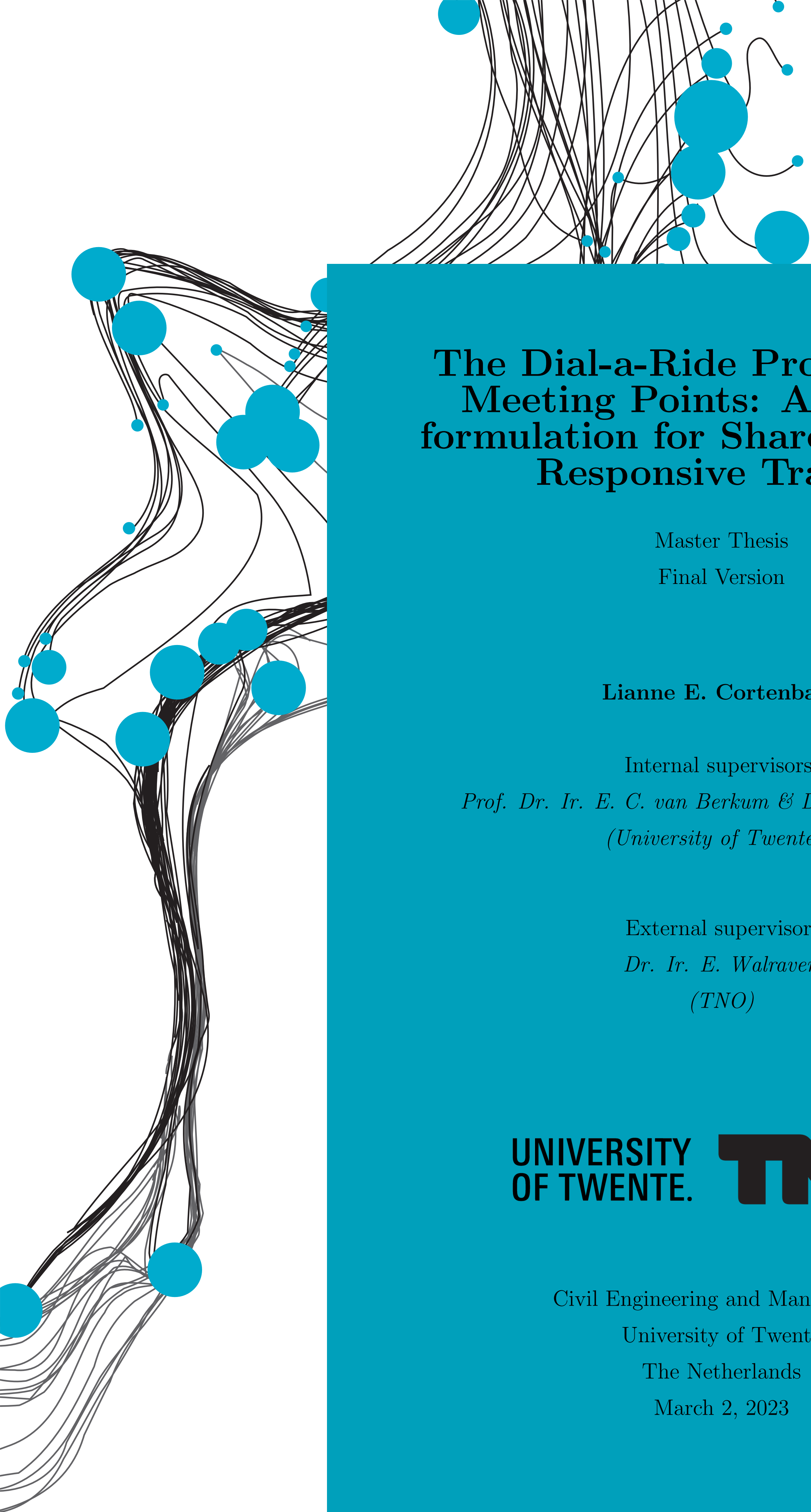


The Dial-a-Ride Problem with Meeting Points: A problem formulation for Shared Demand Responsive Transit

Master Thesis

Final Version

Lianne E. Cortenbach

Internal supervisors:

Prof. Dr. Ir. E. C. van Berkum & Dr. K. Gkiotsalitis

(University of Twente)

External supervisor:

Dr. Ir. E. Walraven

(TNO)

Civil Engineering and Management

University of Twente

The Netherlands

March 2, 2023

I. Contact information

Author	Lianne E. Cortenbach Bsc
Student number	s1864300
Email address	lcortenbach@live.nl
Comissioned by	TNO - Department Sustainable Urban Mobility and Safety
Internship	September 2022 - March 2023
External supervisor	Dr. Ir. E. Walraven
Internal Supervisors	Prof. Dr. Ir. E. C. van Berkum
	Dr. K. Gkiotsalitis
University	University of Twente
Master	Civil Engineering and Management
Track	Transport Engineering and Management
Version	Final Version

II. Preface

This thesis is the final product to obtain a Master Degree in Civil Engineering and Management at the University of Twente. In the last months of my master, I wrote this master thesis during the internship at TNO. I am very happy that I could combine my passion for optimization and traffic modeling in this thesis, by formulating a Mixed-Integer Linear Program for the Dial-a-Ride Problem with Meeting Points as an assignment method for Demand-Responsive-Transit.

First of all, I would like to thank the colleagues at TNO for being very open and interested and giving me the opportunity to get to know TNO as an organisation. This all made doing this master thesis research a very pleasant experience. In particular I want to thank Erwin for his supervision, enthusiasm, feedback and discussions we had about the research. This really helped to bring the research to a higher level.

Next, I want to thank Konstantinos Gkiotsalitis for his daily supervision from the University of Twente, for always encouraging me to look further into the topic and for providing valuable feedback regarding the operations research topics of this thesis.

I also want to thank Eric van Berkum, first of all for guiding me in the right direction for finding a fitting thesis assignment. Besides, thank you for overseeing the process, the feedback for both the research proposal and the final thesis and being part of the graduation committee.

Lastly, I want to thank Lenneke, Frans, Emiel and Niek for their huge support, not only during this thesis period but during the entire studies. I appreciate all your advice and feedback, the discussions we had about the content and the process but most importantly thank you for the mental support that I have always felt.

With this thesis I hope that I have contributed to the knowledge about assignment methods for Demand Responsive Transit and the Dial-a-Ride problem in particular.

Lianne Cortenbach

Rotterdam, March 2, 2023

III. Samenvatting

Recente ontwikkelingen in de communicatie technologie maken vraagafhankelijk vervoer toegankelijker. In de versie waar meerdere passagiers een voertuig delen komt het vaak voor dat een voertuig moet omrijden om passagiers op te halen en weg te brengen. Dit zorgt voor een toename in de reistijd voor zowel de passagiers als de vervoerder zelf. Dit probleem kan worden voorkomen als passagiers een korte afstand van hun herkomst naar een ophaallocatie en van een wegbrenglocatie naar hun bestemming lopen. Deze alternatieve ophaal- en wegbrengpunten worden ook wel 'meeting points' genoemd.

Het doel van dit onderzoek is om het vinden van optimale 'meeting points' te integreren in een bekende probleemformulering voor vraagafhankelijk vervoer. Dit probleem is het Dial-a-Ride Problem (DARP). Het DARP heeft als doel om optimale routes te vinden voor een aantal reisaanvragen van passagiers met verschillende herkomsten en bestemmingen. Een nieuw geheeltallig lineair programmeringsmodel is ontwikkeld. Dit model wordt het Dial-a-Ride problem met meeting points genoemd (DARPmp).

Het vinden van optimale 'meeting points' is opgenomen in het DARP door sets, parameters, variabelen en begrenzingen toe te voegen. Twee voorbewerkingsstappen en twee geldige ongelijkheden zijn geïntroduceerd. Deze kunnen de tijd die nodig is om de optimale oplossing te vinden versnellen. Twee meta-heuristieken zijn ontwikkeld om de optimale oplossing voor het DARPmp bij een groot aantal reisaanvragen te benaderen. Er is gekozen om gebruik te maken van een Tabu Search structuur. In de eerste versie van de Tabu Search wordt eerst gezocht naar een zo goedkoop mogelijke route langs alle herkomsten en bestemmingen, en daarna worden meeting points geintroduceerd. In de tweede versie van de Tabu Search worden meeting points gezocht gedurende het hele zoekproces naar een goedkopere route.

Alle algorithmen zijn toegepast op 12 testnetwerken met 2 tot 4 voertuigen, 16 tot 48 reisaanvragen en 24 tot 72 meeting points. Bij gebruik van de voorbewerkingsstappen en de geldige ongelijkheden is voor de kleinste 5 testnetwerken een optimale oplossing gevonden. Beide Tabu Search algoritmen vinden een oplossing voor alle 12 testnetwerken. De eerste Tabu Search vindt oplossingen met een verschil in kosten variërend van 0% tot 1,65% ten opzichte van de optimale oplossing. De tijd die het algoritme erover doet om een oplossing te vinden varieert van 14,07 tot 850,51 seconden. De tweede versie vindt oplossingen met een verschil in kosten variërend van 0% tot 2,24% ten opzichte van de optimale oplossing. De tijd die het algoritme erover doet om een oplossing te vinden varieert van 38,20 tot 9051,69 seconden. De meeting points zorgen er in sommige gevallen voor individuele passagiers voor dat de reistijd korter wordt, en voor andere passagiers dat de reistijd langer wordt. Het DARPmp heeft wel de potentie om de totale route kosten te verminderen

IV. Summary

Recent developments in the communication technology make Demand-Responsive Transit (DRT) more accessible for everyone. In the ride-sharing version of DRT, vehicles have to make detours from the shortest path to pickup and drop off passengers, which increases the travel time for both the operator and the passengers. This problem could be avoided if passengers are asked to walk a short distance from their origin to their pickup location and from their drop off location to their destination. These alternative pickup and drop off points are called meeting points.

The aim of this work is to incorporate finding optimal meeting points in a well-known problem formulation for DRT, which is the Dial-a-Ride Problem (DARP). The objective of the DARP is to find optimal routes for a number of trip requests from passengers with different origins and destinations. Incorporating meeting points in the DARP is done by formulating a Mixed-Integer Linear Program, which is called the DARPmp.

The DARP is extended to find optimal meeting points while generating optimal routes by adding sets, parameters, variables and constraints. Two preprocessing steps and two valid inequalities are introduced, which improve the computational performance when solving the DARPmp to global optimality. Next to that, two meta-heuristics are proposed to approximate the optimal solution for the DARPmp for a large number of trip requests. A Tabu Search framework is proposed to find solutions for the problem. In the first version of the Tabu Search, a low-cost solution for the DARP is found, for which in the last step the best meeting points are inserted into the routes. For the second version of the Tabu Search, meeting points are inserted in the routes every step of the search proces.

The exact solution method and both Tabu Search algorithms are tested on 12 benchmark instances with 2 to 4 vehicles, 16 to 48 trip requests and 24 to 72 meeting points. When using the preprocessing steps and valid inequalities, an optimal solution is found for the 5 smallest benchmark instances. Both Tabu Search algorithms are able to find a solution for all 12 benchmark instances. The first Tabu Search algorithm finds solutions with an optimality gap ranging from 0% to 1,65%, compared to the optimal solution. The run times for this algorithm range from 14,07 seconds to 850,51 seconds. The second Tabu Search algorithm finds solutions with an optimality gap ranging from 0% to 2,24% and has run times ranging from 38,20 to 9051,69 seconds. On passenger level, the meeting points sometimes decrease the travel time and in other cases they increase the travel time. However, the DARPmp has the potential to minimize the total routing costs.

V. List of Mathematical Definitions

$\in$	in ($i \in I$: item i in set I
$\forall$	for all ($\forall i$: for all items i)
$\sum$	summation
$\mathbb{R}_{\geq 0}$	positive real numbers including 0
$\mathbb{Z}$	$= \{..., -2, -1, 0, 1, 2, ...\}$, the set of all positive and negative integers, including 0
$\lvert A \rvert$	cardinality of a set, meaning the number of items in the set

Contents

List of Figures

List of Tables

1 Introduction

1.1 Demand-Responsive Transit

Demand-Responsive Transit (DRT) is a relatively new mobility concept that distinguishes itself from public transit by providing flexible arrival and departure times and flexible routes, whereas public transit has fixed timetables and fixed routes. Major developments in the communication technology in the past years, like the incorporation of GPS and access to wireless internet for cell phones, allow access to online information and the location of travellers. These developments make this type of transport more accessible for everyone (Masoud & Jayakrishnan, 2017). There are several providers that make DRT possible, from the conventional taxi-services to companies like Uber and Lyft. A pitfall of these DRT services is that they can cause an increase in the number of cars on the road. Trips that were previously made by public transport are replaced by trips in non-shared vehicles (Tirachini & Gomez-Lobo, 2020). Therefore, the incorporation of ride-sharing in these DRT systems has been proposed. In a ride-sharing system, travellers with somewhat similar routes and time schedules share a car to get from their origin to their destination. There are several types of ride-sharing, like carpooling, van-pooling or shared-taxis (Mourad et al., 2019). When properly executed, ride-sharing can have many advantages for society as well as for individual users. Sharing a car results in fewer vehicles on the road and a reduction of the distance travelled by cars, resulting in a reduction of fuel consumption (Jacobsen & King, 2009). Fewer vehicles on the road can also reduce congestion problems (Chen et al., 2019). For individual travellers, sharing a ride can be attractive since their travel costs for gas, toll or parking fees can be reduced because they are split over several persons (Furuhata et al., 2013). The expectation is that the use of these demand-responsive ride-sharing services will increase over the coming years, because there is a wish to mitigate the climate effects of transportation.

A disadvantage of the application of ride-sharing is that passengers are often picked up and dropped off door-to-door. When multiple people share a vehicle, this can result in vehicles having to make detours from the shortest path to pickup and drop off passengers, possibly through slow streets (Fielbaum et al., 2021). This problem could be avoided if passengers are asked to walk a short distance from their origin to their pickup location and from their drop off location to their destination. These alternative pickup and delivery locations are referred to as meeting points. Several studies have shown that introducing meeting points in these ride-sharing system reduce costs terms of travel time and travel distance (Stiglic et al., 2015); (Fielbaum et al., 2021). Figure 1.1 shows an example of a network with meeting points. The figure shows a tour without meeting points on the left, and with meeting points on the right. In the tour in the left part figure some detours are made to pick up and dropoff all passengers. The tour in the right part of the has meeting points included and it shows a more direct route to the destination, without detours.

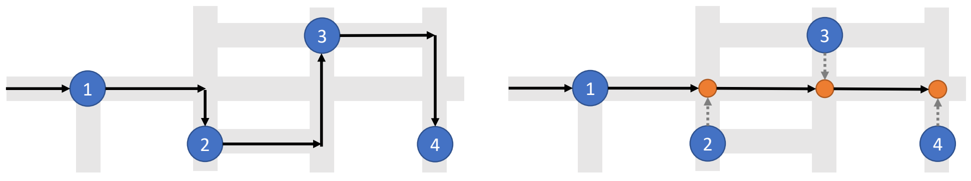


Figure 1.1: Meeting points in a traffic network

Since it is interesting to assess the impact of these services on a traffic network and to make an optimal schedule of this ride-sharing service, several optimization problem formulations have been proposed. One of these formulations is the Dial-a-Ride problem (DARP). The DARP originates from the door-to-door transportation of disabled and elderly people and it can be used to design vehicle routes and schedules corresponding to certain trip requests. Typically, a number of users with specific pickup and drop off requests between an origin and a destination is used as input. The goal of this formulation is to plan routes using several vehicles in an optimal way (Cordeau & Laporte, 2007). This problem can be applied to the ride-sharing DRT services, since this type of transport also deals with requests of travellers that want to go from an origin to a destination in a shared vehicle.

There are multiple different versions of the DARP known. There is a static and a dynamic version in which all requests are known beforehand and requests are revealed during the day respectively (Cordeau & Laporte, 2007). There are also extensions of the model, in which for example transfer possibilities are added (Masson et al., 2014) or battery swapping stations for electric vehicles are incorporated (Masmoudi et al., 2018). A new extension to this model could be the incorporation of the aforementioned meeting points, in a way that the optimal meeting points for trip requests are found, in combination with the optimal route.

1.2 Research questions and contribution

The aim of this research is to extend the DARP to include meeting points. This new formulation will be referred to as the Dial-a-Ride Problem with meeting points (DARPmp). The following research question is formulated: "How should the DARP be extended to find optimal meeting points while generating optimal routes?" This research question is split in the following sub-questions:

1. How should the DARP formulation be adjusted in terms of sets, parameters, variables, objective function and constraints to contain meeting points?
2. What are preprocessing steps and valid inequalities for the DARPmp and what is the impact of those preprocessing steps and valid inequalities on the computational performance of the DARPmp?
3. What meta-heuristic can be used to approximate the optimal solution of the DARPmp?
4. What is the impact of using meeting points on the travel time of the passengers and the objective function value comparing the DARP and the DARPmp?

The contribution of this research is the extension of the DARP in a way that it includes meeting points. This means that the optimal meeting point for a request is found while solving the DARPmp, instead of first defining a meeting point for certain requests and then solving the original DARP. To the best of our knowledge, such an extension does not exist yet. The contribution consists of the formulation of a mixed-integer non-linear program for DARPmp and the linearization of this problem formulation to get a mixed-integer linear program. It also consists of the formulation of a number of preprocessing steps and valid inequalities for the DARPmp. Next to that, two meta-heuristic algorithms are proposed to solve the DARPmp for large instances.

1.3 Thesis structure

The remainder of this thesis is structured as follows. In chapter 2, a literature review about the DARP and meeting points is provided. In chapter 3, a mixed-integer non-linear program is proposed for the DARPmp, which is linearized to a mixed-integer linear program. In this chapter, the sets, parameters, variables, constraints and valid inequalities are described and the application of the DARPmp to a small network is demonstrated. In chapter 4, two meta-heuristics are proposed to approximate the optimal solution of the DARPmp. In chapter 5, the setup and results of the numerical experiments regarding the preprocessing steps and valid inequalities, the exact model and the two meta-heuristics and the impact on the travel time of the passengers are presented. The thesis ends with chapter 6 and 7, in which a discussion and conclusion are provided.

2 Literature Review

In this chapter, a literature review is provided. First, some background information about the DARP is described, as well as some extensions and solution methods of this problem formulation. Next, the application of meeting points and their advantages are presented, as well as a study that combines the DARP and meeting points. In the third section an overview of the literature is provided and lastly, the classic problem formulation of de DARP is given.

2.1 Dial-A-Ride Problem

The DARP is used to design vehicle routes and schedules for travellers that have specific trip requests from certain origins to certain destinations (Cordeau & Laporte, 2007). The original application of this problem arises from the door-to-door transportation of disabled and elderly people. In the most standard form, the trips can be supplied by a fleet of homogeneous vehicles that originate from and return to the same depot. The input for this problem consists of a traffic network represented by a directed graph with corresponding costs and travel times to traverse the arcs. Furthermore there is a capacity for the vehicles and time restrictions for the operation of the vehicles. Next to that there is a set of trip requests from users that want to travel from an origin to a destination. These requests have some characteristics, like a time window for pickup and drop off for example.

The DARP originates from routing problems such as the Traveling Salesman Problem (TSP) and the Capacitated Vehicle Routing Problem (CVRP). The TSP consists of a single vehicle that has to visit all given cities, using the shortest possible route. In the CVRP the aim is to find optimal routes for picking up and delivering certain goods with a limited number of vehicles that have a limited capacity. The DARP is also similar to another problem, which is called the Pickup and Delivery Vehicle Routing Problem (PDVRP) (Cordeau & Laporte, 2007). The PDVRP concerns the transportation of goods from a certain origin to a certain destination. These goods have different characteristics than persons have. They can for example be stalled at some place for a certain time or travel to some place with a large detour, while this is not tolerable for the transportation of humans.

