, \sigma_T)$: the first two variables representing the starting and ending location[1] (l_T^{start} and l_T^{end}), while the following two represent the starting time and duration of the task (t_T^{start} and α_T). Last, σ_T defines the maximum time a task can be delayed[2]. The resulting ending time t_T^{end} is the starting time plus the task duration, i.e. $t_T^{end} = t_T^{start} + \alpha_T$. All tasks start within $t_T^{start} \in [0, \Lambda] \quad \forall T \in \mathcal{T}$, with Λ defining the end time to request any task. An illustration of a single task, its delayed variant, and all associated points in time is shown in Figure 4.2.

Decision Variables: The decision variables of the FSD problem are the number of vehicles and their trajectories, also defining the required delay per task. A single trajectory is an operational plan for one vehicle. Formally, a trajectory is an ordered sequence of tasks so that it is feasible for a single vehicle to serve all of them within the given maximum delays. We denote a trajectory as $g_i = (T_{i,1}, T_{i,2}, ..., T_{i,k})$, thereby $T_{i,k}$ denotes that task T is part of the trajectory g_i and is served at position k. A trajectory starts at the starting location and at the starting time of the first task it serves, after serving this task, the vehicle relocates from the ending location to the starting location of the subsequent task, if needed the vehicle waits until the task starts and serves it, subsequently the vehicle relocates again. This procedure is continued until the vehicle serves the last task, after which the vehicle stops. If the vehicle arrives at the starting location of a task later than its starting time, the

[1] The starting and ending locations can be different or the same, there is no inherent need to return to the starting location.

[2] Without a maximum delay, a single vehicle can serve the full set of tasks by serving all subsequently.

trip is delayed by the difference. The time a task is actually delayed is denoted as ρ_T. A set of trajectories is summarized in the set ω. To be feasible, ω must fulfill that all tasks are served, i.e. the union of all the trajectories results in $\mathcal{T}$, mathematically $\cup_{i \in \omega} g_i = \mathcal{T}$; and that no task is delayed more than its maximum delay, i.e., $\rho_T \leq \sigma_T \; \forall T \in \mathcal{T}$.

Cost Function: The cost function $C(\omega)$ assigns a cost to a set of trajectories ω. It includes costs for the number of vehicles used (fixed capital costs), their total traveling time between tasks (relocation time), and costs for the total delay added. For each vehicle used, i.e. for each executed trajectory, a capital cost of M_{fix} is charged. To ease notation, we summarize the total relocation time of a single trajectory g_i as $\phi(g_i)$. We denote the relocation time from task i to task j as τ_{T_i,T_j} or short as $\tau_{i,j}$. As such, the total relocation time of a single trajectory can also be expressed as $\phi(g_i) = \sum_{j=1}^{|g_i|-1} \tau_{T_{i,j},T_{i,j+1}}$. Note that the execution time of the vehicles during tasks is not part of the cost function because this is a constant value. Additionally, two weights are used to weigh the influence of the total delay and the total relocation time, which we denote as M_ρ and M_ϕ. Their exact values depend on the specific cost structure of the operation[3], which is also the case for M_{fix}. In Section 4.5, we indicate those values for our experiments. Additionally, two weights for the total delay M_ρ and the total relocation time M_ϕ are used. This results in the objective function as follows:

$$c(\omega) = M_{fix} \cdot |\omega| + M_\rho \cdot \sum_{T \in \mathcal{T}} \rho_T + M_\phi \cdot \sum_{g \in \omega} \phi(g) \tag{4.1}$$

Problem: We define Ω as the set of all sets of trajectories ω that are feasible solutions to the FSD. As such, the fleet sizing problem with delays can be posed as:

$$\min_{\omega \in \Omega} c(\omega) \tag{4.2}$$

Note that an instance of the FSD problem is fully characterized by the graph $G = (V, E)$, the set of tasks $\mathcal{T}$, and the constants M_{fix}, M_ρ, and M_ϕ.

4.3.2. Problem Complexity

Here we analyze the complexity of the newly introduced FSD problem. We first compare the FSD problem on an intuitive level to its closest related problem, fleet sizing without delays. Second, we claim and prove that FSD is NP-hard (Section 4.3.2).

On an intuitive level, introducing delay changes three points that are worth highlighting when comparing FSD to fleet sizing without delays:

- First, allowing delays will result in more options to chain individual tasks. For no delays, each pair of tasks for which the starting location of the subsequent

[3] For example, for taxi rides, delays are more important than for logistical operations, resulting in a higher value of M_ρ.

task can not be reached in time can not be chained. For two tasks i and j of the set of tasks $\mathcal{T}$, this can be expressed as

$$t_i^{end} + \tau_{i,j} \leq t_j^{start} \qquad i \in \mathcal{T}, j \in \mathcal{T}$$

This is relaxed with the option of delaying tasks, resulting in

$$t_i^{end} + \tau_{i,j} \leq t_j^{start} + \sigma_j \qquad i \in \mathcal{T}, j \in \mathcal{T}$$

- Second, the individual decisions on whether to chain a pair of tasks are independent of each other in case of no delays. This follows as the starting and ending times are not changed by chaining. Thus, when deciding to chain task i to task j, it is irrelevant which partial trajectory the corresponding vehicle followed before task i. With delays, this is not the case, as the ending time of a task changes if the task is delayed, which in turn might modify the starting time of all the subsequent tasks assigned to the same vehicle. We remark that this fact precludes utilizing the approach based on a bipartite graph, as done by [12].
- Third, the set of all chaining options may contain a pair of tasks i and j twice, once as (i, j) and once as (j, i).

4

Proof: The Fleet Sizing with Delays Problem is NP-Hard

Here, we prove that the FSD problem is within the complexity class of NP-Hard problems. We do so by showing that the FSD can be reduced to the problem of determining whether a Hamiltonian path exists, which is known to be NP-Complete. A *Hamiltonian path* is defined as a path that visits each vertex of a given directed graph $G = (V, E)$ exactly once. As each vertex is exactly visited once, this implies that the path is exactly of length $|V| - 1$.

Let us consider an instance of the Hamiltonian Path problem $G = (V, E)$. We **prove** that: a Hamiltonian path exists on the graph **if and only if** the FSD problem[4] $(G, \mathcal{T}, M_{fix}, M_\rho, M_\phi)$ admits a solution with a cost equal or lower to $M_{fix} + (|V| - 1 \cdot |V|/2) + |V| - 1$.

Reduction:

$$c(e) = 1 \quad \forall e \in E \tag{4.3a}$$
$$\mathcal{T} = \{T_v\}_{v \in V}, \text{ with:} \tag{4.3b}$$
$$l_{T=v}^{start} = l_{T=v}^{end} = v \quad \forall T \in \mathcal{T} \tag{4.3c}$$
$$t_T^{start} = t_T^{end} = 0 \quad \forall T \in \mathcal{T} \tag{4.3d}$$
$$\sigma_T = |V| - 1 \quad \forall T \in \mathcal{T} \tag{4.3e}$$
$$M_\rho = M_\phi = 1 \tag{4.3f}$$
$$M_{fix} = |V|^2 + |V| > (|V| - 1 \cdot |V|/2) + |V| - 1 \tag{4.3g}$$

[4] One instance of the FSD is fully characterized by $G = (V, E), \mathcal{T}, M_{fix}, M_\rho, M_\phi$ (see Section 4.3.1).

The cost to traverse any edge of the graph is set to one (Equation 4.3a). Equation 4.3b specifies that there is exactly one task per vertex in the graph ($|V| = |\mathcal{T}|$); this allows us to index the tasks using the vertices. For each of these tasks, the start and the end locations are at the corresponding vertex (Equation 4.3c). Equation 4.3f sets the weights of added delay and relocation time equally to one. Equation 4.3g specifies that M_{fix} is larger than a given threshold, ensuring that reducing the fleet size is always more important than the other parts of the objective function.

To prove the above-posed claim, we need to verify the claim in both ways. First, that if the FSD is solvable with a cost less or equal than $M_{fix} + (|V| - 1 \cdot |V|/2) + |V| - 1$, then a Hamiltonian path exists on G. Second, if a Hamiltonian path exists on the graph G, then the FSD can be solved with a cost less or equal to $M_{fix} + (|V| - 1 \cdot |V|/2) + |V| - 1$.

Direction 1: *is solvable with a cost less or equal than* $M_{fix}+(|V|-1\cdot|V|/2)+|V|-1$ $\Rightarrow$ *Hamiltonian path exists on G:*
A solution to the FSD with the target costs of $M_{fix}+(|V|-1\cdot|V|/2)+|V|-1$ is only achievable under specific conditions. First, only a single vehicle can be used, causing costs of M_{fix}. Using an additional vehicle would add costs of M_{fix}, leading to total costs larger than $2M_{fix}$, which is, in turn, larger than the target costs (Equation 4.3g), $2M_{fix} > M_{fix} + (|V| - 1 \cdot |V|/2) + |V| - 1$. Second, a solution of FSD must serve all tasks, and therefore, all vertices of the graph are visited, as each task takes place in exactly one vertex. That is, the existence of the solution with the said costs ensures that a single vehicle is able to visit all of the nodes within the maximum allowed delays. This clearly ensures that a Hamiltonian Path must exist: otherwise, the vehicle would need to traverse at least V arcs, violating the maximum delay for the last visited task.

Direction 2: *A Hamiltonian path exists on graph* $G \Rightarrow$ *the FSD can be solved with a cost less or equal than* $M_{fix} + (|V| - 1 \cdot |V|/2) + |V| - 1$*:*
The desired solution is found by making a single vehicle follow exactly the Hamiltonian Path. Each vertex is visited by a Hamiltonian path; thus, each task gets served. The first task is served at $t = 0$. The next task is met a time unit later as a Hamiltonian path will traverse exactly one edge to visit the next vertex. This pattern continues till the last task is served at $t = |V| - 1$, thus satisfying the constraints. As a Hamiltonian path traverses $|V| - 1$ edges, each having the cost of one, this results in costs for total relocation time and delay of exactly $(|V|-1\cdot|V|/2)+|V|-1$. As a single path is formed, a single vehicle is required, adding costs of M_{fix}. As such, A Hamiltonian path is a solution to the FSD problem and satisfies the cost cap.

Conclusion: As the FSD problem simplifies, under the above-described reduction, to the problem of whether a Hamiltonian path exists or not, which is NP-Complete, we conclude that the FSD problem is NP-Hard.

4.4. Method

4.4.1. Mixed Integer Linear Problem

Here, we formalize the FSD as a MILP based on the idea of *chaining*. We connect the individual tasks into chains, each forming a trajectory g for one vehicle. For a single vehicle to be able to serve two tasks i and j in succession, it needs to be able to relocate from the ending location of task i to the starting location of task j and arrive before its starting time plus the maximal delay. This can be expressed as:

$$t_i^{end} + \tau_{i,j} \leq t_j^{start} + \sigma_j \qquad i \in \mathcal{T}, j \in \mathcal{T} \tag{4.4}$$

Recall, $\tau_{i,j}$ denotes the relocation time from task i to task j. It is calculated as the sum of all costs of traversed edges if following the shortest path in between the ending location of task i and the starting location of task j.

We define the set of all pairwise feasible chaining options of tasks as $\mathcal{X}$, it contains all pairs of tasks (i, j) that fulfill Equation 4.4.

To formulate this idea into a tractable problem, we introduce an integer decision variable $x_{i,j}$ for each pair of tasks (i, j). $x_{i,j}$ takes the value of 1 if task i and j are served by one vehicle in succession (i before j), i.e. the two tasks are chained together. Otherwise, it takes the value of 0.

Taking the cost as defined previously (Equation 4.1), we award a constant value of $-M_{fix}$ for every two tasks that are chained.[5] Further, we penalize delay and total relocation time (total driven time between tasks). This allows us to formulate the problem as a MILP as follows, using the newly introduced decision variable and notation:

$$\min_{x,\rho} \sum_{(i,j)\in\mathcal{X}} x_{i,j} \cdot \Big[-M_{fix} + M_\phi \cdot \tau_{i,j} \Big] + M_\rho \cdot \sum_{i\in\mathcal{T}} \rho_i \tag{4.5a}$$

$$\sum_{j=1}^{\mathcal{T}} x_{i,j} \leq 1 \quad \forall i \in \mathcal{T} \tag{4.5b}$$

$$\sum_{i=1}^{\mathcal{T}} x_{i,j} \leq 1 \quad \forall j \in \mathcal{T} \tag{4.5c}$$

$$0 \leq \rho_i \leq \sigma_i \quad \forall i \in \mathcal{T} \tag{4.5d}$$

$$x_{i,j} \cdot \rho_i \leq x_{i,j} \cdot \Big[t_j^{start} + \rho_j - t_i^{end} - \tau_{i,j} \Big] \quad \forall i \in \mathcal{T},\ \forall j \in \mathcal{T} \tag{4.5e}$$

Equation 4.5a describes the objective function of the MILP. $-M_{fix}$ is awarded each time two tasks are chained. Further, we add the time the vehicles need to drive between the two tasks ($\tau_{i,j}$) and the sum of the delay of all tasks ($\rho_i\ \forall i \in \mathcal{T}$). Note that the minimization problem can yield a negative value for the objective function

[5] Each chained formed means one vehicle less is required.

due to awarding $-M_{fix}$, if two tasks are chained. The trivial feasible solution where all variables are zero (no chaining and no delays) yields an objective value of zero, which is not optimal. Equation 4.5b ensures that each task is maximally chained to one subsequent task. Equation 4.5c ensures that each task has a maximum of one preceding task. Note that the Equations 4.5a to 4.5c recover the traditional ILP for fleet sizing if no delays are allowed. A constraint as Equation 4.5e can be omitted in the no-delay case, as for no delays ($\rho = 0$), it is always trivially fulfilled because all (i, j) belong to $\mathcal{X}$, hence satisfying Equation 4.4 which is equivalent to Equation 4.5e for no delays. The set $\mathcal{X}$ does not contain any decision variable and, as such, can be calculated beforehand. Equation 4.5d ensures that the actual delay of a task (ρ_i) is not larger than its maximum delay (σ_i) and not smaller than zero. Equation 4.5e ensures that a vehicle arrives at the next task in time. To be precise, it means that if $x_{i,j} = 1$, the vehicle leaves the final location of i at its ending time plus its potential delay ($t_i^{end} + \rho_i$), drives to the new location taking time $\tau_{i,j}$, and arrives in time to the starting position of task j, including potential added delay of this task ($t_j^{start} + \rho_j$); if $x_{i,j} = 0$, this constraint has no implication. An illustration of this constraint is shown in Figure 4.3.

Unfortunately, the constraint formulated in Equation 4.5e is not linear. To linearize it, we define two new auxiliary variables $A_{i,j}$ and $B_{i,j}$ and apply a temporal scaling to ensure $\sigma \leq 1$. Intuition-wise, the new variables represent $A_{i,j} = x_{i,j} \cdot \rho_i$ and $B_{i,j} = x_{i,j} \cdot \rho_j$. As this would not be linear, these equations are represented by the four Inequalities 4.6a-4.6d and respectively 4.6e-4.6h. The temporal scaling ensures that equations 4.6c and 4.6g can always be satisfied also for the case of $x_{i,j} = 0$.

$$A_{i,j} \leq x_{i,j} \tag{4.6a}$$
$$A_{i,j} \leq \rho_i \tag{4.6b}$$
$$A_{i,j} \geq x_{i,j} + \rho_i - 1 \tag{4.6c}$$
$$A_{i,j} \geq 0 \tag{4.6d}$$
$$B_{i,j} \leq x_{i,j} \tag{4.6e}$$
$$B_{i,j} \leq \rho_j \tag{4.6f}$$
$$B_{i,j} \geq x_{i,j} + \rho_j - 1 \tag{4.6g}$$
$$B_{i,j} \geq 0 \tag{4.6h}$$

Two cases can occur: First, $x_{i,j} = 0$, as such Equation 4.6a demands $A_{i,j} = 0$. Second, if $x_{i,j} = 1$ then $A_{i,j} = \rho_i$ forced by the Equations 4.6b and 4.6c. The same holds for $B_{i,j}$ and ρ_j. Using the auxiliary variables, Equation 4.5e can be rewritten in linear form, see Equation 4.7.

$$A_{i,j} \leq B_{i,j} + x_{i,j} \cdot \left[t_j^{start} - t_i^{end} - \tau_{i,j} \right] \tag{4.7}$$

All used notation of this chapter is summarized in Table 4.1 in Appendix 4.7.1.

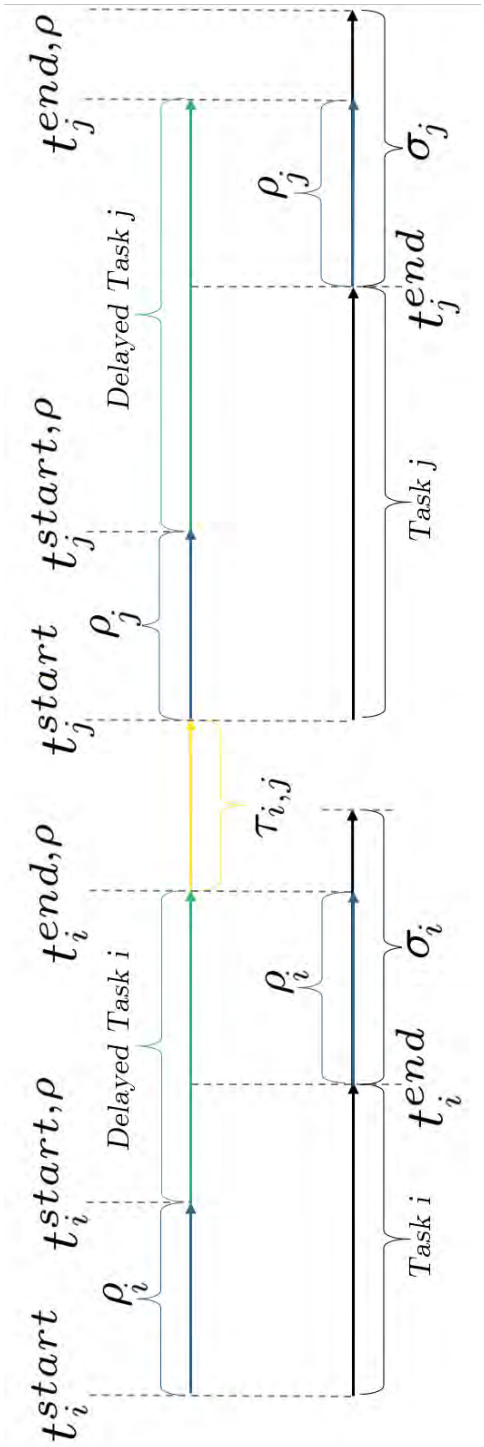


Figure 4.3: Illustration clarifying the constraint of Equation 4.5e. Two tasks, i and j, their delayed variants, and the relocation time between are shown. A vehicle follows the top line of arrows if it serves delayed task i relocates to task j, which was also delayed by ρ_j.

4.4.2. Heuristics

As the size of the problem can get too large to be solvable through the above-proposed method, or the required solving time can get too vast due to its NP-Hard nature, we propose some heuristics for keeping the problem smaller. All heuristics are designed to reduce the number of potential chains, $|\mathcal{X}|$, needed to be considered by the optimizer or by tightening the given constraints. We propose the following four heuristics:

- Artificially reducing the maximally allowed delay time by introducing a new maximum delay variable ρ_{max}

$$\rho_i \leq \rho_{max} \;\; \forall i \in \mathcal{T} \qquad \rho_{max} < \max_{i \in \mathcal{T}} \sigma_i$$

- Only allowing tasks to be chained if the end location of the ending task is less than a given relocation time, τ_{max}, apart from the start location of the following task.

$$\tau_{i,j} \leq \tau_{max} \;\; \forall i \in \mathcal{T}, \forall j \in \mathcal{T} \qquad \tau_{max} < \max_{i,j \in \mathcal{T}} \tau_{i,j}$$

- For each starting task i, we limit the number of pairs $(i, j) \in \mathcal{X}$ to a maximum of z pairs. Tasks are ranked by cost, and the z bests are considered. Here, costs are calculated as $\tau_{i,j} + \rho_{i,j}$, with $\rho_{i,j}$ being the minimal delay if the tasks are considered in isolation. z is a tuneable parameter.

- Due to allowed delays, it can happen that $(i, j) \in \mathcal{X}$ and also $(j, i) \in \mathcal{X}$. This heuristic consists of forbidding this situation, by only allowing trips to be chained if trip i ends before trip j, i.e. $t_i^{end} < t_j^{end}$. We refer to this heuristic as the "NoDuplicates"-heuristic.

4.4.3. Solving the Mixed Integer Linear Problem

For practical reasons, the MILP actually solved differs slightly from the one posed in Equation 4.5. Equation 4.5e is replaced with its linear variant, Equation 4.7. Further, we add a constraint for each pair of tasks, which can not be chained. The corresponding decision variable $x_{i,j}$ is set to zero from the beginning, i.e.

$$(i, j) \notin \mathcal{X} \rightarrow x_{i,j} = 0$$

We solve the resulting problem using the Gurobi solver version 9.5.2 [102]. If heuristics are applied, the problem to solve alters slightly. If setting a new artificial delay bound (ρ_{max}), Equation 4.5d is adapted accordingly. The other three heuristics add an additional check while calculating the set of potential chaining options $\mathcal{X}$, which is used as described above.

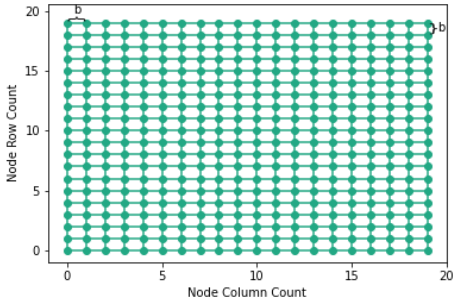


Figure 4.4: A graph representing the street network of the Gridworld environment $n = 20$, thus containing 400 vertices, is depicted. For a vehicle to traverse any edge takes 10 seconds, $b = 10$.

4.5. Experiments and Results

4.5.1. Overview

This section presents our experimental results and is structured as follows: First, we analyze a theoretical example called Gridworld in detail. We compare runs with and without delay, analyze achievable trade-offs between fleet size, total relocation time, and delay, compare different scenarios varying the density of tasks, and analyze the introduced heuristics. Second, we investigate a case study of pooled taxi rides in Manhattan. All computations have been performed on a PC with an AMD Ryzen 5 5600X and 64GB Memory. All resulting MILPs were solved using Gurobi 9.5.2 [102].

4.5.2. Gridworld

The operational environment, which we call *Gridworld*, is a grid-like structure consisting of $n \times n$ vertices arranged in a square. Neighboring vertices are connected and equidistantly spaced with a travel time of b seconds[6]. One instance of Gridworld is fully characterized by n and b. An example of Gridworld is depicted in Figure 4.4. Gridworld has a maximal distance between vertices of $2 \times (n-1) \times b$ seconds. For tasks' start and end locations (l_T^{start} and l_T^{end}), we randomly (uniformly) chose two vertices from the environment graph. Start times t_T^{start} are sampled uniformly within the starting time $t = 0$ and the end of the operation Λ. In Gridworld, we define the duration of a task α_T as the time it takes to travel the shortest path from the start to the end location of the task[7]. As such, the ending time t_T^{end} is determined. This results in an average task length in Gridworld of $\frac{4}{6} \times n \times b$ seconds. Within this section, we use the following parameter settings, unless mentioned otherwise: $n = 40$; $b = 10$; $|\mathcal{T}| = 1,600$; $\sigma_T = 480$; $\Lambda = 8 \cdot 3,600$, $M_\rho = M_\phi = 1$. The parameters have been chosen to be realistic while keeping the required computational times at a reasonable scale. We sampled and solved 5 different demand

[6] This is a simplifying assumption that we consider as reasonable as the proposed approach can accommodate real travel times if available. Real travel times have an effect of the specific parameters of tasks, but not on their general definition nor on the proposed method.

[7] This definition does not harm the generalization of the method and is straightforward to adapt.

scenarios. The mean and standard deviation are reported. The density analysis and the heuristics are analyzed using a single case. Unless mentioned otherwise, no heuristics are applied for all calculations within this section, and the problem at hand is solved to optimality (no time limit and an optimality gap of 0.0001).

Comparison against the no-delay case

First, we compare solutions to the FSD (allowing to delay individual tasks) to the traditional fleet sizing problem (not allowing delays). We do so for different values of M_{fix} (Equation 4.5a), as it shapes the number of vehicles used. Generally, the amount of extra delay and extra traffic to be accumulated to beneficially decrease the fleet size by one vehicle is capped at M_{fix}, as otherwise the objective value (Equation 4.5a) would increase and thus not chaining is better. Thus, a lower absolute value of M_{fix} will lead to a less strong decrease in fleet size. Thereby, for no delays, the value of M_{fix} that equals the largest relocation time between any two vertices is of major importance. For the scenario analyzed here, this value equals $M_{fix} = 780$. A value of $M_{fix} \geq 780$ leads to a minimal fleet without delays, as it is always beneficial to chain two tasks if possible. The threshold value of M_{fix} to not decrease the fleet size at all, in other words, using an individual vehicle per task, depends on the analyzed scenario. The threshold equals the minimum in the sum of relocation time plus the potentially required delay of any two tasks that can be chained.

Figure 4.5 shows a direct comparison of runs with and without allowing delays for $M_{fix} \in [400, 600, 800]$. All results are also listed in Table 4.2 in Appendix 4.7.2. The obtained fleet sizes, total relocation times, added minor delays, as well as the difference in costs, are shown. Runs without delays are depicted in brighter colors. Allowing for delays reduces the fleet size for all values of M_{fix}. The required fleet sizes are reduced by around 1 vehicle for $M_{fix} = 800$, by around 2 vehicles for $M_{fix} = 600$, and by around 1 vehicle for $M_{fix} = 400$. This showcases that allowing delays is beneficial to decrease fleet sizes.[8] This comes at the price of increasing total relocation time and added delay. Respectively a total delay of 14:46 ± 02:51, 11:16 ± 02:02, and 05:50 ± 00:59 minutes and seconds is added. The average delay per task is less than one second for all runs. The total amount of added delay, in comparison to the total relocation time, is small. Please also note that the driving time of the vehicles during tasks is not depicted nor part of the cost function, as it is a constant. The obtained total costs, the objective of the optimization problem, are smaller (minimization problem) if delays are allowed. As such, it is beneficial to consider potential delays in fleet sizing regardless of the importance of single objectives as seen by the overall achievable lower costs.

[8] Larger decreases in fleet size can be obtained for larger values of M_{fix}, see below (Figure 4.6).

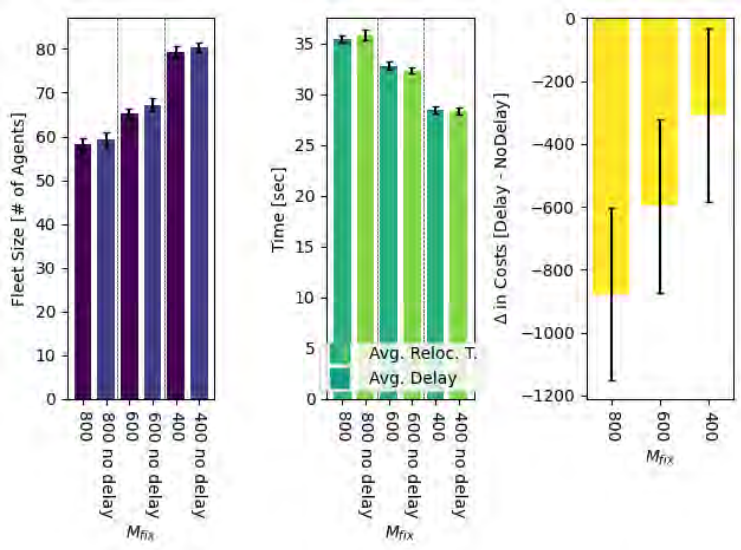


Figure 4.5: Comparison of runs with and without allowing to delay tasks for 3 values of $M_{fix} \in [400, 600, 800]$, scenarios without delay are displayed in lighter colors. Allowing delays decreases the obtained fleet size (purple) by adding some minor delay (dark green). Total objective values (yellow) are lower if delays are allowed (minimization problem). The mean of 5 scenarios is shown, including their standard deviation shown using black bars.

Further Reduction in Fleet Size

In the previous section, we showed the potential of solving the FSD in direct comparison to not considering the option to delay tasks. Here, we highlight the potential of considering delaying tasks to decrease the fleet size even more. When no delays are allowed, the threshold $M_{fix} \geq 780$ ensures that fleet size is always the primary objective. When delays are allowed, larger values of M_{fix} yield even smaller fleets, as chaining decisions are not independent of each other anymore. Consequently, we analyze $M_{fix} > 780$ within this section.

Figure 4.6 shows obtained results for values of M_{fix} from 1,000 to 6,000, in steps of 1,000, and for comparison, the minimum achievable fleet without delays (using $M_{fix} = 800$, same as previous section). Obtained fleet sizes, total relocation time, and added delay are shown. Additionally, results are listed in Table 4.3 in Appendix 4.7.2. All runs are able to achieve a smaller fleet at the price of increasing total relocation time and added delay compared to the non-delay case. The higher M_{fix}, the smaller the obtained fleet size, accompanied by higher sums of total relocation time and added delay. This is expected as the sum of costs (delaying tasks and longer relocation times) can be more to include more tasks within each chain. Regarding fleet size, the absolute number of vehicles saved for increasing M_{fix} more and more diminishes slowly. Comparing the required fleet sizes for $M_{fix} = 6,000$ to the minimum fleet size without delays shows a decrease of needed vehicles of more than 50% (from 59.2 to 28.6 vehicles), at a cost of just 9.7 seconds of average delay per task. Average relocation time increases from 35 seconds to 1:03 minute and seconds. The

average relocation time increases from 34 seconds to 1 minute and 3 seconds.

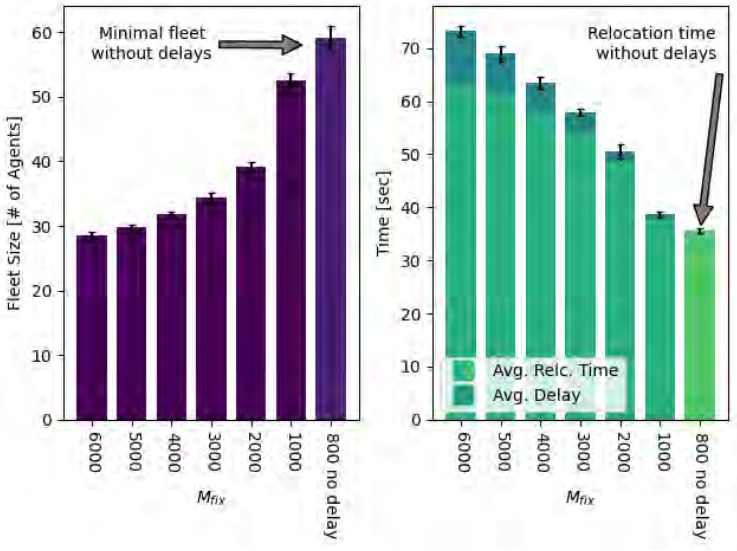


Figure 4.6: Figure showing the effect of the value of M_{fix} on the obtained fleet size and the average delay and average relocation time. For comparison, a run with no delay and only optimizing on fleet size is shown. The mean of 5 scenarios is shown, including their standard deviation shown using black bars.

In conclusion, FSD increases the potential trade-off space and, thus, the potential to decrease the number of required vehicles significantly. This potential is huge but heavily depends on the relation between costs involved, which in turn depends on the nature of the operation at hand.