The DARP is extensively researched and it knows many extensions and variants. One of these extensions is the DARP with multiple depots and a heterogeneous fleet of vehicles, in which the vehicles in the fleet can have different characteristics, like the capacity or depot location (Braekers et al., 2014);(Masmoudi et al., 2016); (Malheiros et al., 2021). There are also versions of the DARP in which transfer possibilities between different vehicles are added, which can result in a route schedule with lower travel costs (Masson et al., 2014); (Pierotti & van Essen, 2021). In the light of the developments around electric vehicles, there are also variants of the DARP that use electric vehicles (Masmoudi et al., 2018); (Ma, 2021); (Bongiovanni et al., 2019). These problem formulations focus on the shortcoming of these electric vehicles of having to be charged once in a while, which takes more time than the regular type of fueling. Another development in the field of transportation is the application of autonomous vehicles. Since these do not have the limitations regarding the driver, there are also different formulations of the DARP that consider autonomous vehicles (Pimenta et al., 2017); (Johnsen & Meisel, 2022); (Bongiovanni et al., 2019).

Next to these extensions and variants, different solution methods for the DARP exist. Since the problem becomes harder to solve when the number of trip requests increases, many studies are performed that aim to reduce the computation time for finding an optimal solution for the DARP. Cordeau and Laporte (2003) for example, proposed a Tabu Search heuristic for the static multi-vehicle DARP. Cordeau (2006) used new valid inequalities in a Branch-and-cut algorithm, reducing the CPU time and the number of nodes explored in the

Branch-and-bound tree. A Granular Tabu Search algorithm for the DARP is proposed by Kirchler and Calvo (2013), to produce good solutions for the DARP within 3 minutes. In the study performed by Braekers et al. (2014), a Branch-and-cut algorithm was adapted to solve the Multi-Depot Heterogeneous DARP for small instances, and a Deterministic Annealing meta-heuristic is proposed to solve larger instances. Masmoudi et al. (2017) use a Genetic Algorithm that uses a construction heuristic, crossover operators and local search techniques to find a good solution for the Heterogeneous DARP. An improved Tabu Search heuristic is proposed by Ho et al. (2018), to find a good solution for the static DARP. Using a construction heuristic and time window adjustments, a faster convergence to the optimal solution was observed.

2.2 Meeting points

One of the pitfalls of DRT services in which ride-sharing is used, is that passengers can experience long detours and extra travel time due to the picking up and dropping off of other passengers. This is not only a disadvantage for the passenger point of view, but also for the operator, since the routes become less efficient if large detours have to be made. For this reason, the concept of meeting points is introduced. Meeting points are alternative pickup or drop off locations that are within walking distance from the origin and to the destination of a certain traveller. Introducing meeting points has shown to increase the matches between drivers and riders and decrease the travel costs (Stiglic et al., 2015) ; (Chen et al., 2019) ; (Fielbaum et al., 2021).

Several studies aim to optimize vehicle routes while using meeting points. For example, Chen et al. (2019) proposed an integer linear program for the ride-sharing problem, which has return restrictions, meeting points and transfer possibilities incorporated. The model is created for commuter and business traffic of a closed community of companies. In this problem formulation the meeting points are determined by using the intersection of possible time intervals for the users. This represents the possible time frame for a passenger to be present at a node. If the intersection of the time interval of two requests is non-empty and a common node in the paths to the destination can be found, a meeting point can be created that serves for both requests. Furthermore, Fielbaum et al. (2021) designed a ride-sharing system in which passengers are requested to walk towards a pickup location or from a drop off point to the destination. Feasible meeting points are identified using several constraints, like bounds on the maximum walking time and by ensuring the user has enough time to walk to the meeting point. They used an heuristic approach to solve more complex versions of the problem. They applied the model to a real case in Manhattan, and they showed that short walks can improve the system by reducing the number of trip rejections and vehicle-hours-travelled. Stiglic et al. (2015) investigated the benefits of meeting points by using a maximum weight bipartite matching problem, extended with single driver, multiple rider characteristics. They designed an algorithm that matches drivers and riders in large scale systems and found that introducing meeting points saves travel costs and increases the number of matched participants.

There is also a study that uses meeting points in the DARP. Gökay (2021) researched various forms of the DARP and applied it to both a rural area with low demand and urban areas with high demand. He investigated the possibility to expand the model by trying to reduce small detours of the vehicles by adding meeting points. This is done by analysing historical demand data to determine hot spots that can be used as pickup or drop off locations. These virtual stations can be different in time. After these locations have been determined, the regular DARP is solved. So in this formulation, the location of the vertices V corresponding to the trip requests are changed beforehand, to include the meeting points.

2.3 Literature overview

An overview of the literature used for this thesis is presented in Table 2.1. All studies regarding the DARP and its extensions and and variants and studies about meeting points are listed in this table. The studies are presented with their corresponding base model, extension, objective, exact solution method or meta-heuristic and formulation type. In the last row of this table, the characteristics of this work are presented. For the base model, a distinction is made between the DARP and other models. For the extension of the model, a distinction is made between transfers (TF), Multi-Depot and Heterogeneous vehicles (MDH), Electric Vehicles (EV), Autonomous Vehicles (AUT) and Meeting Points (MP). Regarding the objective function, minimizing the travel costs (MTC) and minimizing the ride time for passengers (MRT) are categorized. Models with another objective function fall in the category 'other'. For the solution method, a distinction is made between exact methods Branch-and-Cut (B&C) and Brand-and-Bound (B&B) and meta-heuristics. For the meta-heuristics, a column is made for Tabu Search (TS), Genetic Algorithms (GA), Simmulated or Deterministic Annealing (S/DA). All other meta-heuristics fall in the category 'other'. For the formulation type, a distiction is made between an Integer Linear Program (ILP) and a Mixed-Integer Linear Program (MILP).

Table 2.1: Literature review

	Base Model		**Extension**					**Objective**			**Exact**			**Meta-Heuristic**				**Formulation**	
Author	*DARP*	*Other*	*TF*	*MDH*	*EV*	*AUT*	*MP*	*MTC*	*MRT*	*Other*	*B&C*	*B&B*	*Other*	*TS*	*GA*	*S/DA*	*Other*	*ILP*	*MILP*
Cordeau et al. (2003)	✓							✓						✓					
Cordeau (2006)	✓							✓			✓								✓
Kirchler et al. (2013)	✓							✓						✓					✓
Breakers et al. (2014)	✓				✓			✓				✓				✓			✓
Masson et al. (2014)			✓							✓							✓		✓
Stiglic et al. (2015)		✓					✓	✓		✓			✓						✓
Masmoudi et al. (2016)	✓			✓				✓									✓		
Masmoudi et al. (2017)	✓							✓							✓				✓
Pimenta et al. (2017)	✓				✓	✓				✓							✓	✓	
Ho et al. (2018)	✓							✓						✓					✓
Masmoudi et al. (2018)	✓				✓			✓									✓		✓
Bongiovanni et al. (2019)	✓				✓	✓		✓	✓		✓								✓
Chen et al. (2019)		✓	✓				✓			✓			✓				✓	✓	
Fielbaum et al. (2021)		✓					✓	✓		✓			✓				✓	✓	
Gökay (2021)	✓						✓	✓									✓		
Ma (2021)	✓				✓					✓							✓		✓
Malheiros et al. (2021)	✓			✓				✓									✓		✓
Pierotti et al. (2021)	✓		✓					✓				✓						✓	
Johnsen et al. (2022)	✓					✓		✓									✓		✓
This work	✓						✓	✓			✓			✓					✓

2.4 Problem formulation

In this section, the three-index formulation of Cordeau (2006) is presented, to use in the remainder of the thesis as the classic formulation of the DARP. This formulation of the DARP aims to minimize the costs for routes that complete all trip requests. The problem formulation consists of several sets. The problem is formulated on a directed graph $G = (V, A)$, where V is the set of vertices and A is the set of arcs. The set V of vertices can be partitioned into three sets, being the set of pickup nodes $P = \{1, 2, ..., n\}$, the set of delivery nodes $D = \{n+1, n+2, ..., 2n\}$ and the set of depots, which is denoted as $\{0, 2n+1\}$, containing two copies of the depot. The sets of pickup and delivery vertices are ordered in such a way that each trip request is coupled as $(i, n+i)$, in which i is the pickup vertex and $n+i$ is the corresponding drop off vertex. In the case that two passengers have the same pickup location but a different drop off location, the corresponding pickup vertex is duplicated. The same holds if two passengers have the same drop off point but another origin. In this case the drop off point is duplicated. When there are multiple passengers that have the same origin-destination pair, the request of these passengers is represented by a pickup and drop off pair. In this case, each pickup vertex i is coupled to drop off vertex $n+i$, which are associated with one origin-destination pair. The arcs in the set A consist of all outgoing arcs from the depot $\{(0, j) : j \in P\}$, all returning arcs to the depot $\{(i, 2n+1) : i \in D\}$ and all arcs between pickup and delivery nodes $\{(i, j) : i \in P \cup D, j \in P \cup D, i \neq j, i \neq n + j\}$. Lastly there is a set K consisting of all available vehicles.

The problem also consists of several parameters. The cost of traveling arcs $(i, j) \in A$ is given by c^k_{ij}, where k represents the vehicle. The travel time over these arcs is given by t_{ij}. Each vehicle has a ride time that cannot exceed the maximum route duration denoted by T_k and a vehicle capacity Q_k that cannot be exceeded. Every vertex has a corresponding time window in which a passenger needs to be picked up or dropped off, denoted by $[e_i, l_i]$. Next to this time window, each vertex also has a service duration $d_i \geq 0$ where the service duration for the depot equals 0. Each vertex also has an associated passenger load $q_i \geq 0$. For the depots $q_0 = q_{2n+1} = 0$. The pickup and delivery passenger loads are associated such that $q_{n+i} = -q_i$ For each passenger, the maximum allowed ride time is L.

Four variables are used in this model. The first variable is $x^k_{ij} \in \{0, 1\}$ for $(i, j) \in A$, $k \in K$, which is used to indicate if arc (i, j) is traversed by vehicle k. The variable equals 1 if the arc is traversed and 0 otherwise. Next, there is a variable $u^k_i \in \mathbb{N}$ for $i \in V$, $k \in K$, which is used to indicate the time at which vehicle k starts serving node i. Furthermore, there is a variable $w^k_i \in \mathbb{N}$ for $i \in V$, $k \in K$ which is used to update the load of vehicle k when leaving node i. Lastly, there is the variable $r^k_i \in \mathbb{N}$ for $i \in V$, $k \in K$ which indicates the ride time of the passenger corresponding to request $(i, n + i)$ on vehicle k.

$$\min \sum_{k\in K}\sum_{(i,j)\in A} c^k_{ij}x^k_{ij} \tag{2.1}$$

$$\text{s.t.} \sum_{k\in K}\sum_{j:(i,j)\in A} x^k_{ij} = 1, \qquad \forall i \in P \tag{2.2}$$

$$\sum_{j\in P} x^k_{0j} = \sum_{i\in D} x^k_{i,2n+1} = 1 \qquad \forall k \in K \tag{2.3}$$

$$\sum_{j:(i,j)\in A} x^k_{ij} - \sum_{j:(i,j)\in A} x^k_{n+i,j} = 0 \qquad i \in P, k \in K \tag{2.4}$$

$$\sum_{j:(j,i)\in A} x^k_{ji} - \sum_{j:(i,j)\in A} x^k_{i,j} = 0 \qquad \forall i \in P \cup D, k \in K \tag{2.5}$$

$$u^k_j \geq (u^k_i + d_i + t_{ij})x^k_{ij} \qquad \forall (i,j) \in A, k \in K \tag{2.6}$$

$$w^k_j \geq (w^k_i + q_j)x^k_{ij} \qquad \forall (i,j) \in A, k \in K \tag{2.7}$$

$$r^k_i = u^k_{n+i} - (u^k_i + d_i) \qquad i \in P, k \in K \tag{2.8}$$

$$u^k_{2n+1} - u^k_0 \leq T_k \qquad \forall k \in K \tag{2.9}$$

$$e_i \leq u^k_i \leq l_i \qquad \forall i \in V, k \in K \tag{2.10}$$

$$t_{i,n+i} \leq r^k_i \leq L \qquad \forall i \in P, k \in K \tag{2.11}$$

$$max\{0, q_i\} \leq w^k_i \leq min\{Q_k, Q_k + q_i\} \qquad \forall i \in V, k \in K \tag{2.12}$$

$$x^k_{ij} \in \{0, 1\} \qquad \forall (i,j) \in A, k \in K \tag{2.13}$$

Constraints 2.2 and 2.4 ensure that each pickup is carried out exactly once and that each pickup and delivery combination will be served by the same vehicle. Constraints 2.5 are the flow conservation constraints and when combined with constraints 2.3, it will make sure that each route will start and end at the depot. Constraints 2.6 make sure the time is tracked correctly, so when arc (i, j) is traversed by vehicle k, the time the vehicle starts it service at vertex j is greater than or equal to the start time at node i plus the service time at node i and the travel time between vertex i and vertex j. Constraints 2.7 ensure the load of the vehicle is updated correctly. Constraints 2.8 ensure the ride time of each passenger is saved in the correct way, including the service time at the pickup location and the start time of serving vertex i and the start time of serving vertex $i + n$. Constraints 2.9 ensure the ride time of a vehicle does not exceed the maximal route duration and constraints 2.10 guarantee that all pickups and drop offs happen in their corresponding time window. Constraints 2.11 ensure the maximal ride time per passenger is not exceeded and constraints 2.12 ensure the capacity of the vehicle is not exceeded.

The classic formulation of the DARP contains two constraints, being 2.6 and 2.7, that make the problem formulation non-linear. Cordeau (2006) proposed to replace these constraints by constraints 2.14 and 2.15, making the model linear.

$$u_j^k \geq u_i^k + d_i + t_{ij} - M_{ij}^k(1 - x_{ij}^k) \qquad \forall (i,j) \in A, k \in K \tag{2.14}$$

$$w_j^k \geq w_i^k + q_j - W_{ij}^k(1 - x_{ij}^k) \qquad \forall (i,j) \in A, k \in K \tag{2.15}$$

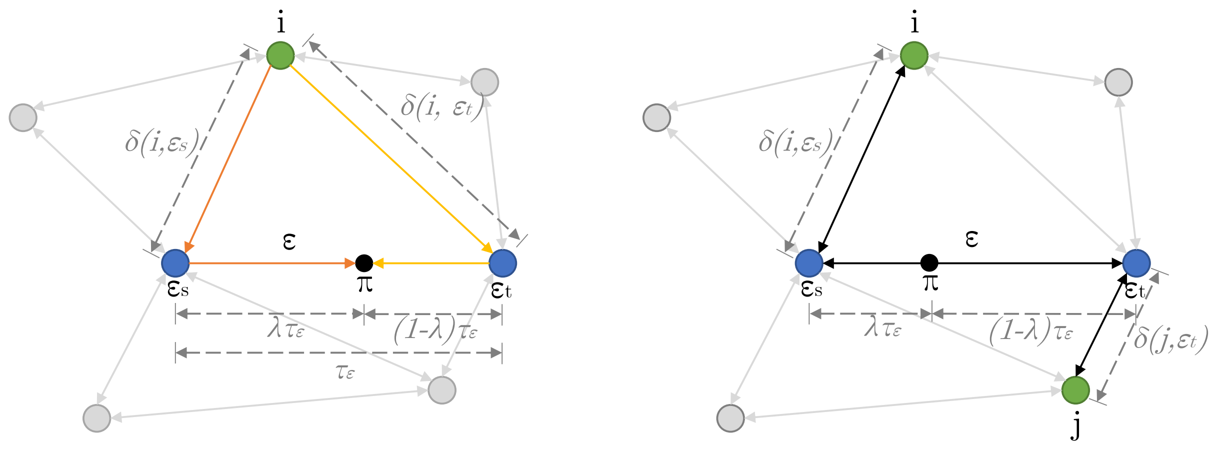
In these constraints, $M_{ij}^k \geq max\{0, l_i + d_i + t_{ij} - e_j\}$ and $W_{ij}^k \geq min\{Q_k, Q_k + q_i\}$, but it suffices to use a very large positive number M instead of these numbers proposed in constraint 2.14 and 2.15.

3 Dial-a-Ride Problem with Meeting Points

In this chapter, a reformulation of the DARP is given such that it contains meeting points. First, a method to generate meeting points in a network is described. Second, the sets and parameters are introduced, after which the variables are described. Next the objective function and constraints are presented and explained. The complexity of the model and some preprocessing steps and valid inequalities are described in the next sections. Lastly, an example of the application of the DARPmp in a small network are presented.

3.1 The M-center method

In order to reformulate the DARP in a way that it contains meeting points, a finite set of new vertices has to be generated. These vertices represent the possible meeting points. The generation of this finite number of points is based on the m-center method, as described by Minieka (1970). The m-center set of a graph is a set that contains m points belonging to the edges or the vertices of this graph. The set minimizes the maximum distance from a vertex to its nearest m-center. The method of generating a finite set of points that is used to find the m-center of a graph is used to generate the possible meeting points in a network.



(a) The shortest distance to a point π in edge ε **(b)** The generation of a point π in edge ε

Figure 3.1: A visual explanation of the M-center method

Let $G = (X, \Gamma)$ be a connected graph with a finite vertex set X, and undirected edge set Γ. An edge $\varepsilon \in \Gamma$ consists of an infinite number of points, characterized by the distance of the point to the endpoints of edge ε. The infinite set of points in edges is denoted as Π. $Q = \Pi \cup X$ is then the infinite set of all points in the graph. The length of an edge is denoted by τ_ε, the starting point of edge ε is ε_s and the endpoint of edge ε is denoted by ε_t. The idea is to reduce the number of points in all edges to a finite number of points. Let $\delta(i, \pi)$ be the shortest distance from vertex $i \in X$ to point π on edge ε. This shortest distance $\delta(i, \pi)$ is a continuous piece-wise linear function. Regarding the endpoints of edge ε, π is situated $\lambda\tau_\varepsilon$ units from ε_s and $(1-\lambda)\tau_\varepsilon$ units from ε_t, as can be seen in Figure 3.1a. The shortest distance from i to π is thus:

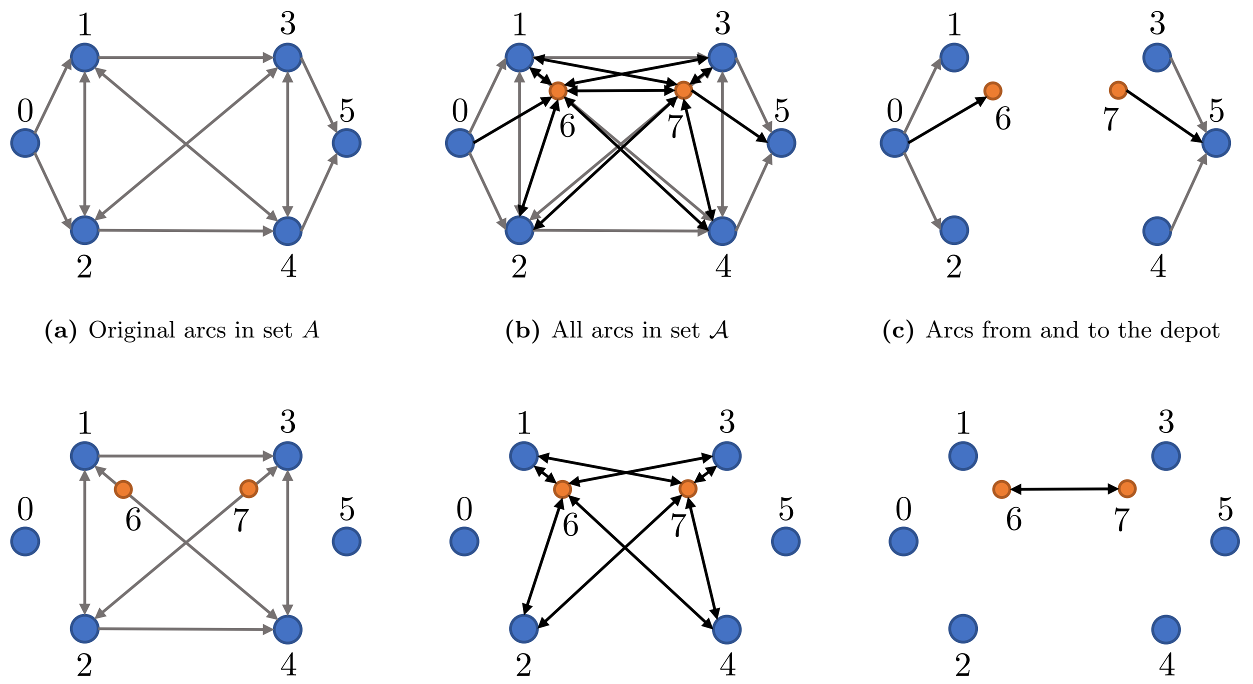
$$d(i, \pi) := \min\{d(i, \varepsilon_s) + \lambda\tau_\varepsilon;\ d(i, \varepsilon_t + (1-\lambda)\tau_\varepsilon\} \tag{3.1}$$

To generate a finite number of points in all edges, $\Pi' \subseteq \Pi$ is defined, which consists of all points π in edges ε, such that for some $i \in X$ and some $j \in X$, where $i \neq j$, π is the unique point in edge ε such that $\delta(i, \pi) = \delta(j, \pi)$. This means that $\delta(i, \varepsilon_s) + \lambda t_\varepsilon = \delta(j, \varepsilon_t) + (1-\lambda)t_\varepsilon$. An example of such a point π is shown in Figure 3.1b. Then the finite set $Q' = \Pi' \cup X$ can be defined. For this set, Minieka (1970) has proven that an m-center set is contained in these points. However for this research, the aim is not to find m-centers in the graph but to use this finite set of points as meeting points in an artificial network.