Density analysis

This section analyses the effect if the number of potential chains is varied and shows that obtainable benefits are robust against external changes.
The number of potential tasks that a vehicle could serve after finishing a task (the number of chaining options) increases if more tasks need to be fulfilled (i.e. if we increase $|\mathcal{T}|$). The same effect occurs the smaller the environment is, as the average relocating time becomes smaller (which we analyze by changing n), and thus it is easier to reach the new starting location in time. To analyze this effect on the proposed method, we vary $|\mathcal{T}|$ and n. We compare runs with no delay, fully minimizing fleet size, to runs allowing delays with $M_{fix} = 5,000$. We vary the size of the environment n as [20,40,60]. As for the number of placed tasks, we use different numbers of total tasks to fulfill, $|\mathcal{T}| \in$ [800, 1,200, 1,600, 2,000, 2,400]. Obtained results are visualized in Figure 4.7. The differences in fleet size (left) and delay plus relocation time (right) for each value pair are shown for the scenarios with and without delay. Values are shown in percentage, the baseline is the corresponding run with no delays, fully minimizing fleet size.

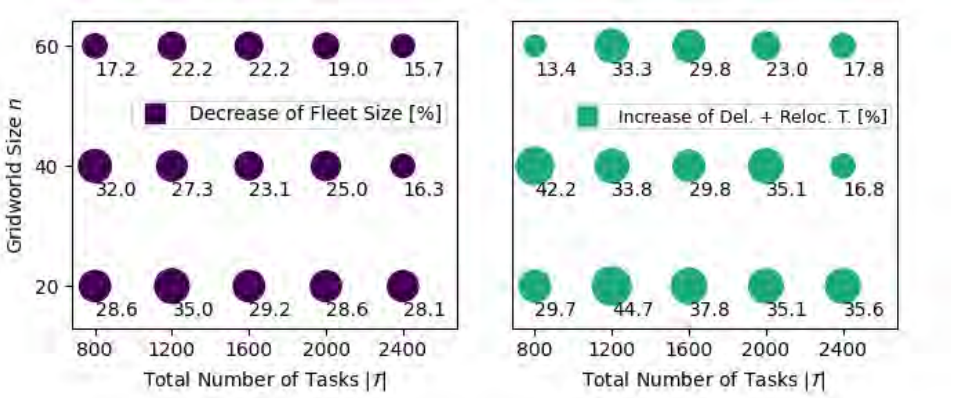


Figure 4.7: The difference in fleet size and total relocation time plus delay for allowing delays to no delays is shown in percentage as the environment size and the number of placed tasks are varied.

The decrease in fleet size ranges from 15.7% to 35%. This decrease is always significant. The increase in total relocation time plus delay lies between 13.4% and 44.7%. Generally, greater changes accompany each other, seen by similar sizes of the corresponding dots in Figure 4.7. No other clear trends show.

Effect of Heuristics

To enhance the scalability of the proposed approach to solve the FSD, we introduce a variety of heuristics in Section 4.4.2. They reduce computational cost while not significantly compromising the quality of the results. This capability is particularly advantageous when tackling larger scenarios. In this section, we apply these heuristics and analyze their effectiveness. Additionally, we vary the strengths of the applied heuristics. For each heuristic, we compare fleet size, total relocation time plus delay, the required computation time for the optimization, and the number of chaining options, $|\mathcal{X}|$. As a baseline for comparison, we use the case with M_{fix} set to 5,000. For ease of comparison, in all figures of this section, all numbers are displayed in percentages relative to the baseline.

Restricting maximum allowed delay: We restrict the maximum allowed delay form $\sigma_i = 480\,\forall i \in \mathcal{T}$ to $\rho_{max} \in [360, 240, 120]$. Results are illustrated in Figure 4.8. Reducing the maximally allowed delay can be effective in decreasing run times. Figure 4.8 shows that the heuristic only slightly increases the fleet size and relocation time plus delay for a strong decrease in required run times. But, if the heuristic is applied strongly, i.e. for lower values of $\rho_{max} = 120$, we see a substantial increase in fleet size.

Restricting maximum allowed relocation time: We restrict the upper limit on the relocation time between two tasks to $\tau_{max} \in [600, 400, 200, 100]$. Results are displayed in Figure 4.9. For larger values of τ_{max}, the fleet size does not increase, but the heuristic is also not effective in reducing required run times. Reducing the

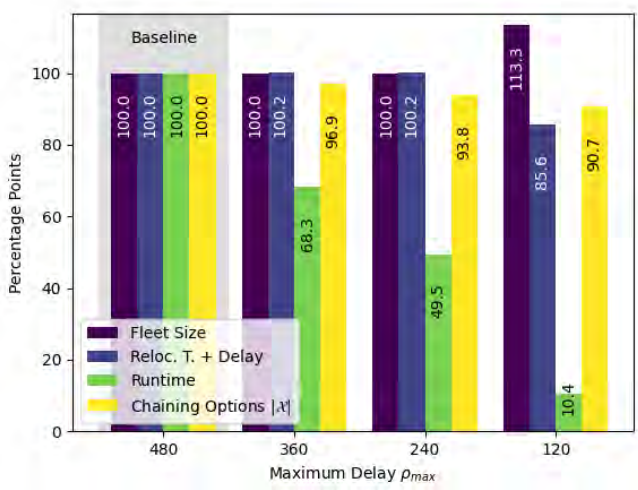


Figure 4.8: Visualization of the effect of reducing the maximal allowed delay ρ_{max} on fleet size, delay plus total relocation time, run time, and the number of chaining options $\mathcal{X}$, all presented as percentages of the run without any heuristics.

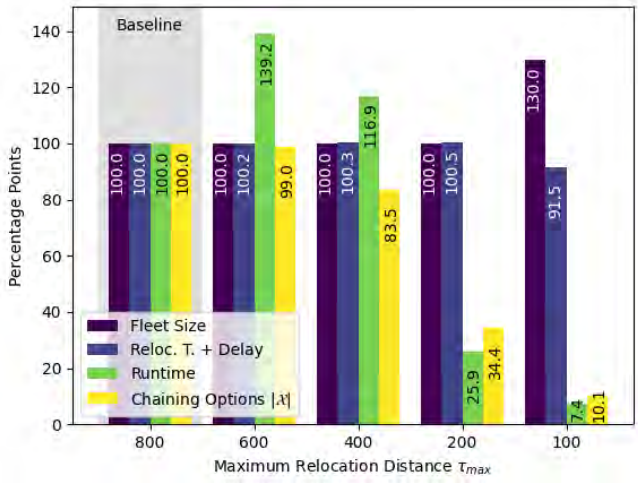


Figure 4.9: Visualization of the effect of reducing the maximal allowed relocation time τ_{max} on fleet size, delay plus total relocation time, run time, and the number of chaining options $\mathcal{X}$, all presented as percentages of the run without any heuristics.

maximum relocation time strongly, $\tau_{max} \in [200, 100]$, greatly decreases the number of chaining options and a decrease in run times. For $\tau_{max} = 200$ the fleet size does not change and total relocation time and delay only increase slightly. In contrast, the fleet size increases strongly for $\tau_{max} = 100$. As such, the heuristic needs to be well-tuned to be effective.
Comparing this heuristic to the previous one, we see a notable difference. Reducing the maximum delay decreases the number of potential chaining options only slightly but significantly affects the required computation time. In contrast, reducing the maximum relocating time reduces the number of potential edges in a stronger fashion but with a smaller effect on the computational time. This indicates that the complexity of the problem is within the newly allowed delay.
Restricting the maximum number of chaining options per task to z: We varied $z \in [250, 200, 150, 100]$. Most notably, the run time increases for all values of z. As such, the heuristic is ineffective, and the sum of relocation times and minimum required delay is ineffective in judging the potential of chaining options. Results are shown in Figure 4.11 in Appendix 4.7.3.
"NoDuplicates"-Heuristic: This heuristic can either be used or not (no additional tuning). The total number of chaining options only decreases very slightly (about 1.3%). The run time increases, and as such, we do not recommend the use of this heuristic. Results are shown in Figure 4.12 in Appendix 4.7.3.

4.5.3. Case Study: Manhattan

This section analyses a real-world instance to analyze the proposed method's potential. We investigate the FSD problem considering taxi rides in Manhattan.
To build the set of tasks $\mathcal{T}$, we utilize one hour of taxi rides data in Manhattan[9] [103]. 19,809 individual transportation requests are placed within this hour. The tasks we consider are not these individual requests but groups of them that are pooled to travel together. Here, the pooling of requests is done following a method based on the Vehicle-Group-Assignment method by [11]. Details on this step can be found in Appendix 4.7.4. The ending time of the transportation tasks is based on the task's duration, determined by the total travel time to serve all passengers following the shortest path. This pooling step leads to a total of 4,255 pooled transportation tasks or trips starting within 1 hour. Figure 4.13 in Appendix 4.7.5 shows distributions of the starting times, trip duration, ending times of the tasks, as well as the number of passengers per task. As the operational environment, Manhattan's road network is represented by a graph, and travel times are estimated for each road segment. These travel times are used for transporting passengers and empty relocation. The graph was obtained following the method described in [104]. Thereby, the travel times of the graph are estimated based on the departure and arrival times of the recorded trips and averaged over a day. The average relative error of the actual travel times to the estimated travel times is minimized.
We run the above-proposed method for the Manhattan instance of the FSD problem with the following settings. For the cost function, we set $M_{fix} = 2,500$ and $M_\phi = M_\rho = 1$. Based on the heuristic analysis for Gridworld (Section 4.5.2),

[9] The used data was recorded on 29.05.2013 between 1 p.m. and 2 p.m.

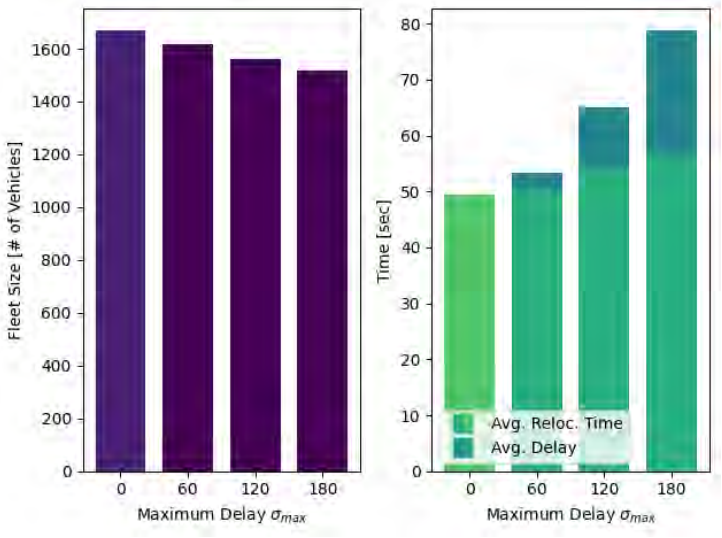


Figure 4.10: Main results of fleet sizing of one hour of pooled taxi rides in Manhattan, in regards to fleet size, average relocation time, and average added delay, are displayed. For higher maximal allowed delay, the required fleet sizes decrease, accompanied by an increase in total relocation time and added delay.

we applied the heuristic capping the maximum relocating time, which we set to $\tau_{max} = 800$ seconds. We do not apply the "NoDuplicates"-heuristic and the heuristic selecting chaining options based on cost. For the maximum delay of all trips, we analyze three scenarios of $\sigma \in [60, 120, 180]$ seconds, which is at the core of this chapter. All results are visualized in Figure 4.10 and listed in Table 4.4 in Appendix 4.7.2.

Each extra minute of allowed delay reduces the minimum required fleet size by about 50 vehicles (52/54/47); in percentage, this is a decrease of around 3% (3.12% / 3.34% / 3.01%). Added delay and relocating time increase, whereby the stronger changes are within the added delay. If 1 minute of maximum added delay is allowed an average delay of 3.2 seconds per task is added. For 2 and 3 minutes of maximal delay, this equals 10.7 and 22.37 seconds per task. This equals 8.41/29.2/62.78 seconds of delay for each vehicle over its entire day.
To conclude, introducing the option of delaying tasks is a way to reduce the number of required vehicles. A slight modification, consisting of the introduction of 1 minute of maximum delay, already allows decreasing the fleet by 3.12% at the mild cost of an average delay of 12 seconds per trip. The proposed method allows stressing this trade-off significantly further, allowing up to 3 minutes of delay, decreasing the required fleet by 9.17% compared to not allowing delays.

4.6. Conclusion

In this chapter, we have posed the problem of *fleet sizing with delays*. It extends the question of "How many vehicles are needed to perform a set of tasks?" by allowing short delays before the beginning of each task, noting that such delays are commonly observed in real-life applications. We proved that the new problem is NP-Hard, and proposed a formulation of the FSD as a MILP. The formulation used is general, and as such, the methods and findings have wide applicability. First, we studied a general case (Gridworld) followed by a real-world case of shared taxi rides in Manhattan.

For Gridworld, we showed that it is beneficial to consider delays within fleet sizing. Lower costs are achieved in direct comparison to not considering them. We further showed that delays allow a decrease in fleet size beyond previous limits. If a small fleet is the highest priority, we find that fleets can be decreased by 50% compared with the minimal fleet without delay while still respecting small limits of maximal delay per task. Additionally, in this chapter, various heuristics are proposed and analyzed. The work is concluded with a real-life case study of taxi rides in Manhattan. Results show the same nature and potential. By adding a maximum of 3 minutes of additional delay per ride, the required fleet size can be decreased by about 9% in comparison to the minimal fleet without delays, at the cost of fewer than 23 seconds of average delay per task.

In summary, using delays improves solutions and increases the option space for potential trade-offs. It allows a reduction of the number of vehicles used strongly.

Future work includes the introduction of delays into other types of approaches, the application of the proposed general method to specific fields and cases other than the mobility of people, and the development of potent heuristics, such as a local search algorithm or tabu search.

4.7. Chapter Appendix

4.7.1. Notation

Notation	Explanation
G	Graph, encoding the environment
V	Set of vertices of the graph, each vertex represents a location, customers order to
E	Set of edges connecting the vertices of the graph
$c(e)$	Costs (time) required to traverse edge e
$\mathcal{T}$	Set of all tasks
T	a single task $T = (l_T^{start}, l_T^{end}, t_T^{start}, \alpha_T, \sigma_T)$
t_T^{start}	Starting time of task T
t_T^{end}	Finishing time of task T
α_T	Duration of a task T
l_T^{start}	Starting location of task T
l_T^{end}	Ending location of task T
Λ	Ending time of the operation, $t_T^{start} \leq \Lambda$
σ_T	Maximum slack of task T
ρ_T	Time task T is delayed
$t_T^{start,\rho}$	Starting time of the delayed task T
$t_T^{end,\rho}$	Ending time of of the delayed task T
g_i	Trajectory i, which is defined as an ordered set of tasks $g_i = (T_{i,1}, T_{i,2}, ..., T_{i,k})$
ω	Set of trajectories
Ω	Set of all feasible sets of trajectories
$c(\omega)$	Cost function
$\mathcal{X}$	Set of all pairs of tasks (i, j), which are chainable
$x_{i,j}$	Binary decision variable, whether tasks i and j are chained
M_{fix}	Constant in the objective function awarded for chaining two tasks
M_ρ	Weight in the cost function to weigh the total delay
M_c	Weight in the cost function to weigh the total relocation time
$\phi(g)$	Sum of times traveled by a vehicle between tasks of trajectory g
$\tau_{i,j}$	Relocation time from task i to task j
$A_{i,j}$	Auxiliary variable to linearize Equation 4.7
$B_{i,j}$	Auxiliary variable to linearize Equation 4.7
ρ_{max}	Artificial maximum delay
τ_{max}	Artificial maximum relocation time
z	Tuneable parameter for selecting the z best edges per task
n	Side length (in number of vertices) of the squared experimental environment Gridworld
b	Distance in seconds between two connected vertices of the experimental environment Gridworld

Table 4.1: Table summarizing the complete notation used throughout this chapter.

4.7.2. Result Tables

M_{fix}	Max Delay σ [sec]	Fleet Size	Total Reloc. Time [h:min:sec]	Total Added Delay [h:min:sec]	Costs
800	480	58.2 ± 1.47	15:33:04 ± 0:05:59	0:14:46 ± 0:02:51	-1176569.4 ± 910.13
800	0	59.2 ± 1.83	15:49:08 ± 0:13:01	0:00:00 ± 0:00:00	-1175692.0 ± 932.98
600	480	65.2 ± 1.17	14:15:46 ± 0:08:16	0:11:16 ± 0:02:02	-868857.2 ± 660.19
600	0	67.2 ± 1.47	14:17:00 ± 0:06:47	0:00:00 ± 0:00:00	-868260.0 ± 666.84
400	480	79.4 ± 1.36	12:24:38 ± 0:08:36	0:05:50 ± 0:00:59	-563211.2 ± 524.8
400	0	80.4 ± 1.2	12:28:56 ± 0:10:21	0:00:00 ± 0:00:00	-562904.0 ± 474.7

Table 4.2: Comparison of the fleet sizing problem allowing delays and not doing so. These results are visualized in Figure 4.5. We list the total delay instead of the average delay per task here, as the averages are all less than one second.

M_{fix}	Max Delay σ [sec]	Fleet Size	Total Reloc. Time [h:min:sec]	Total Added Delay [h:min:sec]
6000	480	28.6 ± 0.49	1 day, 4:04:40 ± 0:07:56	4:19:54 ± 0:19:30
5000	480	29.8 ± 0.4	1 day, 3:21:56 ± 0:20:36	3:16:15 ± 0:18:07
4000	480	31.8 ± 0.4	1 day, 1:45:28 ± 0:15:27	2:31:02 ± 0:16:03
3000	480	34.4 ± 0.8	1 day, 0:05:00 ± 0:08:08	1:26:02 ± 0:07:55
2000	480	39.2 ± 0.75	21:33:02 ± 0:27:54	0:42:19 ± 0:09:07
1000	480	52.6 ± 1.02	16:49:20 ± 0:12:52	0:20:33 ± 0:03:05
800	0	59.2 ± 1.83	15:49:08 ± 0:13:01	0:00:00 ± 0:00:00

Table 4.3: Results table for all experiments using values of M_{fix} greater than 800, and for comparison, the scenario achieving a minimal fleet for no delays. These results are visualized in Figure 4.6.

M_{fix}	Max Delay ρ_{max} [sec]	Max Reloc. Time τ_{max} [sec]	z best	"No Duplicates"	Fleet Size	Avg. Reloc. Time per Trip [min:sec]	Avg. Added Delay per Trip [min:sec]	Runtime [h:min:sec]	Chaining Options $\lvert\mathcal{X}\rvert$
2,500	0	800	x	✓	1,669	0:49	0:00	0:00:26	1,312,052
2,500	60	800	x	✓	1,617	0:50	0:03	0:01:10	1,371,178
2,500	120	800	x	✓	1,563	0:54	0:10	0:08:05	1,432,028
2,500	180	800	x	✓	1,516	0:56	0:22	47:05:04	1,494,541

Table 4.4: Results table containing all results for the scenarios analyzed in the case study of taxi rides in Manhattan.

4.7.3. Plots of Heuristic Experiments

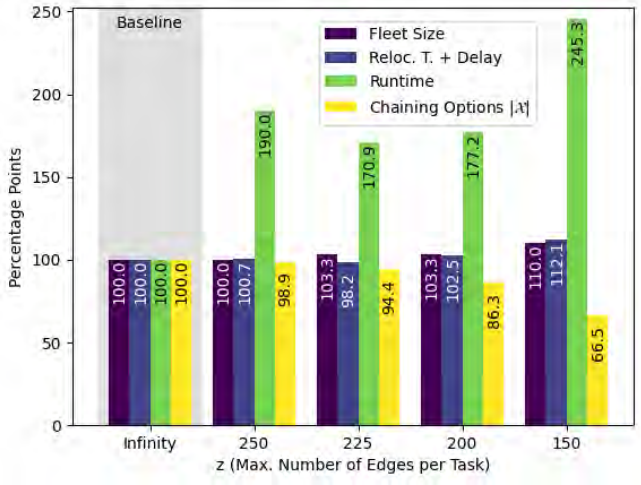


Figure 4.11: Visualization of the effect of restricting the maximum number of chaining options per trip on fleet size, delay plus total relocation time, run time, and the number of chaining options $\mathcal{X}$, all presented as percentages of the run without any heuristics.

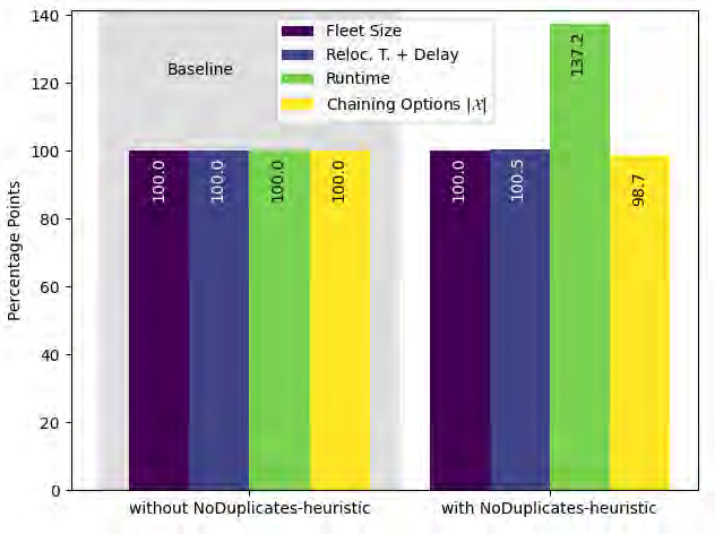


Figure 4.12: Visualization of applying the "NoDuplicates"-heuristic on fleet size, delay plus total relocation time, run time, and the number of chaining options $\mathcal{X}$, all presented as percentages of the run without any heuristics.

4.7.4. Details on Vehicle Group Assignment for Pooling

To create the set of tasks or rides $\mathcal{T}$ used within the case study of Manhattan, the individual transportation requests within Manhattan have been pooled into tasks $T \in \mathcal{T}$ first. This *pooling step* was done leveraging a method close to the VGA [11]. VGA is a receding horizon approach. Each time step, all potential trajectories for a set of given vehicles are calculated. A trajectory defines which requests one vehicle serves and in which order, also determining its path. From this set of potential trajectories, the best, according to a given objective function, are selected by means of an assignment problem, solved as an integer linear problem. For our purpose, VGA is altered to overcome the assumption of a fixed set of vehicles. When calculating all potential trajectories. It is assumed that each request has its own hypothetical vehicle available at its starting location. If a vehicle is not assigned to be used during the assignment step, it is omitted. This procedure was proposed in [105], more details can be found there. The tasks are defined as the trajectories the used vehicles take, their first request defines the task's start and the last request the task's end.

4.7.5. Details on the Manhattan Dataset

Figure 4.13 shows four histograms, all describing the set of trips used for the Manhattan Case study. They illustrate the starting time of the trips (top left), their duration (top right), the resulting ending times (bottom left), and their total size (bottom right).

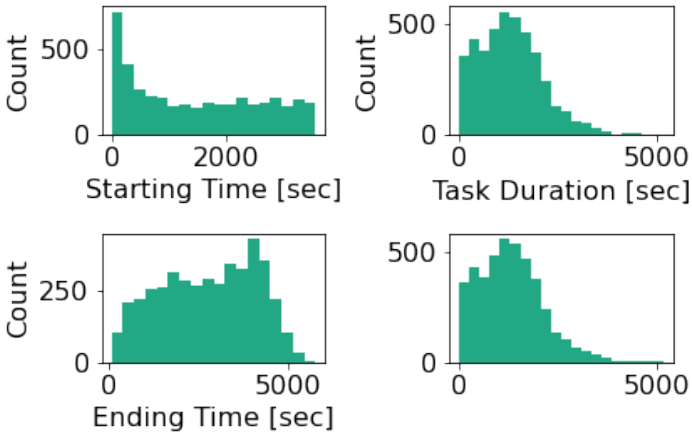


Figure 4.13: This figure shows the distributions of the starting time (top left), their duration (top right), the resulting ending times (bottom left), and their total size (bottom right) of all trips used within the Manhattan case study. Size is measured as the number of individual passengers served by a trip.

5

Fleet Sizing for the Flash Delivery Problem from Multiple Depots a Case Study in Amsterdam

In *previous chapters*, the focus has been on studying either the vehicle routing problem for Flash Deliveries or the fleet sizing problem separately. However, combining these two aspects remains a relevant challenge.
In *this chapter*, we present a novel approach that combines sophisticated routing methods with fleet sizing for Flash Delivery operations. We analyze the effects of this combination and study its application in real-life scenarios. A case study in Amsterdam provides valuable insights and allows us to examine the influence of various parameters on the overall operations.

This chapter is based on:

- M. Kronmueller, A. Fielbaum, J. Alonso-Mora, *"Fleet Sizing for the Flash Delivery Problem from Multiple Depots a Case Study in Amsterdam"*, planned to be submitted to IEEE 27th International Conference on Intelligent Transportation Systems (ITSC) [106]

Abstract

In this chapter, we present a novel approach for fleet sizing in the context of Flash Delivery, a time-sensitive delivery service that requires the fulfillment of customer requests in minutes. Our approach effectively combines individual delivery requests into groups and generates optimized operational plans that can be executed by a single vehicle or autonomous robot. The groups are formed using a modified routing approach for the Flash Delivery problem. Combining the groups into operational plans is done by solving an ILP. To evaluate the effectiveness of our approach, we compare it against three alternative methods: fixed vehicle routing, non-pooled deliveries, and a strategy encouraging the pooling of requests. The results demonstrate the value of our proposed approach, showcasing its ability to optimize the fleet and improve operational efficiency. Our experimental analysis is based on a real-world dataset provided by a Dutch retailer, allowing us to gain valuable insights into the design of Flash Delivery operations and to analyze the effect of the maximum allowed delay, the number of stores to pick up goods from, and the employed cost functions.

5.1. Introduction

In the ever-evolving landscape of retail and logistics, the prominence of Flash Deliveries as a powerful business model is evident through the success of young companies like Flink, Getir, and Gorillas. The growing demand for instant gratification and swift order fulfillment has been the driving force behind the surge in popularity of this time-sensitive delivery approach. Flash deliveries provide customers with the convenience of receiving their requests promptly, challenging traditional retailers to adapt and secure their market share in this highly competitive arena. Collaborations between established players in the industry further exemplify the industry's response to this trend. For instance, in the Netherlands, Albert Heijn partnered with Thuisbezorgd and Deliveroo to provide faster grocery delivery [8]. Similarly, Cornershop merged with Uber, pursuing similar objectives [9].

These processes are accelerated and challenged further by the rapid progress in autonomous delivery robots and autonomous driving technologies. A notable example is Starship Technologies, which has successfully completed millions of autonomous deliveries using their robot solution [3]. This advancement opens up possibilities for operating large fleets at reasonable costs.

In contrast, a potential upside that traditional retailers have is the ban on opening new dark stores, as seen in cities like Amsterdam [10]. Dark stores serve as dedicated pick-up locations, and the prohibition on their establishment presents an opportunity for brick-and-mortar stores to step in and utilize their existing infrastructure as depots. This allows traditional retailers to leverage their physical presence to support Flash Delivery operations.

In planning for Flash Delivery operations, two critical factors come into play: the efficient routing of vehicles or robots and the design of the fleet. The interplay between these factors adds complexity to the overall system. Traditional routing assumes a fixed number of vehicles as input and focuses on optimizing their usage.

Fleet sizing involves determining the optimal number of vehicles required based on how they are utilized during service. To enable fleet sizing with sophisticated routing, we propose a novel method that combines routing optimization and fleet design while considering multiple stores. Taking into account the unique characteristics of multiple stores is particularly important, as it closely resembles the operational setup of traditional retailers. In our study, we utilize real-world data from a Dutch retailer, including information on the locations and number of brick-and-mortar stores and real-life demand patterns. By integrating these aspects, we aim to gain valuable insights into the optimization of Flash Delivery operations.

The contributions of this chapter are twofold: First, we propose a novel combination of methods enabling fleet sizing, including vehicle routing for Flash Delivery operations from multiple stores. Second, we analyze a real-life scenario of deliveries, emulating the entrance of traditional retailers into the Flash Delivery market, shading light onto these operations.

5.2. Related Work

This chapter deals with fleet sizing for the FDP, including the pooling of requests, i.e. vehicles can simultaneously carry multiple requests with similar destinations. To the best of our knowledge, this chapter is the first to do so. As such, this related literature section focuses on fleet sizing in general and on routing for the FDP individually. The related literature predominantly originates from the area of transporting people, such as the dial-a-ride problem and ridesharing. However, there are three key distinctions between these areas and Flash Delivery logistics with autonomous vehicles or robots. First, in logistics, the pick-up location of a request is ambiguous and needs to be decided on. Second, while minimizing delay is crucial in people transportation, Flash Delivery prioritizes operational efficiency and resource utilization over delay reduction. Finally, the usage of autonomous vehicles or robots enables continuous operation.

5.2.1. Fleet Sizing

Fleet Sizing generally answers the question, “How many vehicles are required to serve some demand?”. [107] shows that various effects drive these decisions. The existing literature offers two primary categories of approaches: simulation-based and chaining-based methods. Simulation-based approaches aim to identify optimal fleet designs and sizes by simulating operations with different fleet compositions. For example, in [89], an agent-based micro-simulation model was employed to analyze shared ride services in Austin, Texas. Through cost estimates and simulations with varying fleet sizes, an optimal fleet size was determined using the Golden Section Search method [108]. Chaining-based approaches, on the other hand, involve sequencing requests into chains by reallocating vehicles from completed tasks to subsequent ones. The concept of chaining was initially introduced by [12] to address the minimum fleet problem for taxi rides in Manhattan. They utilized a shareability graph and applied a maximum matching algorithm to find the minimum fleet.

Building upon chaining, several papers extended the approach to ridesharing appli-

cations, enabling vehicles to serve multiple requests. For instance, in [81], chains were iteratively formed using an ILP solver, progressively extending existing chains by adding new tasks. [98] presented a combined optimization model that integrated pooling and chaining, demonstrating the potential for reduced fleet sizes through pooling. [82] proposed a novel order graph capturing complex inter-order shareability and solved a coverage problem over the graph to determine the required fleet sizes. The following two works share a similar idea to the approach presented here, which involves initially calculating how requests or passengers can be served together and then applying chaining. In [83], a routing approach and demand forecasting were employed to maximize their proposed utility metric called "demand utility" on shared trips, with chaining based on [12]. Additionally, [101] utilized temporal and spatial aggregation to form trips, formulating fleet sizing as an ILP and solving it as a minimum flow problem.

5.2.2. Routing for the Flash Delivery Problem

The routing aspect of the FDP represents a specialized domain within dynamic vehicle routing problems. While the FDP shares similarities with the Same-Day Delivery Problem, it poses unique challenges by requiring requests to be fulfilled within minutes after being placed rather than by the end of the day. In the existing literature, only a few works have focused specifically on routing for the FDP, namely [19, 20]. These studies adopt a rolling horizon approach to address the dynamic nature of the problem by dividing it into multiple snapshot problems. Their methodology involves a two-step process for each snapshot. Firstly, a comprehensive set of potential plans for each vehicle is generated. Subsequently, an assignment problem is solved to determine which plans are executed by which vehicles. These works build the foundation for the routing approach applied in this chapter.[1]
Not focusing on Flash Delivery, but the instant delivery problem are the works of [37] and [38]. In [37], a column generation approach is used to optimize the assignment of orders to a heterogeneous fleet of vehicles, considering deadlines of up to hours. In [38], the instant delivery problem with shorter deadlines of 45 minutes is addressed by decomposing it into a series of static problems. Orders are inserted into existing trajectories based on a similarity measure.