3.2 Sets

In the DARPmp, there are n trip requests, where a request couple is referred to as $(i, n+i)$. There is a set K, containing all available vehicles. Next, let $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ be a directed graph. The vertex set of the graph $\mathcal{N}$ can be partitioned into $\{\{0, 2n+1\}, P, D, \mathcal{M}\}$. Here, the set $\{0, 2n+1\}$ contains two copies of the depot. The set $P = \{1, ..., n\}$ contains all pick up vertices and the set $D = \{n+1, ..., 2n\}$ contains all drop off vertices. The set $\mathcal{M} = \{2n+2, ..., |\mathcal{N}|\}$ contains the possible meeting points that are not an original pickup or delivery vertex. These are the points that are referred to in the previous section with Π', so all points in edges of graph $\mathcal{G}$ that are not already existing vertices. The set of meeting points $\mathcal{M}$ has two subsets, being the subset with meeting points that are within reach for a pick up vertex $\mathcal{M}_p$ and the subset with meeting points that are within reach for a drop off vertex $\mathcal{M}_d$. These two sets do not have to be distinct, but can have the same vertices. The set that indicates all vertices in the original network without meeting points is referred to as $V = \{P, D, \{0, 2n+1\}\}$. The arc set $\mathcal{A}$ contains the following arcs:

- All outgoing arcs from the depot to pick up vertices and pickup meeting points $\{(0, j) : j \in P \cup \mathcal{M}_p\}$
- All arcs going from a delivery vertex or a delivery meeting point to the depot $\{(i, 2n+1) : i \in D \cup \mathcal{M}_d\}$
- All arcs between pick up and drop off vertices $\{(i, j) : i \in P \cup D, j \in P \cup D, i \neq j, i \neq n+j\}$
- All arcs going from pick up and delivery nodes to meeting point nodes $\{(i, j) : i \in P \cup D, j \in \mathcal{M}\}$
- All arcs going from meeting points to pickup and delivery nodes $\{(i, j) : i \in \mathcal{M}, j \in P \cup D$
- All arcs between meeting points $\{(i, j) : i \in \mathcal{M}, j \in \mathcal{M}, i \neq j\}$



(a) Original arcs in set A

(b) All arcs in set $\mathcal{A}$

(c) Arcs from and to the depot

(d) Arcs between $i \in P \cup D$ and $j \in P \cup D$

(e) Arcs between $i \in P \cup D$ and $j \in \mathcal{M}$, and $i \in \mathcal{M}$ and $j \in P \cup D$

(f) Arcs between $i \in \mathcal{M}$ and $j \in \mathcal{M}$

Figure 3.2: Structure of the graph $\mathcal{G}$ with meeting points

An example of how a graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ is structured is given in Figure 3.2. In this example, there are two requests and two meeting points, where $P = \{1, 2\}$, $D = \{3, 4\}$, $\mathcal{M} = \{6, 7\}$, $\mathcal{M}_p = \{6\}$ and $\mathcal{M}_d = \{7\}$. It is important to note that vertices 0 and 5 represent the depot and are actually at the same physical location but they are displayed as separate locations for the ease of understanding the figure. Arcs that also exist in the original DARP are grey, new arcs are black, pick up and delivery vertices are blue and meeting points are orange.

Figure 3.2a shows the graph for the original DARP. Figure 3.2b shows the new graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ with all arcs included. All other sub-figures show a subset of the arcs to give more insight in what the description of arcs given above means. 3.2c shows all arcs going from and to the depot, 3.2d shows all arcs going between pick up and delivery vertices, 3.2e shows all arcs going between pick up and delivery vertices and meeting points and vice versa and 3.2f shows all arcs going between all meeting points.

3.3 Parameters

Each arc $(i, j) \in \mathcal{A}$ has an associated cost c_{ij}^k when traversed by vehicle k and an associated travel time t_{ij}. Each vehicle cannot exceed the route duration denoted by T_k and can also not exceed their capacity Q_k, where $k \in K$. Each vertex $i \in V$ has a service duration d_i, where $d_i \geq 0$ and $d_0 = d_{2n+1} = 0$. Each vertex also has a passenger load q_i, where $q_i \geq 0$ for $i \in P$. For the delivery vertices $q_{n+i} = -q_i$. Each vertex $i \in V$ has associated time windows $[e_i, l_i]$ and the maximum ride time for passengers is referred to as L.

The meeting point vertices do not have rigid associated time windows, service duration or passenger load, since it is not know beforehand which requests are coupled to which meeting points. To assign these values to the meeting points, several variables and constraints are used that are explained in the next sections. Furthermore, some new parameters are introduced. The maximum walking distance from a vertex to a meeting point or from a meeting point to a vertex is C^{max} and the minimum walking distance from a vertex to a meeting point is C^{min}. The walking time between vertices is referred to as p_{im} for $i, m \in \mathcal{N}$. Lastly, the maximum total walking time in a trip for a passenger is given by W.

3.4 Variables

A binary variable $x_{ij}^k \in \{0, 1\}$ is used to indicate whether a vehicle $k \in K$ traverses an arc $(i, j) \in \mathcal{A}$. $u_i^k \in \mathbb{R}_{\geq 0}$ is a continuous variable that is used to indicate the time at which vehicle $k \in K$ starts servicing vertex $i \in \mathcal{N}$. Variable $w_i^k \in \mathbb{N}$ is used to update the load of vehicle $k \in K$ when leaving vertex $i \in \mathcal{N}$. In the original DARP, variable $r_i^k \in \mathbb{N}$ for $i \in V$, $k \in K$ is used to indicate the ride time of the passenger corresponding to request $(i, n+i)$ on vehicle k. In the DARPmp, the ride time is not captured in a variable, but it is bounded from above and below in another way in a constraint. This is explained in section 3.5.

Moreover, four new variables are introduced. The first variable is a binary variable, being $y_{im}^k \in \{0, 1\}$ which is used to indicate if vertex $i \in V$ is coupled to meeting point $m \in \mathcal{N}$ in vehicle $k \in K$. Note that a vertex can also be coupled to itself, which would mean in practice that no meeting point is used for this vertex. A vertex can also be coupled to a vertex that is in the sets P or D, which means that an original pick up or delivery vertex is used as a meeting point. Next there are two variables that are used to assign the flexible passenger demand and service duration to a meeting point. The first variable is $z_{im} \in \mathbb{Z}$ for $i \in V, m \in \mathcal{N}$ which is used to couple the passenger demand from request i to meeting point m if vertex i is coupled to meeting point m. The second variable is $s_{im} \in \mathbb{R}_{\geq 0}$ for $i \in V, m \in \mathcal{N}$ which is used to couple the service duration from vertex i to meeting point m, if i and m are coupled. Lastly, a variable to indicate the total walking time of a passenger is used, which is $v_i^k \in \mathbb{R}_{\geq 0}$ for $i \in V$ for $k \in K$.

3.5 Mathematical formulation

In this section, the objective function and constraints of the DARPmp are presented and described. The mathematical formulation for the DARPmp is defined as follows:

$$\min \sum_{k\in K}\sum_{(i,j)\in\mathcal{A}} c_{ij}^k x_{ij}^k \tag{3.2}$$

$$\text{s.t.} \sum_{k\in K}\sum_{m\in\mathcal{N}} y_{im}^k = 1, \qquad \forall i \in P\cup D \tag{3.3}$$

$$\sum_{j:(j,m)\in\mathcal{A}} x_{jm}^k \geq y_{im}^k \qquad \forall i\in V, m\in\mathcal{N}, k\in K \tag{3.4}$$

$$\sum_{j\in P\cup\mathcal{M}_p} x_{0j}^k = \sum_{i\in D\cup\mathcal{M}_d} x_{i,2n+1}^k = 1 \qquad \forall k\in K \tag{3.5}$$

$$\sum_{j:(j,i)\in\mathcal{A}} x_{ji}^k - \sum_{j:(i,j)\in\mathcal{A}} x_{ij}^k = 0 \qquad \forall i\in P\cup D\cup\mathcal{M}, k\in K \tag{3.6}$$

$$\sum_{m\in\mathcal{N}} y_{im}^k - \sum_{m\in\mathcal{N}} y_{n+i,m}^k = 0 \qquad \forall i\in P, k\in K \tag{3.7}$$

$$\sum_{m\in\mathcal{N}}\sum_{i\in V} y_{im}^k \geq 1 \qquad \forall k\in K \tag{3.8}$$

$$y_{im}^k + y_{i+n,m}^k \leq 1 \qquad \forall i\in P, k\in K, m\in\mathcal{N} \tag{3.9}$$

$$s_{im} = \sum_{k\in K} y_{im}^k d_i \qquad \forall i\in P\cup D, m\in\mathcal{N} \tag{3.10}$$

$$z_{im} = \sum_{k\in K} y_{im}^k q_i \qquad \forall i\in P\cup D, m\in\mathcal{N} \tag{3.11}$$

$$u_j^k \geq (u_m^k + \sum_{i\in V} s_{im} + t_{mj})x_{mj}^k \qquad \forall (m,j)\in\mathcal{A}, k\in K \tag{3.12}$$

$$w_j^k \geq (w_m^k + \sum_{i\in V} z_{ij})x_{mj}^k \qquad \forall (m,j)\in\mathcal{A}, k\in K \tag{3.13}$$

$$w_0^k = w_{2n+1}^k = 0 \qquad \forall k\in K \tag{3.14}$$

$$u_{2n+1}^k - u_0^k \leq T_k \qquad \forall k\in K \tag{3.15}$$

$$(e_i + p_{im})y_{im}^k \leq u_m^k \leq l_i + (1-y_{im}^k)T_k \qquad \forall i\in P, m\in\mathcal{N}, k\in K \tag{3.16}$$

$$e_i y_{im}^k \leq u_m^k \leq l_i - p_{im} + (1-y_{im}^k)T_k \qquad \forall i\in D, m\in\mathcal{N}, k\in K \tag{3.17}$$

$$\sum_{m\in\mathcal{N}} y_{i+n,m}^k u_m^k - \sum_{m\in\mathcal{N}} (y_{i,m}^k(u_m^k + \sum_{i\in V} s_{im})) \leq L \qquad \forall i\in P, k\in K \tag{3.18}$$

$$t_{m,\mu} \leq \sum_{m\in\mathcal{N}} y_{i+n,m}^k u_m^k - \sum_{m\in\mathcal{N}} (y_{i,m}^k(u_m^k + \sum_{i\in V} s_{im})) \qquad \forall i\in P, k\in K \tag{3.19}$$

$$\max\{0, \sum_{i\in V} z_{im}\} \leq w_m^k \leq \min\{Q_k, Q_k + \sum_{i\in V} z_{im}\} \qquad \forall m\in\mathcal{N}, k\in K \tag{3.20}$$

$$c_{im} y_{im}^k \leq C^{max} \qquad \forall i\in V, m\in\mathcal{N}, k\in K \tag{3.21}$$

$$c_{im} y_{im}^k \geq y_{im}^k C^{min} \qquad \forall i\in V, m\in\mathcal{N}, i\neq m, k\in K \tag{3.22}$$

$$v_i^k = \sum_{m\in\mathcal{N}} y_{im}^k p_{im} + \sum_{\mu\in\mathcal{N}} y_{i+n,\mu} p_{i+n,\mu} \qquad \forall i\in P, k\in K \tag{3.23}$$

$$v_i^k \leq W \qquad \forall i\in P, k\in K \tag{3.24}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall (i,j)\in\mathcal{A}, k\in K \tag{3.25}$$

$$y_{im}^k \in \{0,1\} \qquad \forall i\in V, m\in\mathcal{N}, k\in K \tag{3.26}$$

The objective function 3.2 aims to minimize the total travel costs. Constraints 3.3 ensure that each pick up or delivery vertex is coupled to a meeting point, being itself or another vertex. Constraints 3.4 guarantee that when a vertex is coupled to a meeting point, a vehicle drives to that meeting point. With constraints 3.5, each vehicle starts at the depot and ends at the depot. Constraints 3.6 are the flow conservation constraints, that ensure that a vehicle can only drive to a vertex if it also leaves this vertex. These constraints does not hold for the depot vertices. To ensure that a passenger is picked up and dropped off by the same vehicle, constraints 3.7 are included. Constraints 3.8 ensure that each vehicle fulfills at least one passenger request, to prevent that a vehicle only drives to a meeting point and back. It is not allowed that a passenger has to walk the entire trip he or she requests, therefore constraints 3.9 are added, which forbids the pick up vertex and the delivery vertex to be coupled to the same meeting point.

The next constraints deal with updating the time and passenger load at the corresponding meeting points. First, constraints 3.10 are used to couple the service duration of a vertex i to the meeting point m when vertex i is coupled to meeting point m. This is the case when $y^k_{im} = 1$. Lets take as an example $i = 1$. By using constraints 3.3 there is only one meeting point μ coupled to vertex 1, so $y^k_{1\mu} = 1$. Because all other y^k_{1m} for $m \in \mathcal{N}$ have to be 0, $s_{1\mu}$ takes the value of the service time d_1 of vertex 1. All other s_{1m} are equal to 0. Constraints 3.11 do the same, but for the passenger load. If a vertex is a delivery vertex where $q_{n+i} = -q_i$, z_{im} can also take a negative value.

For updating the time and passenger loads at all vertices in $\mathcal{N}$, constraints 3.12 and 3.13 are used. Constraints 3.12 ensure that if arc $(m, j) \in \mathcal{A}$ is traversed by vehicle k, then the time the vehicle arrives at vertex j is larger than or equal to the sum of the time the vehicle leaves vertex m plus the time it takes for all passengers to get in $(\sum_{i \in V} s_{im})$, plus the time it takes to travel from vertex i to vertex m (t_{mj}). The summation $(\sum_{i \in V} s_{im})$ is used because multiple pick up or delivery vertices can be coupled to the same meeting point. Constraints 3.13 work in the same way, but are used for the passenger load of the vehicle. When arc (m, j) is traversed by vehicle k, the load at vertex j should be larger than or equal to the sum of the load at vertex m plus the total amount of passengers that will get in at vertex j $(\sum_{i \in V} z_{ij})$. Constraints 3.14 are used to ensure that the vehicle is empty when leaving and entering the depot and constraints 3.15 ensure that the maximal route duration T_k is not exceeded.

The time windows are incorporated in constraints 3.16 and 3.17. These constraints ensure that if a vertex $i \in P$ is coupled to meeting point m $(y^k_{im} = 1)$, the time window $[e_i, l_i]$ is coupled to the meeting point m. This constraint ensures that the time windows are enforced if meeting point m is coupled to pick up or delivery vertex i. Since a passenger has to walk from its origin to a meeting point, or from a meeting point to its destination, these time windows have to be adjusted. In constraints 3.16, all pick up requests are captured, where the walking time to the meeting point is included in the time window. The time a person can be picked up at meeting point m will be as late as the start time of the time window of vertex i plus the walking time from vertex i to meeting point m. Constraints 3.17 capture the delivery vertices, where a person has to be dropped off at latest at the end time of the time window of vertex i, minus the time it takes to walk from meeting point m to vertex i.

Constraints 3.18 and 3.19 ensure that the ride time remains within its boundaries. It is bounded from above by the maximum ride time L and bounded from below by the time it takes to get from the vertex where the passenger is picked up to the vertex where the passenger is dropped of. This also enforces a pick up to be carried out before a drop off. Both constraints contain two summations. The first summation takes the value of the time vehicle k arrives at the meeting point that is coupled to vertex $n + i$. The second summation takes the value of the time vehicle k leaves vertex i. This works since constraint 3.2 enforces that each pick up and drop off vertex is coupled to exactly one meeting point $m \in \mathcal{N}$, in only one vehicle. Therefore, there is at most one y^k_{im} that is equal to 1. If this request is not served by vehicle k, both summations are equal to 0. If vehicle k serves this request, the left hand side in equation 3.18 and the right hand side in 3.19 are equal to the ride time of request i.

Constraints 3.20 ensure that the load of the vehicle remains within its boundaries. It can never be smaller than 0 and when passengers get in at node m, it cannot be smaller than the number of passengers that enter at this vertex. It also never exceeds its capacity Q_k, but when passengers get out it cannot be larger than the capacity Q_k minus the number of passengers that get out of the vehicle at vertex m.

The following constraints apply to the walking distances and times of the passengers. Constraints 3.21 ensure that passengers do not walk more than the maximum walking distance C and constraints 3.22 ensure that passengers do not walk less than C^{min}, if they are not picked up of dropped of at their original pickup or drof off location. The total walking time variable v^k_i is updated by constraints 3.23, where the walking time of pick up vertex i to meeting point m is added to the walking time of meeting point μ to delivery vertex $i+n$, if vertex i is coupled to meeting point m and vertex $i+n$ is coupled to meeting point μ. Constraints 3.24 bound the total walking time of a passenger from above by the maximum walking time W.