5.3. Problem Formulation

Intuitively described, the fleet sizing problem poses a problem in which the number of vehicles and their operational plans need to be found to fulfill a given demand. It becomes the fleet sizing problem for the FDP when all requests need to be delivered within the constraints posed by the Flash Delivery operation. Solutions are optimized based on a given objective.
The inputs are the demand, as a set of requests $\mathcal{R}$ which need to be serviced, the capacity of the vehicles, and a graph $G = (V, E)$ representing the operation environment. The operational environment is represented as a weighted directed graph denoted as $G = (V, E)$, with vertices V representing different locations $l \in V$

[1] Another variation of this approach, generalizing to heterogeneous vehicles, was proposed in [25].

and edges E indicating connections between them. The weight of each edge, denoted as $w(e)$, represents the traversal time. The stores $\mathcal{S}$ form a subset of vertices V, where vehicles can pick up goods to fulfill customer requests. All stores have a full stock of goods at all times.

The demand set $\mathcal{R}$ consists of individual customer requests $r = (l_r^{goal}, t_r)$, where $l_r^{goal} \in V$ represents the goal location and t_r is the request placement time. It is important to note that no specific pickup location for each request is specified, as well as no specific set of products, as we assume each request to be unique. In the FDP, each request must be dropped off within a maximum delay ρ_r^{max}, as [19, 20]. The drop-off delay ρ_r is the difference between the actual drop-off time and the drop-off time if the request was served immediately via the shortest path from the nearest store. Additionally, we consider fixed times t^{load} to load a request to a vehicle and $t^{deliver}$ to deliver it to the customer. The assumed capabilities for vehicles are as follows: Each vehicle has a maximum capacity of κ and drives along the graph, specifying the needed traveling times.

The objective of the fleet sizing problem is to determine the number of vehicles required and their corresponding operational plans ω. An operational plan consists of an ordered set of locations $l \in V$, where each location is assigned one of the following activities: picking up a request, delivering a request to a customer, or waiting for further instructions. The vehicle follows the shortest path between locations. Consequently, following an operational plan results in the delivery of a set of requests denoted as o_ω. Accordingly, the total driving time of a single trajectory ω is $\phi(\omega)$, and the resulting total delay if following this plan is $\rho(\omega)$. The starting time of an operational plan is t_ω^{start}, and t_ω^{end} is the ending time, respectively, starting and ending location are l_ω^{start} and l_ω^{end}.

To execute one operational plan ω, one vehicle is needed. As such, a set of operational plans Ω can be a solution to the fleet sizing problem if it satisfies certain conditions. First, to qualify as a solution, together, all operational plans $\omega \in \Omega$ must successfully deliver all requests. Thereby, each request must be picked up from a store and delivered to the customer before its specified maximum drop-off time. Second, the capacity of each vehicle must not exceed the maximum capacity constraint.

The evaluation of a solution Ω is based on the cost function $J(\Omega)$. The cost function incorporates various factors, including the number of vehicles used (representing fixed capital costs), travel time (representing variable capital costs), and delay costs (representing the quality of service experienced by customers). For each vehicle or executed trajectory in Ω, a fixed capital cost of M_{fix} is incurred. Additionally, the costs of travel time and delay are weighted convexly using a cost weight parameter $\alpha \in [0, 1]$. This results in the cost function as follows:

$$J(\Omega) = M_{fix} \cdot |\Omega| + \sum_{\omega \in \Omega} [(1 - \alpha) \cdot \rho(\omega) + \alpha \cdot \phi(\omega)] \tag{5.1}$$

Let $\mho$ be the set that includes all feasible sets of operational plans Ω, representing solutions to the FDP. Given the set $\mho$, the fleet sizing problem can be formulated

as follows:

$$\min_{\Omega \in \mho} J(\Omega) \tag{5.2}$$

Note that constraints are implicit in the set $\mho$.

5.4. Method

To determine the solution Ω_{sol}, our proposed approach consists of two key steps: pooling and chaining. Pooling involves the coordinated gathering of multiple requests into groups that can be efficiently delivered together. This step includes optimizing the routing for each group, resulting in small operational plans. For clarity, these small operational plans are not yet the final operational plans spanning the entire operation but rather smaller components. Thus, we refer to these small operational plans as tasks T. The notation for tasks is identical to operational plans. For example, a task's starting and ending times are t_T^{start} and t_T^{end}, and in the same manner, the starting and ending locations are l_T^{start} and l_T^{end}. Intuitively, tasks represent individual units of work that can be performed efficiently by one vehicle. Chaining combines these small operational plans or tasks T to create final operational plans ω that cover the entire operation. This is done by assembling the tasks in a sequential manner, considering dependencies, and optimizing the overall delivery process. An overview of this approach is provided in Figure 5.1. For a more in-depth understanding of pooling and chaining, please refer to Sections 5.4.1 and 5.4.2.

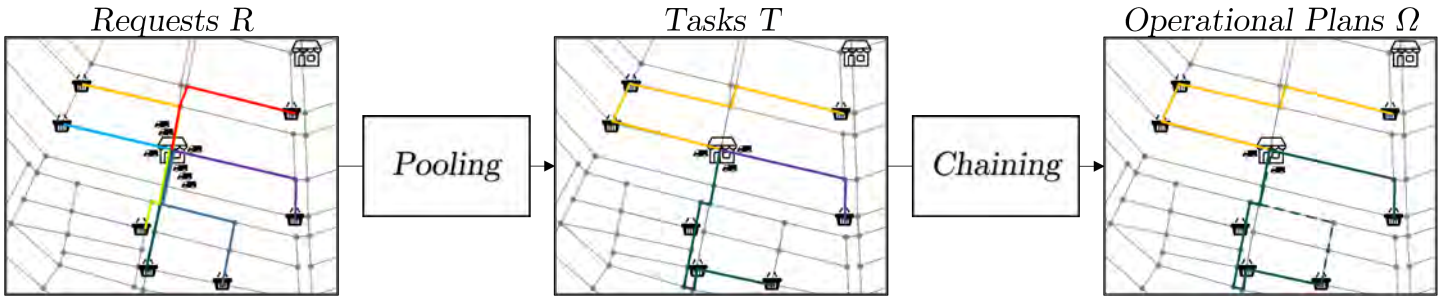


Figure 5.1: Method Overview: Our method takes a set of requests as input. The first step, pooling, involves grouping the requests into groups that can be efficiently delivered together. These groups are referred to as tasks, which include the corresponding routing optimization. The second step, chaining, focuses on sequencing the tasks to create operational plans, with each plan requiring a single vehicle. To provide visual clarity, different colors are used to represent each vehicle at each step of the process. Dashed lines represent vehicles driving in between tasks (chaining).

5.4.1. Pooling

The pooling step of our method is based on a dynamic routing approach for on-demand last-mile logistics from multiple stores [19, 11]. However, we adapt this approach to eliminate the requirement of a fixed fleet of vehicles as input, following the methodology proposed in [105]. By building upon the principles of [19], we can ensure that our method generates high-quality routes that satisfy the constraints of the FDP. To address the dynamic nature of the problem, we employ a rolling horizon approach by dividing the entire operation into multiple snapshot problems. For each snapshot problem, we first calculate a large set of potential routes for the vehicles. We then select the routes to be executed from this set. Routes represent vehicle-specific operational plans, considering the vehicle's current state. These routes are designed to be feasible, adhering to the vehicle's capacity constraints and ensuring the timely completion of all assigned requests. Routes overlap multiple snapshot problems and are subject to change. For algorithmic details on the approach to efficiently calculate the route set, we refer to [19].

Each route is assigned a cost to execute it, following Equation 5.1. The set of vehicles to calculate routes for is not fixed in this chapter but differs for each snapshot problem. In each step, we consider the none idle vehicles of the previous time step and introduce new potential vehicles. We assume that one potential vehicle is available for each request at the closest store to its goal location starting from the request's placement time t_r.

The selection of routes to execute is performed through a coordinated process using an assignment problem, which is formulated and solved as an ILP. The ILP is a standard formulation to assign routes to vehicles such that all users are served[2] and no vehicle is assigned to more than one route; its explicit formulation can be found in [19, 11]. If a new potential vehicle is chosen by the assignment, it is instantiated into the problem and follows the assigned route. Any potential vehicles that are not assigned are disregarded.

Each vehicle follows its assigned route until the next snapshot, at which point the routes of all vehicles are updated and thus can be prolonged. Once a vehicle completes its assigned route and becomes idle, it is removed from the problem. The full route that each vehicle executes, from its creation until it is dropped, constitutes a task T. All tasks T are summarized in the set $\mathcal{T}$.

5.4.2. Chaining

The chaining step is employed to combine the tasks T generated by the pooling step into operational plans ω spanning the entire operation. The objective of chaining is to optimally sequence tasks in a way that allows them to be executed by a single vehicle.

In order for two tasks T_i and T_j to be executed consecutively by a single vehicle, the vehicle must be able to relocate from the end location of task T_i, denoted as $l^{end}_{T_i}$, to the start location of the subsequent task T_j, denoted as $l^{end}_{T_j}$, and reach it before its designated starting time $t^{start}_{T_j}$. The travel time required to drive from

[2]Serving all individual requests is always possible due to the possibility of creating new vehicles.

$l_{T_i}^{end}$ to $l_{T_j}^{start}$ is represented as $\tau_{i,j}$.[3] Thus, two tasks T_i and T_j can be chained if the following equation is satisfied: $t_i^{end} + \tau_{i,j} \leq t_j^{start}$. All pairs of tasks (i, j) that fulfill this equation are summarized in the set $\mathcal{X}$. To coordinate which pairs of tasks from the set $\mathcal{X}$ are actually executed in sequence by one vehicle (chained), an ILP can be formulated and solved [12]. The ILP minimizes the overall costs, Equation 5.3.[4]

$$\min \sum_{i,j \in \mathcal{X}} x_{i,j} \cdot \Big[- M_{fix} + \alpha \cdot \tau_{i,j} \Big] \tag{5.3}$$

Being subject to each task having maximally one preceding and one subsequent task. Successfully chained tasks form a single operational plan ω within the solution Ω_{sol}. The size of the solution $|\Omega_{sol}|$ defines the number of required vehicles, as each plan requires one.

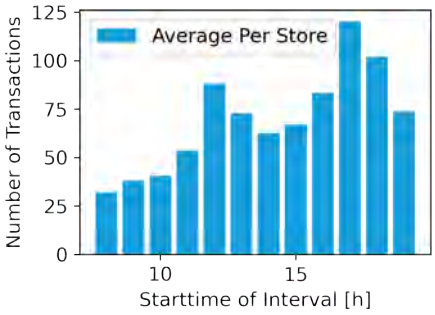
5.5. Dataset

The data set this case study is based on describes the shopping behavior of walk-in customers in regular brick-and-mortar supermarkets in Amsterdam, Netherlands. The locations of 42 stores belonging to a single retail company in the city center are known and considered as pick-up locations $\mathcal{S}$. Figure 5.2a displays a map of Amsterdam's city center, highlighting the locations of all stores in the dataset. The dataset provides information on the number of transactions per hour for each store, although the exact transaction times are not available. This transaction data is available from 8 a.m. to 8 p.m. Figure 5.2b shows the used demand pattern as the average number of transactions for all stores against time. Most notably, clear peaks in demand during noon and the evening are present.

In this study, we simulate a Flash Delivery operation by modifying the original data. One crucial aspect that undergoes changes is the set of requests $\mathcal{R}$. We presume that people reside in close proximity to the stores they frequent. Specifically, we assume that each person exclusively shops at their nearest store, thereby defining an area A_s associated with each store s. This area comprises all vertices $l \in V$ for which store s is the closest one. We iterate through all stores and time windows to construct the individual requests r within the demand set $\mathcal{R}$. For a given store s and a specific time window k (one hour), the provided data includes the number of transactions conducted at that store. We assume that a constant percentage[5] of these transactions will shift from traditional brick-and-mortar stores to the Flash Delivery service. For each individual request r, we sample the goal location l_r^{goal} from the set of vertices within the corresponding area A_s. We assume that customers are uniformly distributed within this area. The request time t_r is also uniformly sampled from the corresponding time window k (one hour).

[3] During relocation, the vehicle is empty.
[4] This is equivalent to the overall cost function in Equation 5.1, as the cost to execute tasks can be excluded as it is constant.
[5] Due to confidentiality reasons, we can not report exact numbers here.

(a) All store locations over a map of Amsterdam.



(b) Average number of transactions per store.

Figure 5.2: Store distribution and demand data of the used data.

5

5.6. Experiments and Results

5.6.1. Experimental Setup

Our experiments focus on Amsterdam's city center, represented by a graph of 2717 vertices and 5632 edges. However, we reduce the set of stores $\mathcal{S}$. This decision is based on the retailer's reasoning that some stores are too busy to be suitable as pick-up locations. As such, we exclude the busiest half, measured in total number of transactions, of all stores from being pick-up locations. Below, we also analyze a scenario using all stores. The demand, as described above, stays identical, as it is not affected by such strategic decisions. We assume a loading and service time of 1 minute for each request ($t^{load} = t^{deliver} = 60[sec]$). The maximum delay allowed during pooling is set to 5 minutes ($\rho_r^{max} = 300[sec]$). A snapshot problem is solved every 100 seconds. Following the logic that delays in on-demand delivery operations are nearly neglectable as long as the request is delivered within the promised time window, we set the cost weight in all functions to $\alpha = 1$, fully focusing on total driving time. In the cost functions, we use a large value of $M_{fix} = 2000[sec]$, making the minimization of fleet size the first priority[6].

Due to the nature of the data used to generate the demand, it exhibits an inherent structure divided into one-hour intervals, clearly seen in Figure 5.2b. First, we apply the proposed method to each interval separately. Second, we repeat the chaining step, chaining the obtained operational plans per interval.

5.6.2. Results

First, "How many vehicles are required?". For the entire day, a total of 459 vehicles are needed. Figure 5.3 illustrates the status of each vehicle throughout the day. The number of working vehicles (green) increases, i.e. the number of vehicles yet to

[6] M_{fix} larger than the maximal time for relocating, determined by the environment, is sufficient to achieve this effect.

start (purple) decreases, reaching the highest fleet utilization during the hour with the highest demand. The steps in the graph are due to the hourly segmentation of the demand.

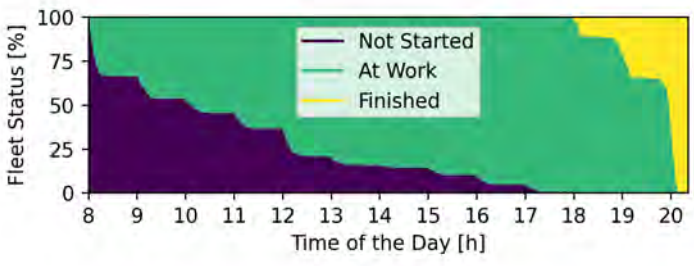


Figure 5.3: The status of each vehicle throughout the day.

To further understand the results we analyze each hour in more detail. The number of requests, the number of tasks (the result of pooling), and the required fleet size per hour are shown in Figure 5.4. As the number of initial requests increases in one interval, more tasks are generated, leading to higher fleet sizes. The difference between the number of requests and tasks becomes more significant with higher demand, indicating that it is easier to group and serve requests together when there is a larger volume of demand.

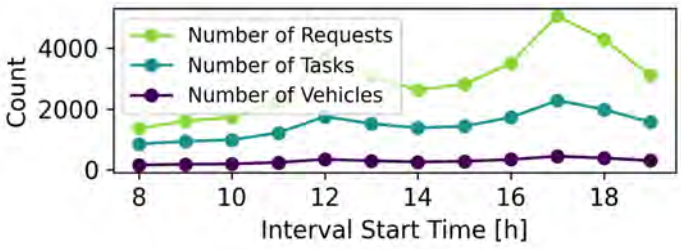


Figure 5.4: The number of requests, the number of tasks (result of pooling), and the required fleet size for each interval for the entire day are shown.