In the proposed model, constraints 3.12, 3.13, 3.18 and 3.19 are nonlinear. To linearize these constraints, big-M formulations are used. For constraints 3.12 and 3.13 where the time and passenger load of the vehicle is updated, the linearization is done in the same way as in the original formulation by Cordeau (2006), by using big-M constraints. The new constraints can be found in formulas 3.27 and 3.28.

$$u^k_j \geq u^k_m + \sum_{i \in V} s_{im} + t_{mj} - M(1 - x^k_{mj}) \qquad \forall (m,j) \in \mathcal{A}, k \in K \tag{3.27}$$

$$w^k_j \geq w^k_m + \sum_{i \in V} z_{ij} - M(1 - x^k_{mj}) \qquad \forall (m,j) \in \mathcal{A}, k \in K \tag{3.28}$$

For constraints 3.18 and 3.19 that ensure the ride time of the passenger is bounded from above and below, linearization is also necessary. Both constraint sets work with big-M formulations. Constraints 3.29 ensure that if vertex $n+i$ is coupled to meeting point μ and vertex i is coupled to meeting point m, the time the vehicle enters vertex μ minus the time the vertex enters vertex m plus the total service time at vertex m, must be smaller than the maximum ride time L. Constraints 3.30 do the same, but they ensure this equation is larger than the travel time from m to μ.

$$u_\mu - M(1 - y^k_{i+n,\mu}) - (u_m + \sum_{i \in V} s_{im}) - M(1 - y^k_{i,m}) \leq L \qquad \forall i \in P, k \in K, m, \mu \in \mathcal{N} \tag{3.29}$$

$$t_{m,\mu} \leq u_\mu + M(1 - y^k_{i+n,\mu}) - (u_m + \sum_{i \in V} s_{im}) + M(1 - y^k_{i,m}) \qquad \forall i \in P, k \in K, m, \mu \in \mathcal{N} \tag{3.30}$$

3.6 Complexity

The complexity of this problem is dependent on the number of integer variables in relation to the input size of the problem. There are 2 types of integer variables in the problem, which are in different ways dependent on the input size. For the x-variable set that describes whether an arc is traversed or not, the number of variables is $|\mathcal{A}| \times |K|$. For the set of y-variables that describe to which meeting point a vertex is coupled in which vehicle, the number of variables is $|\mathcal{N}| \times |K| \times |V|$. The total number of integer variables for the DARPmp therefore equals:

$$|K| \times (|\mathcal{A}| + |\mathcal{N}| \times |V|) \tag{3.31}$$

3.7 Preprocessing and valid inequalities

In this section, two preprocessing steps and 2 valid inequalities for the DARPmp are described. These preprocessing steps and valid inequalities do not remove integer feasible solutions from the solution space for the model described in 3.2 to 3.30 but they exclude solutions that are infeasible for the integer constraints. They are able to strengthen the LP-relaxation of the model. By removing these solutions, the search process can become more focused, which can result in lower computational times for solving the model.

The first preprocessing step focuses on the time windows of the requests. Often, either the pick-up time window or the delivery time window is specified, which is referred to as the critical time window. The corresponding pick up or delivery time window is often set to $[0, T_k]$, which is referred to as the non-critical time window. Using the ride time, service time and travel time, these broader time windows can be tightened. To adjust the time window when the pick-up vertex is the non-critical vertex, the new time window $e_{i,new}$ and $l_{i,new}$ is given by:

$$e_{i,new} = \max\{e_{i,old}, e_{i+n} - d_i - L\} \tag{3.32}$$

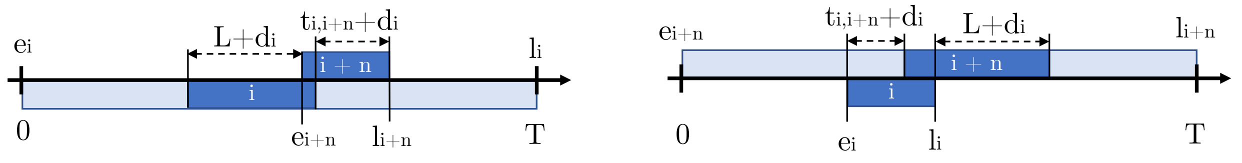
$$l_{i,new} = \min\{l_{i,old}, l_{i+n} - d_i - t_{i,i+n}\} \tag{3.33}$$

In 3.32 and 3.33, $e_{i,old} = 0$ and $l_{i,old} = T_k$. The start time window of the pick-up vertex e_i cannot be earlier than the start time window of the delivery vertex e_{i+n}, minus the maximal ride time L and the service time d_i at pick-up vertex i. The end time window of the pick-up vertex l_i cannot be later then the end time window of the delivery vertex l_{i+n}, minus the time it takes to get from the pick-up vertex to the delivery vertex $t_{i,i+n}$ minus the service time d_i at the pick-up vertex. When the delivery vertex is non-critical, the new time window $e_{i+n,new}$ and $l_{i+n,new}$ for delivery vertex $i+n$ can be obtained as follows:

$$e_{i+n,new} = \max\{e_{i+n,old}, e_i + d_i - t_{i,i+n}\} \tag{3.34}$$

$$l_{i+n,new} = \min\{l_{i+n,old}, l_i + d_i - L\} \tag{3.35}$$

In 3.34 and 3.35, $e_{i+n,old} = 0$ and $l_{i+n,old} = T_k$. The start time window of the delivery vertex e_{i+n} cannot be earlier than the start time window of the pick-up vertex e_i plus the service time at the pick-up vertex d_i and the travel time from the pick-up vertex to the delivery vertex $t_{i,i+n}$. The end time window of the delivery vertex l_{i+n} cannot be later than the end time window of the pick-up vertex l_i plus the service time at the pick-up vertex d_i plus the maximal ride time L. In Figure 3.3b, the tightening of the time windows is presented.



(a) Time window tightening of the pickup time window **(b)** Time window tightening of the drop off time window

Figure 3.3: Tightening of the pickup and drop off time windows

Next to the time windows of the request vertices, time windows for the meeting points can also be defined. The meeting points do not have a time window, because upfront it is not known which passenger is going to use which meeting point. However, for each meeting point, it can be defined which passengers from which vertices can reach this meeting point. The set $\mathcal{PP}_m$ consists of all pick up vertices that can reach meeting point m and the set $\mathcal{PD}_m$ consists of all delivery vertices that can reach meeting point

m. The set $\mathcal{E}_m = \{e_i + p_{i,m}, \forall i \in \mathcal{PP}_m\} + \{e_i, \forall i \in \mathcal{PD}_m\}$ contains all start time windows plus the walking time from the pick-up vertex to the meeting point, of all pick-up vertices that can reach meeting point m and all the start time windows of the delivery-vertices that can reach meeting point m. The set $\mathcal{L}_m = \{l_i, \forall i \in \mathcal{PP}_m\} + \{l_i - p_{m,i}, \forall i \in \mathcal{PD}_m\}$ contains all end time windows of all vertices that can reach meeting point m. The time windows for meeting point m are then defined as follows:

$$e_m = \min(\mathcal{E}_m) \qquad \forall m \in \mathcal{M} \tag{3.36}$$

$$l_m = \max(\mathcal{L}_m) \qquad \forall m \in \mathcal{M} \tag{3.37}$$

The meeting point can only be reached within the time frame of the earliest possible start of a time window and the latest possible end of a time window.

The tightened time windows and the time windows for the meeting points can be used in the next preprocessing step and the first valid inequality. The next preprocessing step is the removal of infeasible arcs. If there is an arc (i, j) that has a travel time $t_{i,j}$ that is too long to reach vertex j before its time window closes, the usage of this arc will always result in an infeasible solution. Therefore, this arc is removed from the arc set. This preprocessing step is shown in 3.38. Arcs for which this inequality holds are removed from the arc set $\mathcal{A}$.

$$e_i + d_i + t_{i,j} > l_j \qquad \forall (i,j) \in \mathcal{A} \tag{3.38}$$

The first valid inequality is adopted from Cordeau (2006) and slightly adjusted to also incorporate the meeting points. This valid inequality strengthens the bounds on the time variables and is defined as follows:

$$u_{i,k} \geq e_i + \sum_{j:(j,i)\in\mathcal{A}} \max(0, e_j - e_i + d_{min} + t_{ij}) x_{jik} \qquad \forall i \in P \cup D \cup \mathcal{M}, k \in K \tag{3.39}$$

$$u_{i,k} \leq l_i - \sum_{j:(i,j)\in\mathcal{A}} \max(0, l_i - l_j + d_{min} + t_{i,j}) x_{ijk} \qquad \forall i \in P \cup D \cup \mathcal{M}, k \in K \tag{3.40}$$

In 3.39 and 3.40, the tightened time windows and the time windows for the meeting points are used. The parameter d_{min} is the minimal time that is necessary as service time. If multiple requests are coupled to a meeting point, this service time becomes larger. If this is the case, the right hand side of 3.39 becomes larger, so then the inequality still holds. For 3.40 the right hand side would become smaller, and thus the inequality also still holds.

The second valid inequality removes solutions that use a sub-tour. Each vehicle traverses a number of arcs that is at least as large as the number of vertices present in this route, minus one. Two of the vertices in the route are always the depot and all vertices should be in one of the routes. The total number of arcs that may be traversed by all vehicles in a feasible solution, is at most twice the number of requests, plus the number of vehicles available. When a meeting point is used for multiple requests, the number of arcs traversed in a route can only become less. The valid inequality is displayed in 3.41.

$$\sum_{k\in K} \sum_{(i,j)\in\mathcal{A}} x_{ijk} \leq 2n + |K| \tag{3.41}$$

3.8 Demonstration in a toy network

To validate the DARPmp, a toy network is created to test the new formulation on, which is displayed in Figure 3.4. This toy instance has 8 trip requests, so $P = \{1, 2, ..., 8\}$ and $D = \{9, 10, ..., 16\}$. The set with additional vertices that can be used as meeting points is $\mathcal{M} = \{18, 19, ..., 31\}$. The depots are vertices 0 and 17. There are 2 vehicles available ($K = \{0, 1\}$) with capacity $Q_k = 3$ and maximum tour duration of $T_k = 300$. The maximal ride time for a passenger $L = 30$. For each pick up request, one person wants to enter the vehicle ($q_i = 1$ for $i \in P$) and at each drop off, one person wants to exit ($q_i = -1$ for $i \in D$). Each vertex has associated time windows. The service duration for each passenger equals 3, ($d_i = 3$ for $i \in V$). The minimal walking distance $C^{min} = 0.7$ and the maximal walking distance $C^{max} = 0.75$. The maximum total walking distance $W = 10$. All details of the input used in this toy network are presented in Appendix A.2. Two tests are performed on this toy network in order to compare the formulations. First, the original DARP is applied to this network. Next the DARPmp is applied to the toy network with the additional set of meeting points $\mathcal{M}$. .

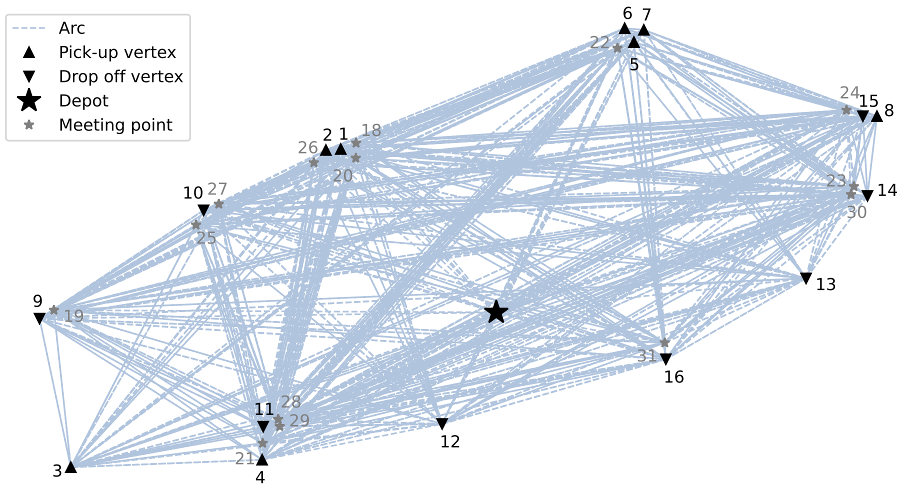


Figure 3.4: Toy network to validate the formulation

For applying the original DARP to the toy network, the meeting point vertices are not taken into consideration since this formulation does not contain meeting points. The formulation presented in Formulas 2.1 to 2.15 is used to solve the instance. The solution can be found in Figure 3.5. Two routes are generated, where the blue route represents vehicle 0 and the green route represents vehicle 1. The specifications of the route of vehicle 0 are presented in Table 3.1 and those of vehicle 1 are presented in Table 3.2. In these tables, the load at every vertex w_i^k and the time the vehicle arrives at each vertex u_i^k are given, together with the corresponding time windows. The total travel costs for this instance are 101.46.

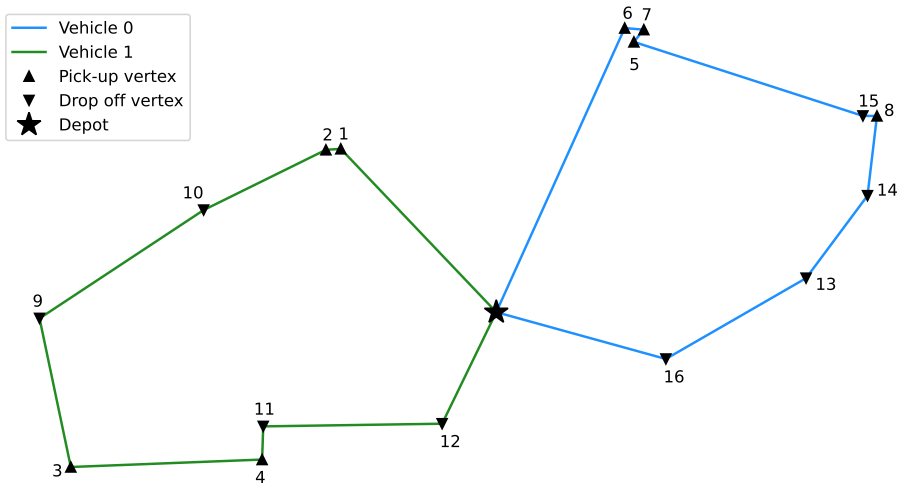


Figure 3.5: DARP solution for the toy network

Table 3.1: DARP: Route of vehicle 0 in toy network

i	d →	6 →	7 →	5 →	15 →	8 →	14 →	13 →	16 →	d
w_i^0	0	1	2	3	2	3	2	1	0	0
e_i	0	20	20	20	60	50	80	80	100	0
u_i^0	0	47	50.8	56.7	70	73.6	80	89.7	106.6	300
l_i	300	80	80	80	100	150	100	100	130	300

Table 3.2: DARP: Route of vehicle 1 in toy network

i	d →	1 →	2 →	10 →	9 →	3 →	4 →	11 →	12 →	d
w_i^1	0	1	2	1	0	1	2	1	0	0
e_i	0	50	50	100	100	70	80	120	150	0
u_i^1	0	78.4	91.2	100	111.4	120.9	132.1	136.5	165.1	300
l_i	300	150	130	130	150	160	180	160	180	300

The solution of the DARPmp using the toy network with the additional set of meeting points can be found in Figure 3.6. In this figure, the blue line shows the tour of vehicle 0 and the green line shows the tour of vehicle 1. The routes look similar to the routes for the original DARP. It can be observed that several meeting points are used, but not necessarily each of them. It shows several possibilities of meeting point usage. For example, a meeting point is used for the sole purpose of minimizing the total costs and not for two passengers to meet at, which occurs at meeting point 19, 25, 26, 20, 30 and 31. Furthermore, it shows an example of using a meeting point for a drop off and a pickup of two different passengers (vertex 21). It shows that it is also possible to use an original pick up location of a passenger as meeting point for multiple passengers, as can be seen at vertex 5 where the passengers of request 5, 6 and 7 get in the vehicle. It also shows that the original drop off vertex of a passenger can be used as a pick up location for the next passenger (vertex 15).

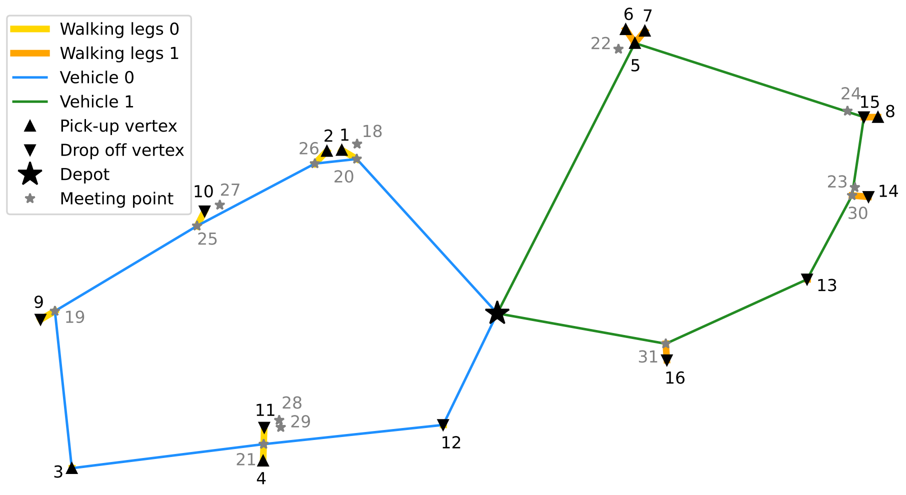


Figure 3.6: DARPmp solution for the toy network

Other important observations are that that each person is picked up before he or she is dropped of and that for both tours, the order of time is correct. Furthermore, the load of the vehicle is updated correctly and does not exceed the vehicle capacity of 3. Each tour does not exceed the maximal time of 300, and each passenger has a ride time that is lower than 30.

Table 3.3 shows the specifications of route 0 and Table 3.4 shows the specifications of route 1. In this table is presented to which requests the used meeting points belong, in the row that is indicated with the symbol i. Furthermore, the weight at each vertex in the route is given (w_m^k), as well as the time the vehicle arrives at this vertex (u_m^k). Next to that, the adjusted time windows for each vertex in the route are given, indicated by e_m as a lower bound and l_m as an upper bound. If multiple request vertices are coupled to a meeting point vertex, then the lower bound of the time window is given by the latest start time of the time window of the coupled vertices. For the upper bound the earliest end time of the coupled vertices is used. The total travel costs for this instance when applying the DARPmp are 96.14, which is lower than the costs for the DARP.

Table 3.3: DARPmp: Route of vehicle 0 in toy network

m	d $\rightarrow$	20 $\rightarrow$	26 $\rightarrow$	25 $\rightarrow$	19 $\rightarrow$	3 $\rightarrow$	21 $\rightarrow$	12 $\rightarrow$	d
i	0	1	2	10	9	3	$11, 4$	12	17
w_m^0	0	1	2	1	0	1	1	0	0
e_m	0	53	53	100	100	70	120	150	0
u_m^0	0	59.9	81.9	100.0	110.1	145.9	157.2	170.9	300
l_m	300	150	130	127.2	147.1	160	157.2	180	300

Table 3.4: DARPmp: Route of vehicle 1 in toy network

m	d $\rightarrow$	5 $\rightarrow$	15 $\rightarrow$	30 $\rightarrow$	13 $\rightarrow$	31 $\rightarrow$	d
i	0	$5, 6, 7$	$15, 8$	14	13	16	17
w_m^1	0	3	3	2	1	0	0
e_m	0	22.8	60	80	80	100	0
u_m^1	0	48.1	67.3	80	87.1	100	300
l_m	300	80	100	100	97.2	127.2	300

4 Meta-heuristic Algorithms

To solve the DARPmp for large-scale instances in short computation times, two meta-heuristic algorithms are proposed. For both algorithms, an initial solution is computed using a construction heuristic algorithm in the first step of the search. Next, this initial solution is improved using several local search operators. This chapter starts with an introduction about meta-heuristics that are used for vehicle routing problems. In the next section, the construction heuristic is described, followed by a section about the local search operators. Lastly, the solution evaluation procedure and the algorithmic structure of the two meta-heuristics are presented.

4.1 Tabu Search

For solving the DARPmp, a meta-heuristic Tabu Search approach is proposed. The Tabu Search method starts with an initial solution S^0 and moves at each iteration i to a new solution S'. This new solution S' is found by evaluating the neighbourhood $N(S)$ of the previous solution. The best solution in terms of the objective function value of this neighbourhood is taken as the new solution S. The neighborhood is constructed by using a local search operator, which generates solutions that contain characteristics of the current solution but are slightly different. A Tabu list is used to avoid cycling through solutions that are already known and getting stuck in a local optimum. In this list, already visited solutions are saved and the algorithm forbids to use these solutions for a certain number of iterations.

The Tabu Search algorithm was first proposed by Glover (1986) and was later applied to several forms of the VRP (Gendrau, 1994);(Cordeau et al., 1997);(Alonso et al., 2007). It has also been used in several studies about the DARP. Cordeau and Laporte (2003) for example, use a Tabu Search approach for the static multi-vehicle DARP. Kirchler and Calvo (2013) use a granular Tabu Search approach for the static DARP. An improved Tabu Search heuristic for the static DARP was proposed by Ho et al. (2018).

Other studies use different search algorithms that also evaluate the neighborhoods of intermediate solutions. Braekers et al. (2014) for example, use a Deterministic Annealing framework to solve the Multi-Depot Heterogeneous DARP, which uses a threshold value to determine if a solution is accepted as the new solution. The neighborhood of this new solution is evaluated, using a number of local search operators. A Genetic Algorithm is proposed by Masmoudi et al. (2017), which combines characteristics of known solutions to obtain better solutions. They also use local search operators to evaluate the neighborhood of these known solutions to obtain better solutions.

The proposed Tabu Search meta-heuristics contain elements of the method of Cordeau and Laporte (2003), Braekers et al. (2014) and Ho et al. (2018). In the following sections, the structure and the characteristics of the proposed methods are explained. In the first algorithm, searching for optimal meeting points is started after a solution for the DARP is found. This algorithm is referred to as TS1. In the second algorithm (TS2), searching for optimal meeting points is performed during the entire search.

4.2 Construction heuristic

As mentioned in the previous section, the Tabu Search starts with an initial solution to the DARPmp. Cordeau and Laporte (2003) use a randomly generated start solution, in which all constraints may be violated. Since the time it takes to find a first feasible solution depends on the quality of the initial solution (Ho et al., 2018), a construction heuristic that generates a solution of a higher quality is proposed. Since a solution to the DARP is also feasible for the DARPmp, meeting points will not be considered in generating a start solution. Meeting points will be added later using one of the local search operators that are explained in section 4.3. The proposed construction heuristic is similar to the heuristic used by Ho et al. (2018), but it does not use a random sequence of requests but it takes into account the direct travel distance of each request, as will be described in the remaining part of this section.

First, the requests are sorted in order of their direct travel distance, starting with the request with the largest travel distance. For each available vehicle, an empty route is constructed, containing two copies of the depot. Then for the first request in the list, it is checked in which positions in can be inserted. It is checked whether the solution is feasible regarding the time windows, ride time, route time and passenger load and the solution with the lowest cost increase is chosen. Then the next request in the list is taken and for the solution generated in the previous step it is again checked what is the best feasible insertion position is in terms of costs when inserting this request in all possible positions in all vehicles. These steps are repeated until all requests are assigned to a vehicle. If it is not possible to insert a request in any of the vehicles while the solution remains feasible, the algorithm is started over again and the request that could not be inserted is the first request to add to the routes. The other requests are added after this step, still in the order of the travel distance. The algorithmic structure is presented in Algorithm 1.

The initial solution for the second algorithm constructed in the same way. In the next step of the algorithm, the best meeting points for this solution are found and incorporated in the solution structure.

Algorithm 1 Construction heuristic

```
Set SortedList = sort requests on travel distance
Initialize $|K|$ routes with two copies of the depot
Assigned = False
while Assigned = False do
    for all requests in SortedList do
        for all vehicles do
            Try to insert the request in all feasible positions of the corresponding route
        end for
        Select the insertion with the lowest costs and update the solution
    end for
    if All requests are assigned then
        Assigned = True
    else
        SortedList is changed to start with the request that could not be assigned
    end if
end while
```

4.3 Local search operators

In the Tabu Search, five local search operators are used to generate a neighborhood for the current solution. The first four operators are adopted from Braekers et al. (2014) and can be applied to the original DARP formulation. The last operator is a meeting point operator that is created for the DARPmp formulation. The first four operators are called *relocate*, *exchange*, *2-opt* and *r-4-opt* and they will be further explained in the remainder of this section. A visual representation of how these operators work can be found in Appendix A.3. The last operator will be referred to as *meetingpoint*.

The first operator is the *relocate* operator. This operator removes a request from a randomly selected route and tries to insert it in the route of all other vehicles. It first tries to put the pick-up vertex in a feasible position and if a feasible position is found, it tries to insert the drop off vertex at a feasible location.

The second operator is the *exchange* operator. The *exchange* operator swaps a user request from one route with a user request of another route. It randomly selects a vehicle and sequentially looks at all the requests coupled to this vehicle. For these requests the best insertion position in all other routes found, by first trying to find a feasible insertion position for the pickup vertex and second a feasible insertion position for the drop off vertex. The request that is removed from one of the other vehicles is inserted in the route of the randomly selected vehicle, where the pickup and delivery vertex are inserted at the same position as they were for the request that is removed from this route.

The *2-opt* operator was first proposed by Potvin and Rousseau (1995) and is designed for problems with time windows. This operator takes two random routes as an input. Both routes are split in half by removing an arc where the vehicle is empty. In the next step, the operator tries to combine the first part of the first route with the second part of the second route and vice versa. It saves all the feasible combinations.

The *r-4-opt* operator removes four successive arcs in a randomly selected route and removes them. The remaining parts of the route are reconnected in all possible ways. The three successive vertices that are between these arcs get a different order using this operator. While changing this order, it is taken into account that the pick up vertex of a request can never be put after the drop off vertex of the same request in a route. All feasible orders are saved.

The last operator is the meeting point operator (*meetingpoint*), which replaces pickup or delivery vertices with meeting point vertices. For each vertex, it is determined which meeting points that are generated with the M-center method are close enough to walk to. For each vertex in the route, it is tried to replace the original pickup or delivery vertex with a meeting point. All feasible meeting point combinations are saved.

4.4 Solution evaluation

The aim of the Tabu Search framework is to minimize the cost function $f(s) = \sum\limits_{k \in K} \sum\limits_{(i,j) \in \mathcal{A}} c_{ij}^k x_{ij}^k$, which means the minimization of the travel costs. By applying the local search operators, different solutions with different travel costs are found. Next to evaluating the costs of a solution generated by a local search operator, the feasibility of the solutions must be evaluated. When applying local search operators to a solution and adjusting the order of vertices in certain routes, the feasibility of the passenger load, route duration, time window and ride time constraints may be affected. Evaluating the solution therefore requires to recompute the arrival time of the vehicles at each vertex, before the route duration, ride time and time window constraints can be evaluated. The arrival time at each vertex is calculated using the forward time slack method of Cordeau and Laporte (2003). This method uses the time windows and maximal ride time as an input and calculates if it is possible to delay the start of the service time at pickup vertices in order to still comply to the ride time and time window constraints. The procedure first minimizes the deviation from the time window constraints, then the deviation from the route duration constraints and next sequentially minimizes the devivation from ride time constraints. Another function checks whether the passenger load is exceeded at any point of the vehicles route, by keeping track of the number of passengers in the vehicle at any vertex. The construction heuristic and local search operators never create routes where the pickup vertex is inserted after the corresponding delivery vertex or routes where the pickup vertex and delivery vertex of the same request are not in the same vehicle. This ensures that these constraints will never be violated. The procedure requires a large sequence of computations. Therefore, this procedure is only used after it is first assured that the load constraint is not violated.

For the meeting point operator, it is first checked if the meeting point is close enough before it is inserted. Therefore the maximum walking distance constraints 3.21 are never violated. Next it is checked that if the total walking distance is not exceeded. For evaluating the feasibility regarding the time windows, ride time and route duration, the same time slack procedure is used as mentioned before, but the time windows are adjusted before hand to comply to constraints 3.16 and 3.17. The same method is used to check if the passenger load is exceeded at any point during the route.

4.5 Algorithm structure

In this section, the proposed structure for both Tabu Search algorithms are described. The first algorithm starts with an initial solution S^0, created with the construction heuristic. The current solution S and the best found solution S^{best} is set to this initial solution. An empty list is created to save solutions that are already visited. These solutions are forbidden to evaluate. When the list reaches a maximum number of items, the solution that was first added to the list is deleted from the Tabu List.

A randomly chosen local search operator of the four operators *relocate*, *exchange*, *2-opt* or *r-4-opt* is applied to the solution S. For each solution, first the feasibility regarding the passenger load is checked, since this does not require much computation time. If the solution is feasible regarding the passenger load, the feasibility regarding the time windows, ride time and route time is checked.

If a feasible solution S' is found which is not already in the Tabu List, the solution with the lowest cost is chosen as the new solution S. Next, this solution is added to the Tabu List. If the costs of the new solution are lower than the cost of the current best solution S^{best}, then S^{best} will become S'. If no feasible solution is found when applying the randomly chosen operator, the current solution S will remain the same. Then, this procedure is repeated in order to go through the search space. In order to save time, the algorithm keeps track of which operator in combination with which vehicle or vehicles is already performed on the current solution S and the best solution S^{best}. If no feasible neighboring solution is found using a certain combination of operator and vehicle, this combination is not applied again to the same solution, since it is already known that it will not produce feasible neighboring solutions. If no better solution is found for a number of ϕ iterations, S will get the current best solution S^{best} again, to make sure the algorithm does not keep looking in a direction where no better solutions are found. If the algorithm has returned to the same solution S^{best} for κ times and has not found a better feasible solution, the algorithm is stopped. For the first algorithm, when the algorithm stops, a solution for the DARP has been found. To this solution, the meeting point operator is applied, which inserts meeting points that creates a route with lowest possible cost for this solution. The pseudocode of the first Tabu Search Algorithm can be found in 2.

The structure of the second algorithm is almost the same as the structure of the first algorithm. It also starts with the initial solution generated with the construction heuristic and has a Tabu List in which the solutions that are already visited are saved. The same local search operators are applied to the current S in a random order. The difference is, that in each iteration, for each neighborhood generated using the local search operators, it is checked if from this solution a feasible solution using meeting points can be generated. The solution with the lowest costs for the meeting point route is selected. If no better solution is found for a number of ϕ iterations, S will get the current best solution S^{best} again. If the algorithm has returned to the same solution S^{best} for κ times and has not found a better feasible solution in the meantime, the algorithm is stopped. No extra operations are performed on this final solution. The algorithmic structure of the second Tabu Search Algorithm 3

Algorithm 2 Pseudocode for TS1

```
solution $S^{best} \leftarrow S^0, S \leftarrow S^0$
initialize $counter_i \leftarrow 0$, $counter_s \leftarrow 0$, $\kappa$, $\phi$, Tabu List
while $counter_i < \kappa$ do
    select a random local search operator $o \in \{relocate, exchange, 2\text{-}opt, r\text{-}4\text{-}opt\}$
    try to identify the solution with the lowest cost that is feasible and not in the Tabu List
    if a solution $S'$ is found then
        $S \leftarrow S'$
        if $f(S') < f(S^{best})$ then
            $S^{best} \leftarrow S'$, $counter_s \leftarrow 0$, $counter_i \leftarrow 0$
        end if
    end if
    $counter_s$ += 1
    if $counter_s > \phi$ then
        $S \leftarrow S^{best}$, $counter_i \leftarrow counter_i + 1$
    end if
end while
for all possible meeting points do
    apply meeting point operator to $S^{best}$ and find best solution with meeting points $S^{best}_{mp}$
end for
return $S^{best}_{mp}$
```

Algorithm 3 Pseudocode for TS2

```
solution $S^{best} \leftarrow S_0, S \leftarrow S^0$
initialize $counter_i \leftarrow 0$, $counter_s \leftarrow 0$, $\kappa$, $\phi$ Tabu List
while $counter_i < \kappa$ do
    select a random local search operator $o \in \{relocate, exchange, 2\text{-}opt, r\text{-}4\text{-}opt\}$
    apply meeting point operator to all solutions in neighborhood
    try to identify the solution with the lowest cost that is feasible and not in the Tabu List
    if a solution $S'$ is found then
        $S \leftarrow S'$
        if $f(S') < f(S^{best})$ then
            $S^{best} \leftarrow S'$, $counter_s \leftarrow 0$, $counter_i \leftarrow 0$
        end if
    end if
    $counter_s \leftarrow counter_s + 1$
    if $counter_s > \phi$ then
        $S \leftarrow S^{best}$, $counter_i \leftarrow counter_i + 1$
    end if
end while
return $S^{best}$
```

5 Numerical Experiments

In this chapter, the results the numerical experiments using both the Branch-and-cut algorithm and the two versions of the Tabu Search are presented. Three types of experiments are performed that all have a different aim. In the first experiment, the aim is to give an insight in the influence of the valid inequalities and the preprocessing steps on the computational performance of the Branch-and-cut algorithm. In the second experiment, the Branch-and-cut algorithm and both Tabu Search algorithms are applied to a set of benchmark instances, with the aim to give insight in the computational performance of all algorithms. With the last experiment, insight is given in the impact of the model on the travel time of the passengers.

In the first section of this chapter, details about the data and the used resources are described. In the following sections, the experiments are presented in the same order as they are presented in this introduction. Each section starts with a more detailed description of the experiment and its experimental design, followed by the results and an interpretation of the results.

5.1 Experimental design

To evaluate the Branch-and-cut algorithm and the two versions of the Tabu Search, various experiments are performed on benchmark instances that are also used in other research. The original instances are retrieved from `http://neumann.hec.ca/chairedistributique/data/darp/branch-and-cut/`. The instances vary from 2 to 4 vehicles and from 16 to 48 trip requests. They all have a constant number of 1 passenger per trip request. In order to use these benchmark instances they first have to be adjusted to contain meeting points, using the m-center method. The adjusted benchmark instances can be found at `https://data.mendeley.com/datasets/h5392z6csr/1`. For all of these instances, $C^{min} = 0,52$ and $C^{max} = 0,53$.

For all experiments, a computer with a AMD Ryzen 7 PRO 5850U processor (1.90 GHz) and 16GB of RAM is used. The Branch-and-cut experiments are performed using Gurobi version 9.5.2 in Python 3.9. The Tabu Search algorithm is also implemented in Python. For this model several parameters are used, which are described in section 4.5. For the first algorithm, the number of iterations ϕ after which S returns back to S^{best} is set to 50. The number of iterations after which no better solution is found and the algorithm is terminated is set to 250. This means that κ is set to 5. For the second algorithm, ϕ set to 10, because when observing the model it showed that the model did not often find a better solution when it was not within 10 iterations. For this algorithm, κ is set to 10. This means that the algorithm is terminated if no better solution was found for 100 iterations.

5.2 The effect of preprocessing steps and valid inequalities

The first experiment focuses the preprocessing steps and valid inequalities described in section 3.7 and their effect on the computational performance of the Branch-and-cut algorithm. The first preprocessing step is the time window tightening (P1), the second preprocessing step is the removal of infeasible arcs (P2), the first valid inequality is tightening the bounds on the service time variable (V1) and the second is the sub-tour elimination constraint (V2). In order to evaluate the effects, the smallest benchmark instance with 2 vehicles and 16 trip requests is used. From this instance, 17 instances are created with the number of meeting points varying from 0 to 16 meeting points. For the analysis, 10 different experiments are performed. P2 and V1 are always tested in combination with P1, because P2 and V1 use the time windows as an input, and P1 already tightens those time windows. Furthermore, every combination of preprocessing steps and valid inequalities is tested. The model is run for a maximum of 3600 seconds. If the optimal solution is not found after 3600 seconds, the model is cut off. The expectation is that each preprocessing step and each valid inequality has an influence on the computational time and that all four combined results in the lowest computational times.

Table 5.1 shows the computational results of the 10 different combinations of preprocessing steps and valid inequalities. When trying to solve the model with no preprocessing steps and valid inequalities applied, the run time takes over 3600 seconds for every instance. This also holds for models where only P1 or P1 and P2 are applied. For all other combinations of preprocessing steps and valid inequalities that are tested, the optimal solution is found for all of the instances within 3600 seconds. The run time increases as the number of meeting points increases, with here and there some exceptions.

Table 5.1: Computational performance in seconds (s) for different sets of preprocessing steps (P) and valid inequalities (V), on instances with different numbers of meeting points (MP)

MP	Without	P1	P1&P2	V2	P1&V2	P1&V1	P1,P2&V1	P1,P2&V2	P1,V1&V2	P1,P2,V1&V2
0	-	-	-	23,89	39,22	5,79	5,60	31,12	5,23	5,22
1	-	-	-	83,85	45,44	6,61	5,91	27,04	6,12	5,79
2	-	-	-	48,01	35,25	6,77	6,51	26,84	6,04	5,93
3	-	-	-	137,29	55,34	7,02	7,21	32,25	7,02	6,82
4	-	-	-	120,57	31,33	7,66	7,17	31,01	6,62	6,70
5	-	-	-	76,63	51,68	8,25	8,98	40,98	7,34	6,85
6	-	-	-	246,40	53,57	13,74	14,66	42,80	7,73	9,18
7	-	-	-	290,44	170,12	23,98	15,22	44,96	8,46	8,33
8	-	-	-	391,29	69,18	17,46	19,64	46,87	10,21	9,33
9	-	-	-	328,68	165,81	22,44	22,69	64,33	10,51	10,21
10	-	-	-	275,84	64,48	65,99	71,18	49,75	11,81	9,59
11	-	-	-	750,42	96,87	52,72	102,51	70,95	11,13	10,93
12	-	-	-	2268,32	104,96	143,92	63,96	59,87	12,66	13,25
13	-	-	-	673,99	219,45	92,30	81,31	68,01	11,67	11,26
14	-	-	-	590,29	106,49	136,17	95,58	65,44	12,79	12,60
15	-	-	-	820,91	189,87	161,69	134,37	73,01	15,29	13,13
16	-	-	-	1375,30	140,76	98,80	118,89	80,94	19,74	14,92

In Figure 5.1, the run times from table 5.1 are plotted in a column chart. For readability purposes of the last experiments, the ax on the left side of the graph is cut off at 400 seconds. The run time for the experiment in which only V2 is applied, exceeds 400 seconds for 11, 12, 13, 14, 15 and 16 meeting points. Next to that, the experiments for which no optimal solution was found within 3600 seconds are not displayed. A more detailed description of this figure is given in Appendix A.4.

Looking at all preprocessing steps and valid inequalities and their influence on the run time individually, it is observed that P1 on its own does not enough to keep the run time within 3600. However, in combination with other preprocessing steps and valid inequalities it does have an effect. When comparing the run time for the experiment with V2 only and the experiment with P1 and V2, the run times drop significantly when incorporating P1 too. It does seem plausible that this valid inequality does not have a large effect on its own, because it only affects the wider time windows of the vertices, which is in this case half of the pickup vertices and half of the delivery vertices.