During the peak hour (17:00-18:00), 5054 requests were grouped into 2278 tasks. Each task serves an average of 2.22 requests and takes 445.7 seconds. To handle these tasks, a fleet of 443 vehicles is required, which is slightly less than for the entire day. Each vehicle serves 11.40 requests on average, and an operational plan takes approximately 55 minutes and 33 seconds. This duration is close to spanning the full hour, indicating effective utilization of the vehicles.
Total traffic by all vehicles shows the same correlation with demand (Figure 5.5). It is observed that traffic originating from the pooling step (pooling traffic) exceeds traffic originating from the chaining step (chaining traffic). This difference becomes

more pronounced with higher demand. Generally, with higher demand, the average chaining traffic per vehicle decreases.

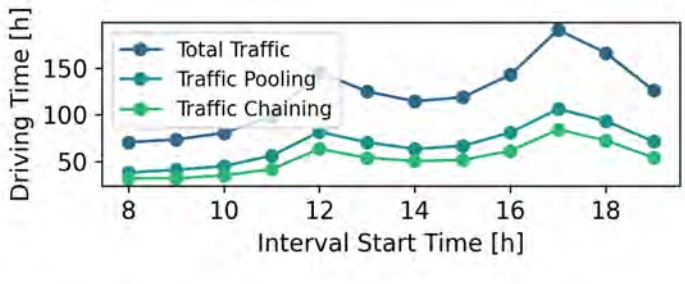
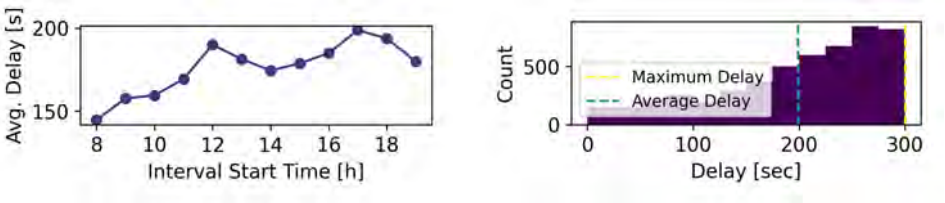


Figure 5.5: The total traffic and its breakdown into polling traffic and chaining traffic for each interval throughout the day is shown.

5

Figure 5.6a presents the average delay per request over the course of the entire day. It differs by about 1 minute between 140 and 200 seconds over the day. About 50 seconds, half of the time step of the pooling algorithm is due to the applied rolling horizon approach. Additionally, Figure 5.6b displays the delay distribution for all requests between 17:00 and 18:00. Each request experiences an average delay of 199 seconds. There is a noticeable increase in the number of requests with higher delays approaching the maximum allowed delay of 300 seconds.[7]



(a) Average delay of intervals.

(b) Distribution peak interval.

Figure 5.6: Two figures showing the average delay of all intervals (a) and the delay distribution of the peak interval (b).

To conduct a comparative analysis and examine the key parameters, the focus of the study is narrowed down to the peak hour interval, from 17:00 to 18:00. This time period is chosen due to its significance, as it represents the hour with the highest number of transactions throughout the entire day.
To range in the performance of the proposed approach, we compare it against three opposing approaches. First, 'encouraged pooling", we apply a strategy encouraging pooling to decrease the number of tasks obtained. To do so, we add costs to a route

[7]Recall that delay was not considered as part of the cost function. We do so as part of the sensitivity analysis below.

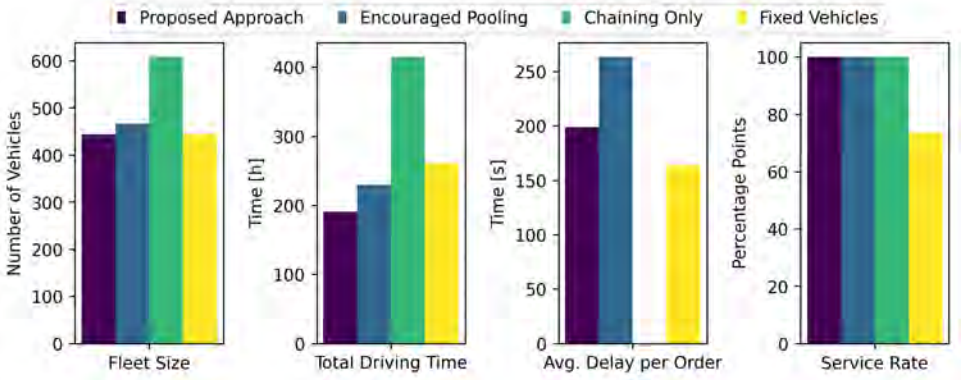


Figure 5.7: Comparison of the proposed approach to three different strategies based on the main KPIs.

if it uses a new potential vehicle. This extra cost was set to equal 1000 seconds. Second, "chaining only", we exclude the pooling step and deliver each request individually. Third, "fixed vehicles", we use a fixed number of vehicles, equivalent to the results of the proposed approach, and route them as [19] (pooling step). The comparative results are presented in Figure 5.7. In the "encouraged pooling" and "chaining only" approaches, the fleet sizes increase, and higher total driving times compared to the proposed approach are needed. The "chaining only" approach has no delay since each request is immediately served with its own vehicle. Service rates are at the enforced 100% for all three methods (Proposed approach, "encouraged pooling" and "chaining only"). In contrast, using a "fixed number of vehicles" does not enforce the service rate but serves as many requests as possible using the available vehicles. We fixed the fleet size to 443, the same number as for the proposed approach. As a result, around 62.5% of requests are served, requiring more driving time and a lower average delay. The main reason for this difference is that the vehicles are not rebalanced as effectively as with the chaining step, which is done in hindsight with full information over the full planning horizon.

Last, we study the effect of the maximum allowed delay ρ_r^{max}, the number of stores to pick up goods, and the cost weight between delay and driving time. We vary the studied variable exclusively and compare fleet size, traffic, and delay.

Delay: We vary the maximal delay as $\rho^{max} = [4, 6, 8]$. Results are shown in Figure 5.8. The higher the maximum allowed delay, the lower are required fleet sizes, accompanied by lower traffic, but at the price of higher values of average delay. This is somehow expected. Most interestingly, are the changes in the split between pooling traffic and chaining traffic. Both decrease with higher maximum delay, but the amount of change in chaining is more, as more requests get served together, which then befits the fleet size.

Number of Stores: For the experiments, up to here, the busier half of all stores have not been considered as pick-up locations. Here, we compare the influence of the number of used stores on the obtained results. We exclude 10 more and 10 fewer

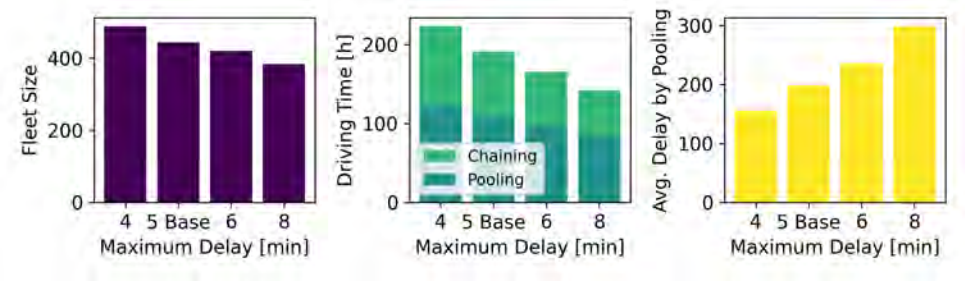


Figure 5.8: Comparison of fleet size, traffic, and average delay for different values of allowed maximum delay.

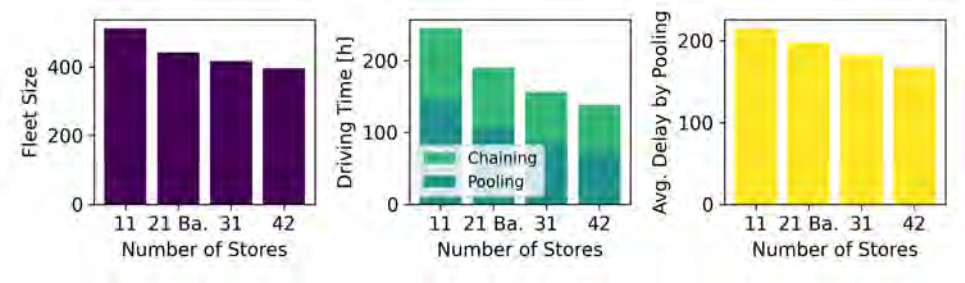


Figure 5.9: Comparison of fleet size, traffic and average delay for different number of available stores.

stores, as well as using all stores of the retailer. Results are visualized in Figure 5.9. All KPIs improve the more stores are used. For fleet size and traffic, the changes get smaller the more stores are used. So the gains of using one additional store when having 11 stores are larger than when already using 41 stores. Changes in delay are more constant.

Cost Weight for Pooling α: The relation between total driving time and delay experienced by customers is captured in the used cost functions. For all experiments so far, we did not consider delay as a cost, here, we do so by varying alpha in $\alpha = [0.9, 0.95]$. The obtained results are illustrated in Figure 5.10. As a direct result average delay decreases, the lower α the more. This comes at the cost of an increased fleet size. Changes in traffic are minor.

5.7. Conclusion

We presented a novel approach for fleet sizing for the FDP. The comparison with alternative strategies demonstrates the benefits of our approach, showing that the integration of both pooling and chaining steps leads to improved performance compared to using only one of these strategies. Furthermore, by utilizing a real-world dataset, we were able to gain valuable insights into the operation of Flash Delivery services. We explored the effects of store selection, maximum delay, and cost weight-

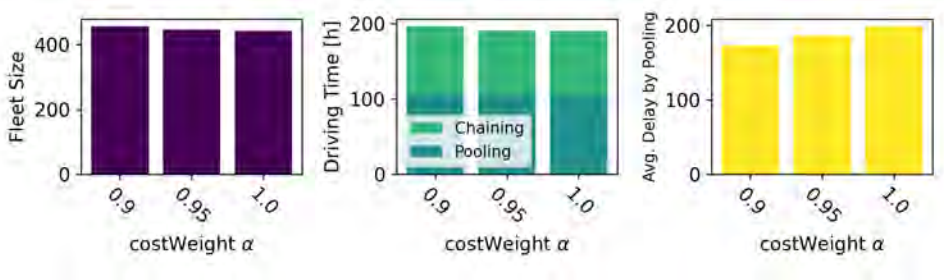


Figure 5.10: Comparison of fleet size, traffic, and average delay for different values of the cost weight α.

ing on fleet size, traffic, and delay. These findings provide practical knowledge for designing and managing Flash Delivery systems in urban environments. For future work, it is essential to reduce assumptions and incorporate real-life features such as traffic conditions.

6

Conclusion

6.1. Conclusion

This thesis is centered around the operational planning of Flash Delivery services, with a primary focus on addressing the VRP and the FSP. Dedicated algorithms have been proposed to tackle these specific problems, and their individual effectiveness has been demonstrated. Furthermore, the study extends its scope by exploring the combined application of these algorithms through a real-world case study using actual data, provided by a Dutch retailer. The comparative analyses conducted within this thesis validate the efficiency of the proposed approaches. Further, the impact of key parameters was studied to offer valuable insights into the overall performance of Flash Delivery operations. As a result, this work lays a solid foundation for the optimization of operational processes in the realm of Flash Deliveries.

In Chapter 2, the Flash Delivery problem was formally defined as a dynamic vehicle routing problem, which is modeled as a MDP. The proposed method for solving the problem leverages the idea of first calculating a large set of plans for all vehicles and, secondly, choosing which plans to execute by means of an optimization problem. This method efficiently considers multiple depots for each order, autonomously selecting the most suitable one. Vehicles can visit depots to load additional orders, enhancing operational efficiency. The method's scalability is demonstrated in scenarios with numerous orders, vehicles, and depots. Extensive computational experiments revealed a 20% improvement over a greedy approach, showcasing the effectiveness of the proposed routing method. Simulations further confirmed the value of considering multiple depots and performing pre-empty depot returns. It was shown that the proposed routing method is suitable to tackle the routing problem of the FDP. It builds the foundation of this dissertation and was reused and extended in other parts of this dissertation.

Chapter 3 introduced an optimization-based approach to route a heterogeneous fleet of vehicles for Flash Deliveries. It extended the in Chapter 2 proposed routing approach to heterogeneous modes of transportation, including heterogeneous vehicles, such as small delivery robots, cars, bikes, and drones. Through experiments, the proposed methods demonstrated their merits, showcasing higher delivery rates with reduced driving distances. Moreover, the study delved into the size and composition of different fleets, revealing that a larger number of drones can enhance service rates at the expense of increased travel distances.

Chapter 4 tackled the FSP. The traditional FSP is extended by introducing additional flexibility to delay individual transportation tasks. A new problem is introduced and formulated, FSD. FSD is proven to be NP-hard. A novel method formulating the FSD as a MILP is proposed. Results demonstrated that incorporating task delays had two significant effects: reductions in fleet sizes and an expanded trade-off space between the number of vehicles, execution time, and added delay. A comparison with traditional fleet sizing approaches revealed overall lower costs achieved through the introduction of delays. The real-life case study of taxi rides in Manhattan showcased the practical benefits of task delays. With a maximum of just

3 minutes of additional delay per ride, the required fleet size decreased by about 9% in comparison to the minimal fleet without delays, while the average delay per task was less than 23 seconds. In conclusion, leveraging delays improves solutions and allows for more flexible trade-offs, resulting in potential reductions in the number of vehicles required for Flash Delivery operations.

Chapter 5 introduced a novel approach combining fleet sizing and vehicle routing for Flash Delivery operations from multiple stores. A real-life scenario emulating the entrance of traditional retailers into the Flash Delivery market was analyzed. The comparison with alternative strategies demonstrates the benefits of our approach, showing that the integration of both pooling and chaining steps leads to improved performance compared to using only one of these strategies. Our experimental analysis is based on a real-world dataset provided by a Dutch retailer. By exploring the effects of store selection, maximum delay, and cost weighting on fleet size, traffic, and delay, this chapter offers practical knowledge for effectively managing Flash Delivery systems. These findings provide practical knowledge for designing and managing Flash Delivery systems in urban environments.

Overall, these contributions address the key challenges in operating a Flash Delivery service. However, there are still several remaining issues towards improving obtained results or before the proposed methods are applicable to everyday practices. Towards that goal, possible directions for future work are described below.

6.1.1. Answering the Posed Main-Research Question

This thesis attempts to answer the following main-research question: "How can the planning for Flash Delivery operations regarding vehicle routing and fleet sizing be accomplished efficiently and effectively?"

Vehicle routing and fleet sizing are inherently complex, demanding potent algorithms to compute efficient and effective solutions. Throughout this thesis, the goal has been to develop novel algorithms that introduce new aspects and opportunities to address these challenges while remaining solvable within reasonable time and resource constraints.
In essence, the effectiveness of the solutions derived can be enhanced when algorithms can encompass a broader range of real-world problem facets, in conjunction with plausible assumptions. For example, the inclusion of pre-empty depot returns or factoring in delays during fleet sizing enhances the operational efficiency of the planned activities. Similarly, the employment of a variety of vehicle types and their cooperative potential in task fulfillment adds another layer of complexity and realism.
Moreover, if problems along the value chain can be collectively solved, it presents advantages. This not only streamlines the planning process but also allows for the elimination of certain assumptions. In the context of this thesis, the integration of vehicle routing and fleet sizing showcases such a situation, eliminating the need to assume fixed routes or a simple routing rule, allowing more effective planning.

In summary, developing more comprehensive algorithms leads to more efficient and effective Flash Delivery operations.

6.2. Future Research

Although this thesis has provided a step forward toward efficient Flash Deliveries, many challenges and avenues for future research remain. In the following, we recommend several research avenues for Flash Deliveries regarding VRP and the FSP.