For P2, the impact on the run time is less noticeable. Looking at the experiment with P1 and V1 and the experiment with P1, P2 and V1, the differences in run time are minor. On the other hand, looking at the difference between the experiment with P1 and V2 and the experiment with P1, P2 and V2, P2 seems to have some more impact. The run time especially decreases for the instances with more meeting points. When comparing the experiment where P1, V1 and V2 are used to the experiment with all valid inequalities, the

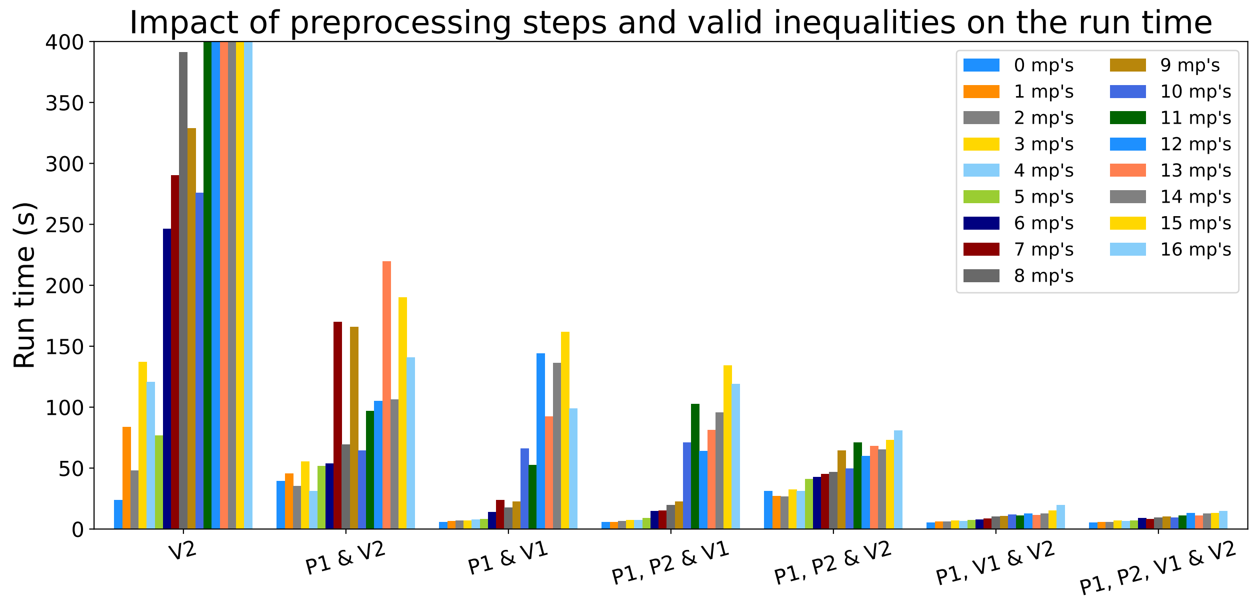


Figure 5.1: Computational results for the DARPmp for instance a2-16 with different sets of preprocessing steps (P) and valid inequalities (V) and different numbers of meeting points

results look similar. However, most of the run times are slightly lower when P2 is used as well. It seems that P2 mainly has an effect on the instances with a larger number of meeting points. A larger number of meeting points means that there are more arcs, so also possibly more arcs that are infeasible and can be removed, which can explain why this step mainly affects instances with more meeting points. The observation that the preprocessing step itself does not have a large effect on the run time, might be explained by not having many arcs that are infeasible due to the time windows.

V1 seems to have a large effect on the run time. When comparing the experiment where only P1 and V2 are used, to the experiment where P1, V1 and V2 are used, it can be observed that the run time decreases significantly for all instances. When comparing the experiment with P1, P2 and V2 to the experiments where all valid inequalities are used it is again visible that the run time decreases significantly when V1 is used. This effect is especially visible for the instances with a larger number of meeting points. The large impact on the run time can be explained by the inequality being able to cut off a larger number of infeasible solutions than P1 and P2. This valid inequality is bounding the service time of all vertices, which has more potential than only removing infeasible arcs for example.

Lastly, looking into the effect of V2, applying only this valid inequality already results in solving all instances within 3600 seconds. When comparing the experiment with P1 and V1 tot the experiment with P1, V1 and V2, it can be observed that V2 has the largest effect on the run time for the instances with a larger number of meeting points. The same holds when comparing the experiment with P1, P2 and V1 and the experiment with all valid inequalities. The valid inequality puts a bound on the number of x-variables, which means that it puts a bound on the number of arcs that can be traversed. The number of arcs is higher in the instances with more meeting points, which can be an explanation for the observation that V2 mainly has an effect on the instances with more meeting points.

Overall, a pattern of higher run times for a higher number of meeting points is observed. However, there some instances where this pattern does not hold. An example is for the experiment where P1 and V2 are applied and the instance with 7 meeting points has a run time of 170,12 seconds and the instance with 8 meeting points has a run time of 69,18 seconds. These fluctuations in run times can be explained. When adding a meeting point to the graph, the input sets of the problem become larger and the number of variables increases. The expectation is that the run time is larger in this case. However, adding something to the input model can sometimes also alter the initial conditions for the Branch-and-bound algorithm, resulting in a more favourable starting position and finding an optimal solution faster.

The findings are in line with the expectations that are mentioned in the first part of this section. Each preprocessing step or valid inequality has an influence on the run time and the run times are lowest when all preprocessing steps and valid inequalities are used. Overall, the run time increases as the number of meeting points increases, with the exceptions that are discussed in the previous part of this section. All preprocessing steps or valid inequalities that are applied in this work are also used in other research regarding vehicle routing problems and specifically in research regarding the DARP (Cordeau, 2006), (Masmoudi et al., 2018), (Bongiovanni et al., 2019). The observation that using these preprocessing steps and valid inequalities result in lower run times for the Branch-and-cut algorithm of the DARPmp is therefore a logic result.

5.3 Computational results for large-scale instances

In the second experiment, the computational performance of the Branch-and-cut algorithm without preprocessing steps and valid inequalities (B&C), the Branch-and-cut algorithm with preprocessing steps and valid inequalities (B&C PV) and both the Tabu Search algorithms (TS1 and TS2) are tested on larger instances. For this comparison, all aforementioned instances with 2 to 4 vehicles and 16 to 48 trip requests are used and adjusted such that the ratio trip requests to meeting points in the instances is 1:1.5. The Branch-and-cut algorithms are run one time per benchmark instance and both Tabu Search algorithms are run five times per benchmark instance to obtain the results. The Tabu Search algorithms are run five times because the outcome for both the objective function value and the run time can differ significantly due to the randomness of the model. The expectation is that the run times of B&C will be higher than the run times of B&C PV. Next to that, it is expected that B&C PV will be faster for smaller instances than TS1 and TS2, but that it will become slower than TS1 and TS2 when the benchmark instances get more vehicles and more trip requests. Furthermore, it is expected that TS1 produces solutions faster than TS2 but that the resulting routes have higher costs. For TS2, it is expected that it produces solutions slower than TS1 due to the application of the meeting point operator in every iteration, but that the resulting routes have lower costs.

In Table 5.2, the computational results of B&C, B&C PV and TS1 and TS2 are presented. In the first column, the instances are presented in a format that represents the number of vehicles, the number of trip request and the number of meeting points. For example the instance in the first row, has 2 vehicles, 16 trip requests and 24 meeting points. Furthermore, the table presents for both Branch-and-cut versions the objective function value (OFV) of the optimal solution and the run time in seconds. For both Tabu Search algorithms, the best found objective function value (OFV), the gap with this best found objective function value and the objective function value found with the Branch-and-cut algorithm (Gap), the average objective function value of all runs (AVG) and the average run time in seconds is presented.

It can be observed that none of the instances can be solved without using the preprocessing steps and valid inequalities in the Branch-and-cut algorithm. When applying all preprocessing steps and valid inequalities, the first five instances can be solved. The run time increases as the instance gets more trip requests and more vehicles, which is something that is also observed in other research (Cordeau, 2006) (Bongiovanni et al., 2019). Both Tabu Search algorithms were able to find a feasible solution for all the instances. The run times for both Tabu Search algorithms seem to increase as the number of trip requests increases. They do not seem to increase as the number of vehicles increases. The run time for a4-16-24 is for example lower than the run time for a2-16-24. However, this pattern is also observed for other meta-heuristics using variants of

Table 5.2: Computational results of the Branch-and-cut algorithm without valid inequalities, the Branch-and-cut algorithm with valid inequalities, Tabu Search algorithm 1 and Tabu Search algorithm 2

	B&C		B&C PV		TS1				TS2			
Instance	**OFV**	**Time (s)**	**OFV**	**Time (s)**	**OFV**	**Gap**	**AVG**	**Time (s)**	**OFV**	**Gap**	**AVG**	**Time (s)**
a2-16-24	-	-	285,53	30,77	286,57	0,361%	286,57	31,71	286,57	0,361%	288,41	160,66
a2-20-30	-	-	337,62	74,53	337,62	0,000%	337,62	139,09	337,62	0,000%	337,62	1050,51
a2-24-36	-	-	423,15	134,65	423,53	0,090%	425,29	157,14	423,20	0,011%	424,88	918,83
a3-18-27	-	-	295,20	2414,48	295,21	0,003%	299,53	22,42	301,81	2,240%	302,70	77,40
a3-24-36	-	-	336,57	14036,27	342,12	1,650%	342,99	154,63	341,13	1,354%	343,83	692,67
a3-30-45	-	-	-	-	486,13	-	486,94	210,40	486,42	-	491,90	1112,18
a3-36-54	-	-	-	-	583,38	-	584,52	328,55	556,94	-	559,73	3226,27
a4-16-24	-	-	-	-	276,32	-	279,44	14,07	276,32	-	279,46	38,20
a4-24-36	-	-	-	-	369,18	-	378,66	66,13	375,61	-	379,83	251,16
a4-32-48	-	-	-	-	478,02	-	486,93	138,21	475,46	-	494,11	740,74
a4-40-60	-	-	-	-	561,28	-	565,22	421,38	562,30	-	568,96	2239,43
a4-48-72	-	-	-	-	675,10	-	685,48	850,51	676,42	-	680,51	9051,69

the same instances (Braekers et al., 2014) (Malheiros et al., 2021).

When comparing B&C PV with TS1, it can be observed that B&C PV produces slightly better results in terms of the objective function value. Only for the instance a2-20-30, the same solution is found. For the other instances, TS1 produces results with a gap ranging from 0,003% to 1,650%. TS1 only visits solutions that are feasible for the DARP, to which in the last step the meeting point operator is applied. For the instances with a gap larger than 0%, the solution that is found by B&C PV is infeasible for the DARP, if the incorporated meeting points are translated back to their original pickup and delivery vertices. Therefore, the optimal solution is not found using TS1 and a gap between the result using B&C PV and TS1 remains. Regarding the run time, B&C PV produces results faster than TS1 for the smaller instances. For the larger instances TS1 is faster. The run time of TS1 increases as the number of trip requests increases. These findings are in line with the expectations mentioned in the beginning of this section.

When comparing B&C PV with TS2, B&C PV produces better results regarding the objective function value. Only for the instance a2-20-30 the same objective function value is found. TS2 algorithm produces results for the other instances with a gap between 0.011% and 2,240%. TS2 is able to visit solutions that are infeasible for the DARP, resulting in a lower gap for instances a2-24-36 and a3-24-36 compared to TS1. However, TS2 does still not find the optimal solution. It might be that the operators used are not the best for this particular problem, or that not enough iterations are performed find the optimal solution. Looking at the run time, B&C PV is faster for the first three instances, but TS2 is faster for the two instances with 3 vehicles. For most of the instances the run time increases as the number of trip requests increases, except for the instance a2-20-30, which has a higher average run time than the instance a2-24-36. These findings are in line with the expectations set in the first part of this section.

When comparing TS1 and TS2 regarding the best found objective function value, the results look similar. For 3 instances, the same solution was found as the best solution. For 5 instances, TS1 finds a better solution and for 4 instances TS2 finds a better solution. For most instances, the difference is not larger than 7. The biggest difference in run time is for the instance a3-36-54 where the difference is 26,44 and TS2 performs better. Looking at the average values, there is one instance for which algorithms have the same average, 8 instances for which the average of TS1 is lower and 3 instances for which the average of TS2 is lower. Comparing the two algorithms in terms of run time, TS1 is on average 5 times faster for each instance than TS2. These findings are not completely in line with the expectations set in the first part of this section. It was expected that the run times for TS2 would be higher than the run times for TS1, but that TS2 would produce better results than TS1. There are several reasons that might explain why TS2 does not find better results than TS1. The first reason could be that TS2 is cut off when no better solution is found for 100 iterations, while TS1 is cut off when no better solution has been found for 250 iterations. However, if the stopping criterion for TS2 was also set to 250 iterations, the run times of TS2 would increase even more. The second explanation could be that the solution space is searched differently in TS2. It does not follow the pattern of feasible DARP solutions but of feasible DARPmp solutions. It could be that once a solution that is infeasible for the DARP is found, it is difficult to get a better feasible solution in the next steps.

5.4 Evaluation framework

Lastly, the DARP and the DARPmp formulation are compared using some performance indicators. The aim of the first experiment is to show the gain of using meeting points for total routing costs. To do so, the travel costs when using the DARP and when using the DARPmp are calculated using B&C VP, TS1 and TS2. The expectation is that the travel costs are lower for the DARPmp than for the DARP.

The aim of the second experiment is to give insight in the effect of using meeting points on the travel time of the passengers. For this experiment, B&C VP, TS1 and TS2 are used as well as a model to solve the DARP. For each passenger their direct travel time and their actual travel time is calculated. For B&C VP of the DARPmp, the results of the 5 benchmark instances for which an optimal solution is found are used. For the results of the DARP that are used to compare the DARPmp to, the model in equations 2.1 to 2.13 is used. The model found the optimal solution for the first 9 instances within 4 hours. For both Tabu Search algorithms, the average of the 5 runs performend for these 9 benchmark instances are used, in order to be able to compare them to the DARP results. The expectation is that mostly short trips have a larger travel time than the direct travel time, since they are more easy to fit in to the schedule. This effect arises from the fixed maximal ride time that is used as a constraint in both the DARP and the DARPmp.

In Figure 5.2, the objective function values of the DARPmp using B&C VP, TS1 and TS2 are plotted as a percentage of the objective function value found solving the same instance with the DARP to global optimality. The black line indicates the objective function value for the DARP. If a bar reaches above the black line, it means that the solution method found a solution with higher costs than the DARP.

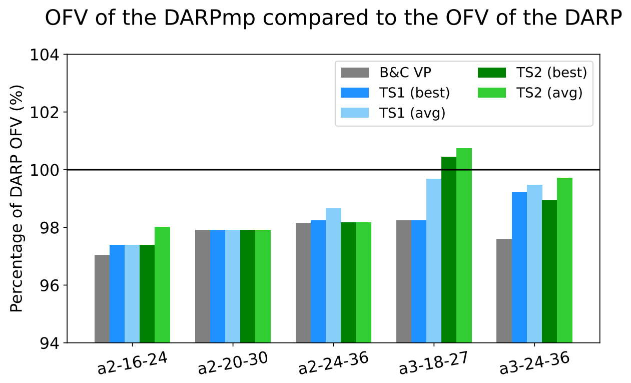
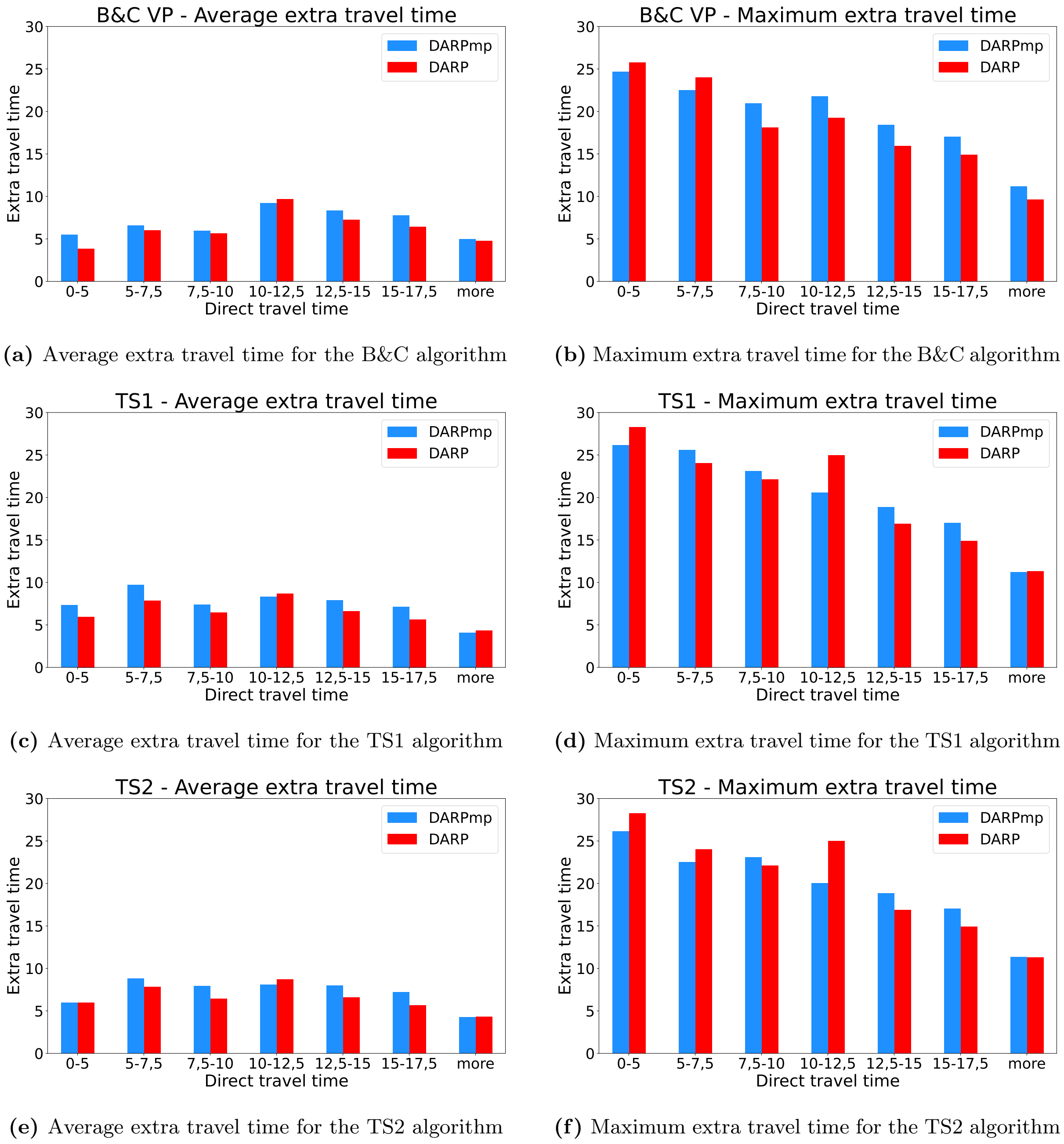


Figure 5.2: Comparison of the objective function value of the DARPmp as a percentage of the objective function value of the DARP

Looking at the results, it can be observed that the routing costs for the DARPmp using B&C VP, TS1 and TS2 are lower for each instance compared to the costs when using the DARP, except for TS2 on instance a3-18-27. For the B&C VP, the improvement lies around 2.5%. However, this percentage is highly dependent on the number of meeting points used in the network and the maximum walking distance set as a parameter.

The findings are mostly in line with the expectations set in the beginning of this chapter. The only exception is that for one instance, TS2 did not find a set of routes with lower travel costs than the DARP. However, when solving the DARPmp to global optimality, a set of routes with lower travel costs is always found.

The results of the second experiment are presented in 6 sub-figures in Figure 5.3. The figures in the left column show the average extra time and the figures in the right column show the maximum extra travel time compared to the direct distance. The blue colour represents the travel time for the passengers in the DARPmp model, which is the in-vehicle time plus the time they spent walking. The red colour represents the total travel time for the DARP model. Since no walking is involved in this model, the total travel time for the DARP is the same as the ride time. To visualise the results, the trips are grouped in bins on their direct travel time, with ranges 0-5, 5-7,5, 7,5-10, 10-12,5, 12,5-15, 15-17,5 and more than 17,5. This way, an insight can be created in the relation between the direct travel time and the extra travel time.



(a) Average extra travel time for the B&C algorithm

(b) Maximum extra travel time for the B&C algorithm

(c) Average extra travel time for the TS1 algorithm

(d) Maximum extra travel time for the TS1 algorithm

(e) Average extra travel time for the TS2 algorithm

(f) Maximum extra travel time for the TS2 algorithm

Figure 5.3: The effect of the different solution methods on the travel time for passengers

Looking at the results in Figure 5.3, the average extra travel time first seems to increase as the direct travel time increases. When the direct travel time exceeds 12,5, the run time seems do decrease again. The maximum extra travel time decreases as the direct travel time increases. There is no clear pattern visible for which of the two models results in a lower extra travel time. For some instances, the DARP performs better and for other instances the DARPmp performs better. A more detailed description of these sub-figures can be found in Appendix A.4.

Since the results of the Branch-and-cut algorithm are based on only 5 test instances and the results of the Tabu Search algorithms are based on 9 instances, they are difficult to compare. As can be observed in the figures, the observed extra travel times and maximum extra travel times for the DARP look different in Figures 5.3c, 5.3d, 5.3e and 5.3f compared to Figures 5.3a and 5.3b. However, the figures for the two Tabu Search algorithms can be compared. At first glance, the two figures look similar. The only differences can be found in bin 5-7,5, where the average extra travel time for the DARPmp is lower for TS2. This also holds for the maximum travel time.

The findings are not completely in line with the expectations set at the beginning of this section. On average, the extra travel time is actually higher for middle long trips. This observation can be explained. For short trips, the origin and destination are close to each other. This means that visiting the destination right after the origin can often be easily done in one go, because there is a small chance that there is another vertex for which it is profitable to visit in between these vertices. For larger trips, the origin and destination are further away, so there are more vertices on the way that could be visited. However, because of the ride time limit, there is not much room to extend the trip a lot. For middle long trips, the origin and destination are farther away from each other so there is again a higher chance of finding other vertices to visit in between, but there is more time available to deviate from the direct trip length. Therefore, the middle long trips have on average a larger extra travel time. When looking at the maximal extra travel time found, the short trips do have the longest extra travel time found, since there is room to have such an extension of the travel time. For trips larger than 17,5, there is only room to extend the trip with 12,5 at most.

The results do not show a consistent pattern regarding the extra travel time. For some direct trip lengths, the DARPmp has a lower average travel time than the DARP, and for other direct trip lengths the DARP has a lower average travel time than the DARPmp. Therefore, it is hard to state which model is better for the passenger point of view. The exact model shows however that there are always benefits for the total routing costs when using meeting points. The exact model always finds a solution that is at least as good as the solution for the DARP, since a solution for the DARP is always a feasible solution for the DARPmp. When developing both Tabu Search algorithms further such that they produce better results and converge to the optimal solution, this approach could give benefits in terms of the total routing costs for even larger input sizes. Looking at other works that use Tabu Search algorithms for the DARP, better objective function values are found in lower computation times, which proves that there is potential for the proposed Tabu Search algorithms to find better solutions in faster run times (Malheiros et al., 2021), (Braekers et al., 2014). This especially holds for TS1, which uses a DARP solution to apply meeting points to. It is also good to mention that these results are highly dependent on the choice of the location of the meeting points and the characteristics of the networks that are used to obtain these results. A more elaborate discussion about this is given in Chapter 6.

6 Discussion

In this chapter, the main findings of the research are discussed in a wider societal context. In the first section, the effects when applying the model in a real world setting are discussed. Situations in which the model would be most beneficial is also discussed in this section. In the following section, the effect of the objective function on the results of the model are discussed. In the last section, a discussion is provided about the disadvantages of using meeting points.

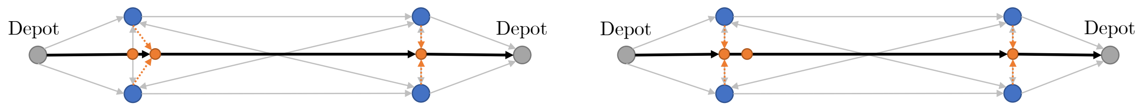
6.1 Application in a real city

For this research, benchmark instances are used to test the models on. The advantage of these instances is that they are also used in other research. This way, the results regarding the objective function value and computational time can be compared. However, this does not exactly represent how the models would work in a real city. First of all, the selection of meeting points would work differently in the real world than what is now done in the benchmark instances. The selection of meeting points that is used in the benchmark instances is based on a minimum and maximum walking distance. Using the m-center method, a number of meeting points is generated. Next, the desired number of meeting points is selected and used, such that the formulations can be tested. In a real city, these feasible meeting points would be determined in another way. The minimum and maximum walking distance would still hold, but the points within this range can be selected in another way. The feasibility of the points could also be determined looking at if it is possible for a vehicle to stop and if it is possible for a passenger to wait at this location. If an operator wants to use meeting points in its service, it should consider how to select these feasible meeting points.

Furthermore, the effect of the meeting points on the travel time can be different in a real city. This effect is now tested on meeting points that only lie on the arcs of the artificial network. The distance over these arcs from one pickup or drop off vertex to the other pickup or drop off vertex is the euclidean distance. In a city, the distance between two points is measured over the network. Such a net work can also have one-way streets, or streets that are often busy and where congestion occurs. In these kind of situations, meeting points could especially be helpful, since by using them the vehicles can avoid these one-way or busy streets. These effects are difficult to make visible in artificial benchmark instances.

6.2 Objective function

For the current model, an objective function is used that minimizes the travel costs, which is equivalent to the most common objective for DARP variations, which is minimizing the total distance traveled (Molenbruch et al., 2017). This objective function mainly focuses on the operator of the fleet of vehicles and does not focus on service quality. The service quality is only incorporated in the maximum ride time, time windows and maximum walking time constraints. This raises the question what implications this objective function can have for the passengers. One of these implications is illustrated using a simple example displayed in Figure 6.1. If there are two meeting points for a request that are feasible regarding the maximal walking distance, which both lie on the shortest path for the vehicle, there is no guarantee that the model chooses the meeting point that is closest by for the passenger. It can still choose both meeting points, because they do not result in different costs for the operator. This can result in the model generating the route that is displayed in Figure 6.1a, while the route in Figure 6.1b is better for the passenger and the same for the operator. When incorporating the user costs in the objective function, can result in selecting the meeting point that is best for the passenger.



(a) Only minimizing the operators costs **(b)** Using the best meeting point for the passenger

Figure 6.1: Implications when only using the operators costs in the objective function

Other researchers did incorporate the service quality in their objective function. Molenbruch et al. (2017) describe several studies that use multiple objectives to find an optimal route and found that many authors that do consider multiple objectives, also incorporate service quality related objectives. These extra objectives range from minimizing the users ride times to minimizing the vehicle idle times. Also incorporating the service quality for the users in the objective function can be beneficial for the users of the system, however one should think carefully about how to incorporate this part. When the focus is too much on the service quality and too much weight is put on the walking distance of the passenger, it can also occur that the meeting points are never used.

In another research, Militão and Tirachini (2021) designed a system to determine the optimal fleet size for a door-to-door shared demand-responsive transport system. In their objective function, they incorporated both operator and users costs, where the operator costs consists for example of the operation of each vehicle per hour, the variable costs associated with the number of kilometres like maintenance costs, the number of vehicles and the total distance travelled. The user costs consist of time savings and total waiting and in-vehicle time. The taking into account user costs did result in having to reject passengers that imposed long detours. However, they suggest that using meeting points can maybe be used to deal with the high costs of servicing passengers that induce long detours.

6.3 Pitfalls of meeting points

Another concern with the meeting points is if this results in an equitable situation. For passengers that are disabled, walking a small distance would probably not be possible. Next to that, the results of the evaluation framework also show that passengers with a small trip length face the possibility to get a large detour before they reach their destination. Furthermore, passengers with a small trip also have a risk of having to walk longer than they are actually in the vehicle. Another concern when using meeting points could be the feeling of safety for the passengers. Being inside a vehicle with a stranger might not be the biggest problem if there is a driver present. But when you are dropped of at the same meeting point with a stranger in the middle of the night, that could give an unsafe feeling.

An option to avoid these disadvantages, is to not enforce the users to use the meeting points, but to offer them a discount if they would use them. This is already done by Uber in their express-pool feature. This way, everybody could make use of the service, but you are rewarded if you walk a small distance.

7 Conclusion

In this chapter, an answer to the research question and all subquestions is provided. This chapter starts with a section in which all sub-questions and the main question are answered. In the last section, some suggestions for further research are given.

7.1 Summary and findings

Before drawing the main conclusion, a short elaboration on each sub question is provided.

(1) How should the DARP formulation be adjusted in terms of sets, parameters, variables, objective function and constraints to contain meeting points?

The DARP can be adjusted such that it contains meeting points. To do so, a set of meeting points is added, as well as arcs that make traveling from and to these meeting points possible. Next to that, some new parameters are added, being the maximum walking distance, the maximum total walking time for a passenger and the walking time between vertices. Four new variables are introduced. One is used to indicate if a meeting point is used by a certain request, two are used to transfer the service time and passenger load to a meeting point and the last one is used to keep track of the total walking distance of the passenger. With these new sets, parameters and variables, the original constraints are adjusted and some new constraints are added to find optimal meeting points while finding optimal routes for the DARPmp.

(2) What are preprocessing steps and valid inequalities for the DARPmp and what is the impact of those preprocessing steps and valid inequalities on the computational performance of the DARPmp?

Two preprocessing steps and two valid inequalities for the DARPmp are introduced. The preprocessing steps are the time window tightening and the removal of infeasible arcs. The valid inequalities are the bounds on the service time and a sub-tour elimination constraint. All four have an effect on the computational performance of the Branch-and-cut model for the DARPmp. All preprocessing steps and valid inequalities combined give the best computational performance. The largest instance that could be solved to global optimality, is the instance with 3 vehicles, 24 trip requests and 36 meeting points. Without using the preprocessing steps and valid inequalities, none of the instances could be solved.

(3) What meta-heuristic can be used to approximate the optimal solution of the DARPmp?

In this work, two meta-heuristic algorithms are proposed to solve the Dial-a-Ride problem with meeting points for large-scale instances. Both algorithms are based on a Tabu Search framework, where the first algorithm finds a solution for the DARP and then inserts meeting points after the stopping criterion is reached. The second algorithm inserts meeting points in all solutions in the neighborhood of the current solution. The first algorithm finds solutions with an optimality gap ranging from 0% to 1,65%. The run times for this algorithm range from 14,07 seconds to 850,51 seconds. The second algorithm finds solutions with an optimality gap ranging from 0% to 2,24% and has run times ranging from 38,20 to 9051,69 seconds. Regarding the run times, the first Tabu Search algorithm is always faster. Looking at the solutions found, the first algorithm also produces routes with lower total costs for most of the instances, however this algorithm has an important shortcoming. If the optimal solution for an instance using the DARPmp is infeasible for the DARP, the first algorithm would never find this solution since it only visits solutions that are feasible for the DARP. This shortcoming does not hold for the second algorithm. When more time is spent on implementing the second algorithm in a smarter way, this algorithm would have more potential to find better solutions in shorter run times. Some suggestions to improve the computational performance are described in section 7.2.

(4) What is the impact of using meeting points on the travel time of the passengers and the objective function value comparing the DARP and the DARPmp?

To compare the DARP and the DARPmp, an evaluation framework is proposed to compare the direct travel time to the actual travel time. Comparing the DARP and the DARPmp, there is no evident pattern that shows that one is better than the other. For some combinations of direct travel time and algorithm used, the DARP performs better and in other combinations the DARPmp performs better. Regarding the routing costs, the exact model for the DARPmp always produces lower routing costs than the exact model for the DARP. Both Tabu Search algorithms also find solutions with lower routing costs for the smaller instances, with some exceptions. When the algorithms would be developed further, there is a possibility for also finding better solutions for this instance.

How should the DARP be extended to find optimal meeting points while generating optimal routes?

When combining the answers to the four sub-questions, a main conclusion can be drawn. The aim of this research is to extend the DARP to find optimal meeting points. This is done by formulating a mixed-integer linear program with preprocessing steps and valid inequalities and two meta-heuristics to find solutions for the DARPmp. The DARP can be extended to find optimal meeting points while generating optimal routes by adding some sets, parameters, variables and constraints. Preprocessing steps and valid inequalities can be used to improve the computational performance of the DARPmp. A Tabu Search framework can be used to approximate the optimal solution for the DARPmp. On passenger level, the meeting points sometimes decrease the travel time and in other cases increases the travel time, however it has the potential to minimize the total routing costs.

7.2 Further research

The preprocessing steps and valid inequalities presented in this work have shown to reduce the computational time of the Branch-and-cut algorithm for the DARPmp. Next to the preprocessing steps and valid inequalities used in this work, there are others that are used for the DARP or other types of VRP's. Cordeau (2006) also uses inequalities that bound the load variables w_i^k, capacity constraints, precedence constraints and generalized order constraints as valid inequalities for the DARP. For further research, it can be interesting to explore if these inequalities can also be adapted to be feasible for the DARPmp.

As already mentioned in the discussion, another objective function can have an impact on the resulting set of routes. For further research it could be interesting to use other objective functions in the DARPmp and compare the effects on performance indicators like the total travel costs and detours for the passengers. Other objective function attributes that could be considered are the minimization of the inconvenience for passengers, minimizing the vehicle emissions, minimizing the empty-vehicle kilometres, optimizing the passenger occupancy rate and the matching of passengers in the same vehicle.

Next to another objective function, the effect of other local search operators can be researched. Parragh et al. (2010) used for example the *Swap neighborhood*, *Chain neighborhood* and *Zero split neighborhood* operators to explore neighboring solutions. These operators change sequences of vertices and try to insert them in the best possible way. It could be studied if using these operators will make the algorithm converge to the optimal solution faster. Next to these other operators, other meta-heuristics to approximate the optimal solution of the DARPmp can be explored. Meta-heuristics that are used for other versions of the DARP are for example a Genetic Algorithm (Masmoudi et al., 2017), Deterministic Annealing (Bräysy et al., 2014) and Variable Neighborhood Search (Parragh et al., 2010).

It could also be interesting to study how to improve the computational performance of the current algorithm. The computational performance of the algorithm is already improved in some ways in this work. This is for example done by only evaluating the feasibility and routing costs of routes that are changed, and by first trying to find a feasible insertion position for the first vertex of a request, and then finding a feasible insertion position for the second vertex instead of evaluating this at the same time. Next to this, some improvements can be made in the way the feasibility is checked. Now, the model first calculates the service time of all vertices and determines afterwards if this is feasible regarding the time windows. To improve the computational performance, the time window check should be incorporated in the calculation of the service time and this procedure should be cut off at the first moment an infeasible service time is found.

When the computational performance is improved, it could also be interesting to test the Tabu Search on larger test instances. The repository of benchmark instances also contains instances with 5 to 8 vehicles and 40 to 96 trip requests. It would also be interesting to test the meta-heuristic on the network of a real city in a case study, with real travel data or trip characteristics which is done by Tirachini et al. (2020) in Mexico and by Fielbaum et al. (2021) in Manhattan for example. This way, the effects of using meeting points in larger networks or a real city can be estimated.

References

Alonso, F., Alvarez, M. J., & Beasley, J. E. (2007). A tabu search algorithm for the periodic vehicle routing problem with multiple vehicle trips and accessibility restrictions. *Journal of the Operational Research Society*, 1-14. `https://doi.org/10.1057/palgrave.jors.2602405`

Bongiovanni, C., Kaspi, M., & Geroliminis, N. (2019). The electric autonomous dial-a-ride problem. *Transportation Research Part B*, *122*, 436-456. `https://doi.org/10.1016/j.trb.2019.03.004`

Braekers, K., Caris, A., & Janssens, G. K. (2014). Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots. *Transportation Research Part B*, *67*, 166-186. `https://doi.org/10.1016/j.trb.2014.05.007`

Bräysy, O., Dullaert, W., Hasle, G., Mester, D., & Gendreau, M. (2014). An effective multirestart deterministic annealing metaheuristic for the fleet sieze and mix vehicle-routing problem with time windows. *Transportation Science*, *42*(2), 371-386. `http://dx.doi.org/10.1287/trsc.1070.0217`

Chen, W., Mes, M., Schutten, M., & Quint, J. (2019). A ride-sharing problem with meeting points and return restrictions. *Transportation Science*, *53*(2), 401-426. `https://doi.org/10.1287/trsc.2018.0832`

Cordeau, J. F. (2006). A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, *54*(3), 573-586. `https://doi.org/10.1287/opre.1060.0283`

Cordeau, J. F., Gendreau, M., & Laporte, G. (1997). A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks: An International Journal*, *30*(2), 105-119. `https://doi.org/10.1002/(SICI)1097-0037(199709)30:2<105::AID-NET5>3.0.CO;2-G`

Cordeau, J. F., & Laporte, G. (2003). A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B*, *37*, 579-594. `https://doi.org/10.1016/S0191-2615(02)00045-0`

Cordeau, J. F., & Laporte, G. (2007). The dial-a-ride problem: models and algorithms. *Annals of operations research*, *153*(1), 29-46. `DOI:10.1007/s10479-007-0170-8`

Fielbaum, A., Bai, X., & Alonso-Mora, J. (2021). On-demand ridesharing with optimized pick-up and drop-off walking locations. *Transportation Research Part C*, *126*, 1-24. `https://doi.org/10.1016/j.trc.2021.103061`

Furuhata, M., Dessouky, M., Ordóñez, F., Brunet, M. E., Wang, X., & Koenig, S. (2013). Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B*, *57*, 28-46. `https://doi.org/10.1016/j.trb.2013.08.012`

Gendrau, M. (1994). A tabu search heuristic for the vehicle routing problem. *Management Science*, *40*(10), 1276-1290. `https://doi.org/10.1287/mnsc.40.10.1276`

Glover, F. (1986). Future paths for integer programming and links to artifical intelligence. *Computers Operations Research*, *13*(5), 533-549. `https://doi.org/10.1016/0305-0548(86)90048-1`

Gökay, S. (2021). *Scalable real-time ride-sharing with meeting points for flexible on-demand public transportation* (Unpublished doctoral dissertation). Aaachen: RTW Aachen University.

Ho, S., Nagavarapu, S. C., Pandi, R. R., & Dauwels, J. (2018). An improved tabu search heuristic for static dial-a-ride problem.
`https://doi.org/10.48550/arXiv.1801.09547`

Jacobsen, S. H., & King, D. M. (2009). Fuel saving and ridesharing in the us: Motivations, limitations, and opportunities. *Transportation Research Part D*, *14*, 14-21. `https://doi.org/10.1016/j.trd.2008.10.001`

Johnsen, L. C., & Meisel, F. (2022). Interrelated trips in the rural dial-a-ride problem with autonomous vehicles. *European Journal of Operational Research*. https://doi.org/10.1016/j.ejor.2022.02.021

Kirchler, D., & Calvo, R. W. (2013). A granular tabu search algorithm for the dial-a-ride problem. *Transportation Research Part B*, *56*, 120-135. https://doi.org/10.1016/j.trb.2013.07.014

Ma, T. Y. (2021). Two-stage battery recharge scheduling and vehicle-charger assignment policy for dynamic electric dial-a-ride services. *PloS one*, *16*(5), e0251582. https://doi.org/10.1371/journal.pone.0251582

Malheiros, I., Ramalho, R., Passeti, B., Bulhões, T., & Subramanian, A. (2021). A hybrid algorithm for the multi-depot heterogeneous dial-a-ride problem. *Computers and Operations Research*, *129*, 105196. https://doi.org/10.1016/j.cor.2020.105196

Masmoudi, M. A., Breakers, K., Masmoudi, M., & Dammak, A. (2017). A hybrid genetic algorithm for the heterogeneous dial-a-ride problem. *Computers and Operations Research*, *81*, 1-13. https://doi.org/10.1016/j.cor.2016.12.008

Masmoudi, M. A., Hosny, M., Breakers, K., & Dammak, A. (2016). Three effective metaheuristics to solve the multi-depo multi-trip heterogeneous dial-a-ride problem. *Transportation Research Part E*, *96*, 60-80. https://doi.org/10.1016/j.tre.2016.10.002

Masmoudi, M. A., Hosny, M., Demir, E., Genikomsakis, K. N., & Cheikhrouhou, N. (2018). The dial-a-ride problem with electric vehicles and battery swapping stations. *Transportation Research Part E*, *118*, 392-420. https://doi.org/10.1016/j.tre.2018.08.005

Masoud, N., & Jayakrishnan, R. (2017). A decomposition algorithm to solve the multi-hop peer-to-peer ride-matching problem. *Transportation Research Part B*, *99*, 1-29. https://doi.org/10.1016/j.trb.2017.01.004

Masson, R., Lehuédé, F., & Péton, O. (2014). The dial-a-ride problem with transfers. *Computers Operations Research*, *41*, 12-23. https://doi.org/10.1016/j.cor.2013.07.020

Militão, A. M., & Tirachini, A. (2021). Optimal fleet size for a shared demand-responsive transport system with human-driven vs automated vehicles: A total cost minimization approach. *Transportation Research Part A*, *151*, 52-80. https://doi.org/10.1016/j.tra.2021.07.004

Minieka, E. (1970). The m-center problem. *SIAM Review*, *12*(1), 138-139. https://doi.org/10.1137/1012016

Molenbruch, Y., Braekers, K., & Caris, A. (2017). Typology and literature review for dial-a-ride problems. *Annals of Operations Research*, *259*, 295-325. https://doi.org/10.1007/s10479-017-2525-0

Mourad, A., Puchinger, J., & Chu, C. (2019). A survey of models and algorithms for optimizing shared mobility. *Transportation Research Part B*, *123*, 323-346. https://doi.org/10.1016/j.trb.2019.02.003

Parragh, S. N., Doerner, K. F., & Hartl, R. F. (2010). Variable neighborhood search for the dial-a-ride problem. *Computers Operations Research*, *37*, 1129-1138. https://doi.org/10.1016/j.cor.2009.10.003

Pierotti, J., & van Essen, J. T. (2021). Milp models for the dial-a-ride problem with transfers. *EURO Journal on Transportation and Logistics*, *10*, 100037. https://doi.org/10.1016/j.ejtl.2021.100037

Pimenta, V., Quilliot, A., Tousaint, H., & Vigo, D. (2017). Models and algorithms for reliability-oriented dial-aride with autonomous electric vehicles. *European Journal of Operational Research*, *257*, 601-613. https://doi.org/10.1016/j.ejor.2016.07.037

Potvin, J. Y., & Rousseau, J. M. (1995). An exchange heuristic for routeing problems with time windows.