6.2.1. Vehicle Routing for Flash Deliveries

- **Real-World Representation:** Generally speaking, the better real-world scenarios are represented within the analyzed problems, the better. Taking assumptions is common and necessary in research, as otherwise, the problems become impossible to solve in a reasonable time and would take away the focus of what research is supposed to contribute. Nevertheless, decreasing the number and strength of assumptions poses opportunities for future research. To only name some of the many, we want to highlight the common assumption of static travel times, the neglection of traffic and congestion in cities, the neglection of parking times, and the absence of unforeseen disruptions of the operation. Specific to the research presented in this dissertation, considering longer loading and delivery times for more complex orders and incorporating lead times for pick-ups from stores would lead to more accurate and practical solutions. Addressing these aspects can enhance the applicability and effectiveness of Flash Delivery operations in real-world settings.

- **Anticipatory Methods:** Future research could focus on extending the proposed method to incorporate anticipation, aiming to reduce the risk of unfavorable states in the dynamic Flash Delivery operation. Anticipating future scenarios can be challenging, especially in large-scale problems with rapidly changing dynamics. Generally, larger problems can tolerate less extensive ways to anticipate or simulate future scenarios. Developing efficient algorithms that can anticipate future orders and dynamically adjust routing decisions for large-scale Flash Delivery services is an important direction for further investigation. Additionally, incorporating stochastic information about potential orders and integrating orders known ahead of time into the optimization process could lead to improved operational efficiency.

- **Integration with Other Logistic Processes:** Investigating the integration of Flash Deliveries with other logistical processes presents intriguing avenues for future research, both vertically and horizontally. Vertical fusion involves exploring how Flash Deliveries can be combined with preceding steps in the value chain, leading to more seamless and efficient overall operations. Similarly, horizontal fusion entails integrating Flash Deliveries with other delivery processes, such as return processes or next-day delivery services, to enhance

the overall efficiency and effectiveness of both operations. Studying these combinations could unlock new possibilities and efficiencies in last-mile logistics.

- **Number and Location of Depots:** In the future, two intriguing questions are how many depots should be operated within a given area, as well as where and how many depots should be considered per order. Integrating routing considerations into these questions appears to be a promising avenue that can lead to further improvements in Flash Delivery operations.
- **Less Urgency:** The routing algorithms presented in this dissertation are specifically tailored for Flash Delivery operations. However, an interesting area for future research lies in their application and adaptation to other logistical operations with varying levels of urgency, such as deliveries within an hour or within a few hours. Exploring how these algorithms can be extended to suit different time frames could yield valuable insights for diverse delivery services. How the approaches combine and compare to other methods for the DVRP, which can emerge in case of less urgency, is a critical aspect in this avenue of future work.
- **Details for Heterogeneous Vehicles:** Chapter 3 introduced the extension to heterogeneous vehicles, which relies on accurately representing the capabilities of different vehicle types. This enables a thorough analysis of their strengths and weaknesses and facilitates informed decision-making. As such, representing them as well as possible is an opportunity for future improvements. For instance, considering the battery life of drones is crucial to avoid over-optimistic use in the obtained results. Additionally, novel ways of combining heterogeneous vehicles in different integration strategies and the corresponding VRPs pose interesting future research possibilities.
- **User Experience and Customer Preferences:** Investigating methods to incorporate customer preferences and feedback into the operational planning process could enhance service quality and customer satisfaction.

6.2.2. Fleet Sizing

- **Including Dynamics of Operation:** Addressing the disparity between fleet sizing approaches, which usually assume complete information, and the dynamic real-time execution presents a crucial challenge. Otherwise, obtained results, like an optimal fleet size, will not be sufficient to obtain wanted costs and service quality.
- **Real-World Representation:** Indeed, just like for the VRP, accurately representing the FSP and the underlying problem is essential to obtain meaningful results and practical solutions. For example, accounting for uncertainties in the operation would contribute to more robust and adaptable fleet sizing strategies.
- **Delay in Fleet Sizing:** In Chapter 4, the FSD was introduced. Two key directions for future work in this area pose. First, enhancing the scalability of

the FSD approach is crucial to handle large-scale Flash Delivery operations. Developing powerful heuristics tailored to the FSD problem will enable faster or better scaling solutions. Second, exploring the application of delays in other types of transportation problems opens up intriguing possibilities.

- **Fleet Design:** The approaches proposed for the FSP in this dissertation assume the use of homogeneous vehicles. However, expanding the problem context to include the decision on the types of vehicles to be used could lead to more specialized fleets and enhance their applicability in real-life scenarios. Incorporating the decision on the fleet composition from diverse vehicle options increases the problem's complexity strongly.

Bibliography

[1] Gorillas, "Gorillas celebrates 16 million orders worldwide," 2022. [Online]. Available: https://gorillas.io/en/blog/gorillas-celebrates-16-million-orders-worldwide

[2] Kantar, "Markt van flitsbezorging groeit onstuimig in nederland," 2022. [Online]. Available: https://www.kantar.com/nl/kantar-nieuws/onderzoek-markt-flitsbezorging

[3] S. Technologies. (2023) Comapny website of starship technologies. [Online]. Available: https://www.starship.xyz/

[4] N. Christie and H. Ward, "The health and safety risks for people who drive for work in the gig economy," *Journal of Transport and Health*, vol. 13, pp. 115–127, 2019.

[5] Y. Zheng, Y. Ma, L. Guo, J. Cheng, and Y. Zhang, "Crash involvement and risky riding behaviors among delivery riders in china: The role of working conditions," *Transportation Research Record*, vol. 2673, no. 4, pp. 1011–1022, 2019.

[6] A. M. Amiri, M. R. Ferguson, and S. Razavi, "Adoption patterns of autonomous technologies in logistics: evidence for niagara region," *Transportation Letters*, vol. 14, no. 7, pp. 685–696, 2022.

[7] A. Fielbaum, F. Ruiz, G. Boccardo, D. Rubio, A. Tirachini, and J. Rosales-Salas, "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*, vol. 31, pp. 63–77, 2023.

[8] Albert Heijn Nieuws, "Albert heijn breidt samenwerking met deliveroo en thuisbezorgd.nl uit," 2022. [Online]. Available: https://nieuws.ah.nl/albert-heijn-breidt-samenwerking-met-deliveroo-en-thuisbezorgdnl-uit

[9] Cbinsights Research Briefs, "Uber acquires cornershop," 2021. [Online]. Available: https://www.cbinsights.com/research/uber-acquires-cornershop/

[10] M. Peters and H. Ernste, "Discovering a new phenomenon inside a dutch urban context: 'flash delivery'," *Master Thesis at Radboud University*, 2022.

[11] J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus, "On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment," *Proceedings of the National Academy of Sciences*, vol. 114, no. 3, pp. 462–467, 2017.

[12] M. Vazifeh, P. Santi, G. Resta, S. Strogatz, and C. Ratti, "Addressing the minimum fleet problem in on-demand urban mobility," *Nature*, vol. 557, no. 7706, pp. 534–538, 05 2018.

[13] X. Bai, A. Fielbaum, M. Kronmüller, L. Knoedler, and J. Alonso-Mora, "Group-based distributed auction algorithms for multi-robot task assignment," *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 2, pp. 1292–1303, 2023.

[14] J. van Lochem, M. Kronmueller, P. van 't Hof, and J. Alonso-Mora, "Anticipatory vehicle routing for same-day pick-up and delivery using historical data clustering," *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020.

[15] A. Fielbaum, M. Kronmueller, and J. Alonso-Mora, "Anticipatory routing methods for an on-demand ridepooling mobility system," *Transportation*, vol. 49, p. 1921–1962, 2021.

[16] S. Bhatia and M. Kronmueller, "Anticipatory route optimization in on-demand same-day grocery delivery," *Master Thesis at Delft University of Technology*, 2022.

[17] C. Claij and M. Kronmueller, "Fleet design for last-mile on-demand logistics," *Master Thesis at Delft University of Technology*, 2022.

[18] J. Pierotti, M. Kronmueller, J. Alonso-Mora, J. T. van Essen, and W. Böhmer, "Reinforcement learning for the knapsack problem," in *Optimization and Data Science: Trends and Applications*. Springer International Publishing, 2021, pp. 3–13.

[19] M. Kronmueller, A. Fielbaum, and J. Alonso-Mora, "On-demand grocery delivery from multiple local stores with autonomous robots," in *2021 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, 2021, pp. 29–37.

[20] ——, "Online flash delivery from multiple depots," *Transportation Letters*, vol. 0, no. 0, pp. 1–17, 2023.

[21] T. K. Ralphs, L. Kopman, W. R. Pulleyblank, and L. E. T. Jr., "On the capacitated vehicle routing problem," *Math. Program.*, vol. 94, no. 2-3, pp. 343–359, 2003.

[22] M. Bernardo, B. Du, and J. Pannek, "A simulation-based solution approach for the robust capacitated vehicle routing problem with uncertain demands," *Transportation Letters*, vol. 13, no. 9, pp. 664–673, 2021.

[23] J. R. Montoya-Torres, J. López Franco, S. Nieto Isaza, H. Felizzola Jiménez, and N. Herazo-Padilla, "A literature review on the vehicle routing problem with multiple depots," *Computers &Industrial Engineering*, vol. 79, pp. 115–129, 2015.

[24] M. Bernardo, B. Du, and A. B. Matias, "Achieving robustness in the capacitated vehicle routing problem with stochastic demands," *Transportation Letters*, vol. 15, no. 3, pp. 254–268, 2023.

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

[26] S. A. Voccia, A. M. Campbell, and B. W. Thomas, "The same-day delivery problem for online purchases," *Transportation Science*, vol. 53, no. 1, pp. 167–184, 2017.

[27] M. W. Ulmer, B. W. Thomas, and D. C. Mattfeld, "Preemptive depot returns for dynamic same-day delivery," *EURO Journal on Transportation and Logistics*, vol. 8, p. 327–361, 2019.

[28] J.-F. Côté, T. A. de Queiroz, F. Gallesi, and M. Iori, "Dynamic optimization algorithms for same-day delivery problems," in *cirrelt.ca*, 04 2021.

[29] G. Ghiani, E. Manni, A. Quaranta, and C. Triki, "Anticipatory algorithms for same-day courier dispatching," *Transportation Research Part E: Logistics and Transportation Review*, vol. 45, no. 1, pp. 96–106, 2009.

[30] N. Azi, M. Gendreau, and J.-Y. Potvin, "A dynamic vehicle routing problem with multiple delivery routes," *Annals of Operations Research*, vol. 199, no. 1, pp. 103–112, 2012.

[31] M. A. Klapp, A. L. Erera, and A. Toriello, "The one-dimensional dynamic dispatch waves problem," *Transportation Science*, vol. 52, no. 2, pp. 402–415, 2016.

[32] ——, "The dynamic dispatch waves problem for same-day delivery," *European Journal of Operational Research*, vol. 271, no. 2, pp. 519–534, 2018.

[33] ——, "Request acceptance in same-day delivery," *Transportation Research Part E: Logistics and Transportation Review*, vol. 143, p. 102083, 2020.

[34] M. W. Ulmer and S. Streng, "Same-day delivery with pickup stations and autonomous vehicles," *Computers &Operations Research*, vol. 108, pp. 1–19, 2019.

[35] R. Bent and P. Van Hentenryck, "Scenario-based planning for partially dynamic vehicle routing with stochastic customers," *Operations Research*, vol. 52, pp. 977–987, 2004.

[36] C. Ackva and M. Ulmer, "Consistent routing for local same-day delivery via micro-hubs," *Working Paper Series*, 2022.

[37] L. Zhen, J. Wu, G. Laporte, and Z. Tan, "Heterogeneous instant delivery orders scheduling and routing problem," *Computers &Operations Research*, vol. 157, p. 106246, 2023.

[38] G. Xue and Z. Wang, "Order acceptance and scheduling in the instant delivery system," *Computers &Industrial Engineering*, vol. 182, p. 109395, 2023.

[39] D. Reyes, A. Erera, M. Savelsbergh, S. Sahasrabudhe, and R. O'Neil, "The meal delivery routing problem," *Optimization Online*, vol. 6571, 2018.

[40] B. Yildiz and M. Savelsbergh, "Provably high-quality solutions for the meal delivery routing problem," *Transportation Science*, vol. 53, no. 5, pp. 1372–1388, 2019.

[41] M. W. Ulmer, B. W. Thomas, A. M. Campbell, and N. Woyak, "The restaurant meal delivery problem: Dynamic pickup and delivery with deadlines and random ready times," *Transportation Science*, vol. 55, no. 1, pp. 75–100, 2021.

[42] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot task allocation: A review of the state-of-the-art," *Cooperative Robots and Sensor Networks 2015*, pp. 31–51, 2015.

[43] J.-F. Cordeau and G. Laporte, "The dial-a-ride problem: models and algorithms," *Annals of Operations Research*, vol. 153, no. 1, pp. 29–46, 2007.

[44] A. Fielbaum, X. Bai, and J. Alonso-Mora, "On-demand ridesharing with optimized pick-up and drop-off walking locations," *Transportation Research Part C: Emerging Technologies*, vol. 126, p. 103061, 2021.

[45] N. Agatz, A. Erera, M. Savelsbergh, and X. Wang, "Optimization for dynamic ride-sharing: A review," *European Journal of Operational Research*, vol. 223, no. 2, pp. 295–303, 2012.

[46] A. Mourad, J. Puchinger, and C. Chu, "A survey of models and algorithms for optimizing shared mobility," *Transportation Research Part B: Methodological*, vol. 123, pp. 323–346, 2019.

[47] S. Narayanan, E. Chaniotakis, and C. Antoniou, "Shared autonomous vehicle services: A comprehensive review," *Transportation Research Part C: Emerging Technologies*, vol. 111, pp. 255–293, 2020.

[48] B. Yu, N. Ma, W. Cai, T. Li, X. Yuan, and B. Yao, "Improved ant colony optimisation for the dynamic multi-depot vehicle routing problem," *International Journal of Logistics Research and Applications*, vol. 16, no. 2, pp. 144–157, 2013.

[49] H. Xu, P. Pu, and F. Duan, "A hybrid ant colony optimization for dynamic multidepot vehicle routing problem," *Discrete Dynamics in Nature and Society*, vol. 2018, 2018.

[50] A. Vitetta, "The importance of modeling path choice behavior in the vehicle routing problem," *Algorithms*, vol. 16, no. 1, 2023.

[51] G. Musolino, A. Polimeni, C. Rindone, and A. Vitetta, "Travel time forecasting and dynamic routes design for emergency vehicles," *Procedia - Social and Behavioral Sciences*, vol. 87, pp. 193–202, 2013.

[52] G. Musolino, A. Polimeni, and A. Vitetta, "Freight vehicle routing with reliable link travel times: a method based on network fundamental diagram," *Transportation Letters*, vol. 10, no. 3, pp. 159–171, 2018.

[53] A. Polimeni and A. Vitetta, "Optimising waiting at nodes in time-dependent networks: Cost functions and applications," *Journal of Optimization Theory and Applications*, vol. 156, 03 2013.

[54] M. W. Ulmer, J. C. Goodson, D. C. Mattfeld, and B. W. Thomas, "On modeling stochastic dynamic vehicle routing problems," *EURO Journal on Transportation and Logistics*, vol. 9, no. 2, p. 100008, 2020.

[55] Uber, "Uber marketplace matching," 2022. [Online]. Available: https://www.uber.com/us/en/marketplace/matching/

[56] M. W. Ulmer, J. C. Goodson, D. C. Mattfeld, and M. Hennig, "Offline–online approximate dynamic programming for dynamic vehicle routing with stochastic requests," *Transportation Science*, vol. 53, no. 1, pp. 185–202, 2019.

[57] M. Hyland, F. Dandl, K. Bogenberger, and H. Mahmassani, "Integrating demand forecasts into the operational strategies of shared automated vehicle mobility services: spatial resolution impacts," *Transportation Letters*, vol. 12, no. 10, pp. 671–676, 2020.

[58] R. Z. Farahani and M. Hekmatfar, *Facility location: concepts, models, algorithms and case studies*. Springer Science & Business Media, 2009.

[59] G. Berbeglia, J.-F. Cordeau, and G. Laporte, "Dynamic pickup and delivery problems," *European Journal of Operational Research*, vol. 202, no. 1, pp. 8–15, 2010.

[60] V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia, "A review of dynamic vehicle routing problems," *European Journal of Operational Research*, vol. 225, no. 1, pp. 1–11, 2013.

[61] H. N. Psaraftis, M. Wen, and C. A. Kontovas, "Dynamic vehicle routing problems: Three decades and counting," *Networks*, vol. 67, no. 1, pp. 3–31, 2016.

[62] Çağrı Koç, T. Bektaş, O. Jabali, and G. Laporte, "Thirty years of heterogeneous vehicle routing," *European Journal of Operational Research*, vol. 249, no. 1, pp. 1–21, 2016.

[63] A. Goel and V. Gruhn, "Solving a dynamic real-life vehicle routing problem," *Operations Research Proceedings*, vol. 2005, pp. 367–372, 01 2005.

[64] V. Pillac, C. Guéret, and A. Medaglia, "A fast reoptimization approach for the dynamic technician routing and scheduling problem," *Operations Research/Computer Science Interfaces Series*, pp. 347–367, 01 2018.

[65] F. Ferrucci and S. Bock, "Real-time control of express pickup and delivery processes in a dynamic environment," *Transportation Research Part B: Methodological*, vol. 63, pp. 1–14, 2014.

[66] M. Schyns, "An ant colony system for responsive dynamic vehicle routing," *European Journal of Operational Research*, vol. 245, no. 3, pp. 704–718, 2015.

[67] M. W. Ulmer and B. W. Thomas, "Same-day delivery with heterogeneous fleets of drones and vehicles," *Networks*, vol. 72, no. 4, pp. 475–505, 2018.

[68] X. Chen, M. W. Ulmer, and B. W. Thomas, "Deep q-learning for same-day delivery with vehicles and drones," *European Journal of Operational Research*, vol. 298, no. 3, pp. 939–952, 2022.

[69] M. W. Ulmer, "Dynamic pricing and routing for same-day delivery," *Transportation Science*, vol. 54, no. 4, pp. 1016–1033, 2020.

[70] I. Dayarian, M. Savelsbergh, and J.-P. Clarke, "Same-day delivery with drone resupply," *Transportation Science*, vol. 54, no. 1, pp. 229–249, 2020.

[71] S. H. Chung, B. Sah, and J. Lee, "Optimization for drone and drone-truck combined operations: A review of the state of the art and future directions," *Computers and Operations Research*, vol. 123, p. 105004, 2020.

[72] M. ApS, *MOSEK for c++. Version 7.1.*

[73] M. Sahnoun, Y. Xu, F. B. Abdelaziz, and D. Baudry, "Optimization of transportation collaborative robots fleet size in flexible manufacturing systems," in *2019 8th International Conference on Modeling Simulation and Applied Optimization (ICMSAO)*, 2019, pp. 1–5.

[74] B. Golden, A. Assad, L. Levy, and F. Gheysens, "The fleet size and mix vehicle routing problem," *Computers &Operations Research*, vol. 11, no. 1, pp. 49–66, 1984.

[75] G. Pantuso, K. Fagerholt, and L. M. Hvattum, "A survey on maritime fleet size and mix problems," *European Journal of Operational Research*, vol. 235, no. 2, pp. 341–349, 2014.

[76] G. C. de Bittencourt, R. D. Seimetz Chagas, V. A. Silva, I. G. Peres Vianna, R. P. Longhi, P. C. Ribas, and V. J. M. Ferreira Filho, "A solution framework for the integrated problem of cargo assignment, fleet sizing, and delivery planning in offshore logistics," *Computers & Industrial Engineering*, vol. 161, p. 107653, 2021.

[77] I. F. A. Vis, R. M. B. M. de Koster, and M. W. P. Savelsbergh, "Minimum vehicle fleet size under time-window constraints at a container terminal," *Transportation Science*, vol. 39, no. 2, pp. 249–260, 2005.

[78] H. Zhang, C. J. R. Sheppard, T. E. Lipman, and S. J. Moura, "Joint fleet sizing and charging system planning for autonomous electric vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 11, pp. 4725–4738, 2020.

[79] M. Xu and Q. Meng, "Fleet sizing for one-way electric carsharing services considering dynamic vehicle relocation and nonlinear charging profile," *Transportation Research Part B: Methodological*, vol. 128, pp. 23–49, 2019.

[80] D. Banerjee, A. L. Erera, and A. Toriello, "Fleet sizing and service region partitioning for same-day delivery systems," *Transportation Science*, vol. 56, no. 5, pp. 1327–1347, 2022.

[81] A. Wallar, J. Alonso-Mora, and D. Rus, "Optimizing vehicle distributions and fleet sizes for shared mobility-on-demand," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 3853–3859.

[82] C. Wang, Y. Song, Y. Wei, G. Fan, H. Jin, and F. Zhang, "Towards minimum fleet for ridesharing-aware mobility-on-demand systems," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, 2021, pp. 1–10.

[83] B. Qu, L. Mao, Z. Xu, J. Feng, and X. Wang, "How many vehicles do we need? fleet sizing for shared autonomous vehicles with ridesharing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 14 594–14 607, 2022.

[84] P. JONES and J. ZYDIAK, "The fleet design problem," *The Engineering Economist*, vol. 38, no. 2, pp. 83–98, 1993.

[85] A. Zhao, J. Xu, J. Salazar, W. Wang, P. Ma, D. Rus, and W. Matusik, "Graph grammar-based automatic design for heterogeneous fleets of underwater robots," *2022 International Conference on Robotics and Automation (ICRA)*, pp. 3143–3149, 2022.

[86] A. Rjeb, J.-P. Gayon, and S. Norre, "Sizing of a homogeneous fleet of robots in a logistics warehouse," *17th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2021*, vol. 54, no. 1, pp. 552–557, 2021.

[87] I. Markov, R. Guglielmetti, M. Laumanns, A. Fernández-Antolín, and R. de Souza, "Simulation-based design and analysis of on-demand mobility services," *Transportation Research Part A: Policy and Practice*, vol. 149, pp. 170–205, 2021.

[88] P. M. Boesch, F. Ciari, and K. W. Axhausen, "Autonomous vehicle fleet sizes required to serve different levels of demand," *Transportation Research Record*, vol. 2542, no. 1, pp. 111–119, 2016.

[89] D. Fagnant and K. Kockelman, "Dynamic ride-sharing and fleet sizing for a system of shared autonomous vehicles in austin, texas," *Transportation*, vol. 45, 01 2018.

[90] A. Fielbaum, A. Tirachini, and J. Alonso-Mora, "Economies and diseconomies of scale in on-demand ridepooling systems," *Economics of Transportation*, vol. 34, p. 100313, 2023.

[91] Y. Mei, Y.-H. Lu, Y. Hu, and C. Lee, "Determining the fleet size of mobile robots with energy constraints," in *2004 IEEE International Conference on Intelligent Robots and Systems (IROS)*, vol. 2, 2004, pp. 1420–1425.

[92] G. Wang, "Comparative study on solving the minimum fleet of shared autonomous vehicles," *CICTP 2020*, pp. 522–534.

[93] Y. Yang, Z. Yuan, X. Fu, Y. Wang, and D. Sun, "Optimization model of taxi fleet size based on gps tracking data," *Sustainability*, vol. 11, no. 3, p. 731, Jan 2019.

[94] H. Zhang, C. J. R. Sheppard, T. E. Lipman, and S. J. Moura, "Joint fleet sizing and charging system planning for autonomous electric vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 11, pp. 4725–4738, 2020.

[95] J. C. Castillo, D. Knoepfle, and G. Weyl, "Surge pricing solves the wild goose chase," in *Proceedings of the 2017 ACM Conference on Economics and Computation*, ser. EC '17, 2017, p. 241–242.

[96] M. Schröder, D.-M. Storch, P. Marszal, and M. Timme, "Anomalous supply shortages from dynamic pricing in on-demand mobility," *Nature Communications*, vol. 11, 09 2020.

[97] J. E. Hopcroft and R. M. Karp, "An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs," *SIAM Journal on Computing*, vol. 2, no. 4, pp. 225–231, 1973.

[98] S. Hao, X. Liu, L. Miao, W. K. V. Chan, and M. Qi, "Qualifying the benefits of ride-sharing on reducing fleet size," *Journal of Physics: Conference Series*, vol. 1903, no. 1, p. 012019, apr 2021.

[99] R. Auad-Perez and P. Van Hentenryck, "Ridesharing and fleet sizing for on-demand multimodal transit systems," *Transportation Research Part C: Emerging Technologies*, vol. 138, p. 103594, 2022.

[100] G. Wang, "Research on the fleet size of shared autonomous vehicles in the future city: An example in shanghai," *CICTP 2020*, pp. 4537–4549.

[101] M. Balac, S. Hörl, and K. W. Axhausen, "Fleet sizing for pooled (automated) vehicle fleets," *Transportation Research Record*, vol. 2674, no. 9, pp. 168–176, 2020.

[102] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2023. [Online]. Available: https://www.gurobi.com

[103] B. Donovan and D. Work, "New york city taxi trip data (2010-2013)," 2016. [Online]. Available: https://databank.illinois.edu/datasets/IDB-9610843

[104] P. Santi, G. Resta, M. Szell, S. Sobolevsky, S. Strogatz, and C. Ratti, "Quantifying the benefits of vehicle pooling with shareability networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 111, 09 2014.

[105] M. Čáp and J. Alonso-Mora, "Multi-objective analysis of ridesharing in automated mobility-on-demand," in *Proceedings of Robotics: Science and Systems*, 2018.

[106] M. Kronmueller, A. Fielbaum, and J. Alonso-Mora, "Fleet sizing for the flash delivery problem from multiple depots a case study in amsterdam," *arXiv*, 2023.

[107] A. Fielbaum, A. Tirachini, and J. Alonso-Mora, "Economies and diseconomies of scale in on-demand ridepooling systems," *Economics of Transportation*, vol. 34, p. 100313, 2023.

[108] R. Shao and L. Chang, "A new maximum power point tracking method for photovoltaic arrays using golden section search algorithm," in *2008 Canadian Conference on Electrical and Computer Engineering*, 2008, pp. 619–622.

Acknowledgements

I am deeply grateful to Javier Alonso-Mora, whose guidance and mentorship throughout the four years have been invaluable. His support and provision of everything I needed have been instrumental in shaping this dissertation. I extend my sincere appreciation to Robert Babuska for providing high-level guidance and being approachable whenever I needed advice. A special thanks to Andres Fielbaum for his invaluable feedback, explanations, and inspiring ideas. His contributions have significantly influenced the development of this dissertation and me, as a researcher.
I would like to express my gratitude to Ahold Delhaize for their financial support and for making the AIRLab project possible. The connections and discussions forged over the course of four years have been immensely valuable. I am thrilled that my Ph.D. project was part of the AIRLab project. Especially, I want to sincerely thank Bassem Zaarour, Bart Voorn, and Thomas Rodenberg for their support, collaboration, and constant efforts to make connections and to organize valuable data.
My gratitude also goes out to all the members of the Autonomous Multi Robots Lab. Sharing this journey with all of you, whether through discussions, coffee breaks, or moments of fun, has been a true pleasure.
I am deeply thankful to my friends whose unwavering support and encouragement have been a constant source of strength. A special shoutout to Daniel Kolb for his motivating presence and ability to bring a smile to my face, no matter the circumstances. Friedrich Sbresny, thank you for your willingness to share and discuss matters, even during challenging times. To Max Spahn and Merle Losch, thank you for the enjoyable board game sessions, bike rides, and dinners. Your company has made my Ph.D. journey all the more delightful. I owe a debt of gratitude to Alvaro Serra Gomez and Corrado Pezzato for getting me out of the office and to the bouldering wall. Your friendship and the chats we shared, whether during these activities or over coffee, have been uplifting.
My family's support has been a pillar of strength throughout this journey. I am immensely grateful to my parents for their unwavering support, belief in me, and encouragement, regardless of my plans and ambitions. You have played a significant role in shaping me into the person I am today. To my two incredible sisters, I am continuously amazed and grateful to have you in my life.
Finally, I want to express my deepest appreciation to Antonia Huefner, who has been the best thing that ever happened to me. Her constant support, love, and presence have brought immeasurable happiness to this journey. I am incredibly fortunate to have her in my life.

Curriculum Vitæ

Maximilian Kronmüller was born in Stuttgart, Germany, in 1992. He earned his Master of Science degree in "Applied and Engineering Physics" from the Technical University of Munich in 2018. During his master's thesis titled "Application of deep neural networks to event type classification in IceCube" he joined the "Experimental Physics with Cosmic Particles" group led by Prof. Elisa Resconi. Subsequently, he continued as a researcher in the same group. His work was focused on developing algorithms and methodologies to analyze and interpret data from the IceCube neutrino detector.

In September 2019, Maximilian began his Ph.D. journey under the supervision of Prof. Javier Alonso-Mora, becoming a member of the Department of Cognitive Robotics at Delft University of Technology. His Ph.D. project centered on vehicle routing and fleet sizing for Flash Delivery operations, hosted within the "AIRLab - AI for Retail" initiative.

Maximilian's research interests encompass a wide range of topics, including Flash Delivery services, dynamic vehicle routing problems, online multi-task assignment, fleet sizing, learning-based methods for routing decisions, and dynamic optimization.

List of Publications

Published

7. **M. Kronmueller**, A. Fielbaum, J. Alonso-Mora, *"Online Flash Delivery from Multiple Depots"*, in Transportation Letters, vol. 0, no. 0, pp. 1-17, 2023.

6. X. Bai, A. Fielbaum, **M. Kronmueller**, L. Knoedler and J. Alonso-Mora, *"Group-Based Distributed Auction Algorithms for Multi-Robot Task Assignment"*, in IEEE Transactions on Automation Science and Engineering, vol. 20, no. 2, pp. 1292-1303, 2023.

5. **M. Kronmueller**, A. Fielbaum and J. Alonso-Mora, *"Routing of Heterogeneous Fleets for Flash Deliveries via Vehicle Group Assignment"*, in IEEE 25th International Conference on Intelligent Transportation Systems (ITSC), pp. 2286-2291, 2022.

4. A. Fielbaum, **M. Kronmueller**, J. Alonso-Mora, *"Anticipatory Routing Methods for an On-Demand Ridepooling Mobility System"*, in Transportation, vol. 49, pp. 1921–1962, 2022.

3. J. Pierotti, **M. Kronmueller**, J. Alonso-Mora, T. van Essen, W. Boehmer, *"Reinforcement Learning for the Knapsack Problem"*, in Optimization and Data Science: Trends and Applications, pp. 3-13, 2021.

2. **M. Kronmueller**, A. Fielbaum, J. Alonso-Mora, *"On-Demand Grocery Delivery From Multiple Local Stores With Autonomous Robots"*, in Proceedings of the International Symposium on Multi-Robot and Multi-Agent Systems (MRS), pp.29-37, 2021.

1. J. van Lochem, **M. Kronmueller**, P. van 't Hof, J. Alonso-Mora, *"Anticipatory Vehicle Routing for Same-Day Pick-up and Delivery using Historical Data Clustering"*, in IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), pp. 1-6, 2020.

Under Review

1. **M. Kronmueller**, A. Fielbaum, J. Alonso-Mora, *"Reducing the Minimal Fleet Size by Delaying Individual Tasks"*, submitted to IEEE Transactions on Intelligent Transportation Systems, 2023.

In Preparation

1. **M. Kronmueller**, A. Fielbaum, J. Alonso-Mora, *"Fleet Sizing for the Flash Delivery Problem from Multiple Depots a Case Study in Amsterdam"*, planned to be submitted to IEEE 27th International Conference on Intelligent Transportation Systems (ITSC)