Journal of the Operational Research Society, *46*, 1433-1446. https://doi.org/10.1057/jors.1995.204

Stiglic, M., Agatz, N., Savelsbergh, M., & Grandisar, M. (2015). The benefits of meeting points in ride-sharing systems. *Transportation Research Part B*, *82*, 36-53. https://doi.org/10.1016/j.trb.2015.07.025

Tirachini, A., Chaniotakis, E., Abouelela, M., & Antoniou, C. (2020). The sustainability of shared mobility: Can a platform for shared rides reduce motorized traffic in cities? *Transportation Research Part C*, *117*, 102707. https://doi.org/10.1016/j.trc.2020.102707

Tirachini, A., & Gomez-Lobo, A. (2020). Does ride-hailing increase or decrease vehicle kilometers traveled (vkt)? a simulation approach for santiago de chile. *International Journal of Sustainable Transportation*, *14*(3), 187-204. https://doi.org/10.1080/15568318.2018.1539146

A Appendices

A.1 List of all symbols

A	:	Set of arcs of graph $G = (V, A)$
C^{min}	:	Minimum allowed walking distance
C^{max}	:	Maximum allowed walking distance
D	:	Set of delivery vertices
G	:	Graph representing the network. $G = (V, A)$, or $G = (X, \Gamma)$
K	:	Set of available vehicles
L	:	Maximum allowed ride time
M	:	Large number used for the big-M constraints
$N(S)$	:	The neighborhood of solution S
P	:	Pick-up vertices
Q_k	:	Capacity of vehicle k
Q	:	Set of infinite number of points in the whole graph $G = (V, E)$
Q'	:	Set of finite number of points in the whole graph $G = (V, E)$
S	:	The current solution in the Tabu Search algorithm
S'	:	The new found solution in the Tabu Search algorithm
S^0	:	The initial solution for the Tabu Search algorithm
S^{best}	:	The best found solution for the Tabu Search algorithm so far
T_k	:	Maximum route duration
V	:	Set of all vertices of graph $G = (V, A)$, where $V = \{P, D, \{0, 2n + 1\}\}$
W	:	Maximal walking time for each passenger
X	:	Set of vertices in the graph $G = (X, \Gamma)$
$\mathcal{A}$	:	Set of arcs in graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$
$\mathcal{G}$	:	Graph representing the network with meeting points
$\mathcal{M}$	:	Set of meeting points
$\mathcal{M}_d$	:	Set of delivery meeting points
$\mathcal{M}_p$	:	Set of pick up meeting points
$\mathcal{N}$	:	Set of vertices in graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$
Γ	:	Set of undirected edges in graph $G = (X, \Gamma)$
Π	:	Set of infinite number of points in all edges of graph $G = (X, \Gamma)$
Π'	:	Set of finite number of points in all edges of graph $G = (X, \Gamma)$
c_{ij}^k	:	Costs for traversing arc $(i, j) \in A$ with vehicle k
d_i	:	Sevice time of vertex i
e_i	:	Start time of time window for vertex i
h	:	The average walking speed of a passenger
i	:	Used to indicate the start node
j	:	Used to indicate the end node
k	:	Used to indicate the vehicle number
l_i	:	End time of the time window for vertex i
m	:	Used to indicate meeting point vertices
n	:	Number of requests

p_{im}	:	The walking time from vertex $i \in V$ to meeting point $m \in \mathcal{V}$
q_i	:	Passenger demand at node i
r_i^k	:	Variable to indicate the ride time of for passenger i on vehicle k
s_{im}	:	Variable to transfer service time duration of vertex i to meeting point m
t_{ij}	:	Travel time from vertex i to vertex j
u_i^k	:	Variable to indicate the time vehicle k starts servicing vertex i
v_i^k	:	Variable to save the walking time of passenger i
w_i^k	:	Variable that indicates the load of vehicle k when leaving node i
x_{ij}^k	:	Variable that indicates if arc $(i,j) \in A$ is traversed by vehicle k
y_{im}^k	:	Variable that indicates if vertex i is coupled to meeting point m in vehicle k
z_{im}	:	Variable to transfer the passenger demand at node i to meeting point m
$\delta_{i,\pi}$	:	The shortest distance from vertex $i \in X$ to point π on edge ε
ε	:	Used to indicate an edge
ε_s	:	Starting point of edge ε
ε_t	:	End point of edge ε
κ	:	The number of times the Tabu Search can return to S^{best} without finding a better solution
μ	:	Used to indicate a second meeting point
τ_ε	:	The length of edge ε
ϕ	:	Number of iterations after which the Tabu Search returns to S^{best}

A.2 Toy network

Table A.2: Input for the toy network

i	x_c	y_c	d_i	q_i	e_i	l_i
0	0.000	0.000	0	0	0	300
1	-6.643	6.976	3	1	50	150
2	-7.278	6.934	3	1	50	130
3	-18.176	-6.614	3	1	70	160
4	-10.003	-6.296	3	1	80	180
5	5.879	11.543	3	1	20	80
6	5.486	12.134	3	1	20	80
7	6.304	12.067	3	1	20	80
8	16.263	8.382	3	1	50	150
9	-19.512	-0.265	3	-1	100	150
10	-12.502	4.361	3	-1	100	130
11	-9.956	-4.879	3	-1	120	160
12	-2.314	-4.765	3	-1	150	180
13	13.234	1.452	3	-1	80	100
14	15.859	4.976	3	-1	80	100
15	15.663	8.382	3	-1	60	100
16	7.245	-2.000	3	-1	100	130
17	0.000	0.000	0	0	0	300

The other input values are:

- $K = \{0, 1\}$
- $T_k = 300$
- $Q_k = 3$
- $L = 30$
- $C^- = 0.7$
- $C = 0.75$
- $h = 0.25$
- $W = 10$

A.3 Local search operators

Relocate

In Figure A.1, a visual representation of the *relocate* operator is given. In this example, the passenger in the first vehicle that is represented by the dark blue circles gets relocated to the other vehicle. This is done by taking out the passenger from the first route and joining the remaining pick up and drop off vertices of this route together. Next, it is tried to find a feasible insertion position for the pick-up vertex (O). When a feasible insertion position is found, the delivery vertex (D) is inserted in a feasible position. The resulting route set in the last row of the figure is one example of a route set that can be created using this operator. The entire neighborhood can consist of more routes.

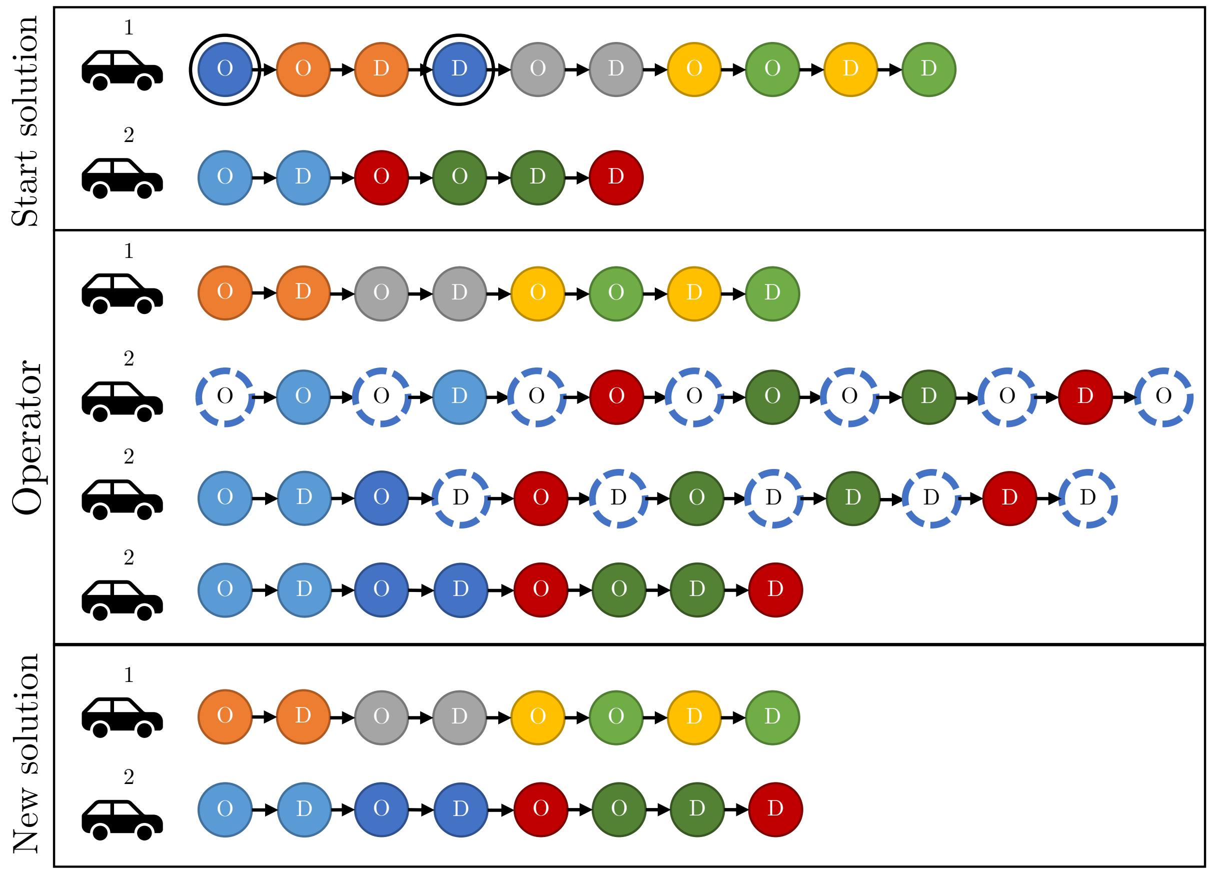


Figure A.1: Visual representation of retrieving a new solution using the *relocate* operator

Exchange

In Figure A.2, a visual representation of the *exchange* operator is given. In this example, the passenger in the first vehicle that is represented by the dark blue circles gets exchanged with the yellow passenger in the second vehicle. This is done by taking out the yellow passenger from the second route and inserting it in the first route at the exact same position as the dark blue passenger. Next, it is tried to find a feasible insertion position for the pick-up vertex (O) of the dark blue passenger in the second route. When a feasible insertion position is found, the delivery vertex (D) is inserted in a feasible position. The resulting route in the last row of the figure is one example of a route set that can be created using this operator. The entire neighborhood can consist of more routes.

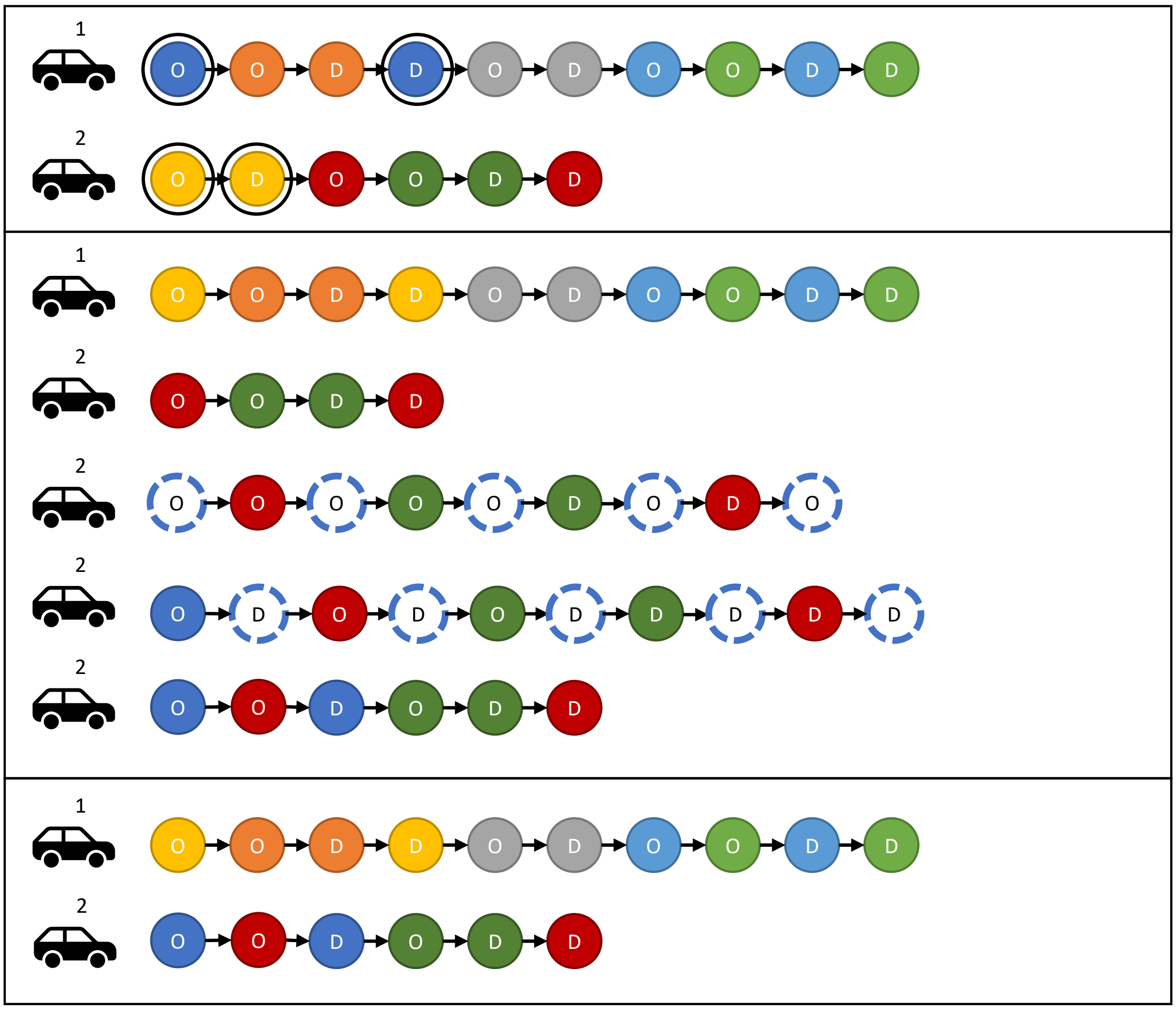


Figure A.2: Visual representation of retrieving a new solution using the *exchange* operator

2-opt

In Figure A.3, a visual representation of the *2-opt* operator is given. In the first step, for each arc in the solution it is determined if the vehicle is empty. If the vehicle is empty, the route can be sliced in half at this arc. This is done for both vehicles. In the next step, the first part of the first route is joined to the second part of the second route and vice versa. Note that a slice could also have been made between the drop off vertex (D) of the grey passenger and the pick up vertex (O) of the yellow passenger. The resulting route set in the last row of the figure is one example of a route set that can be created using this operator. The entire neighborhood can consist of more routes.

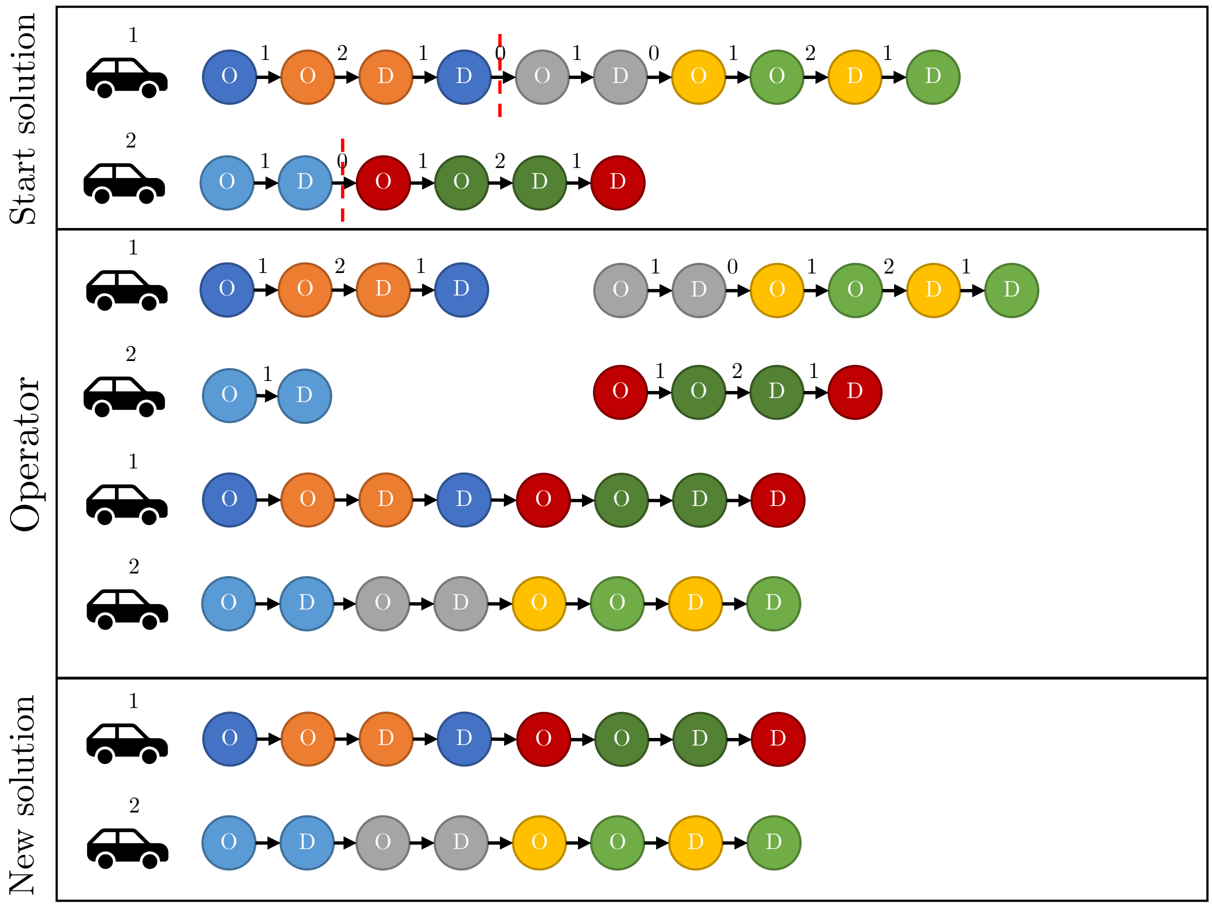


Figure A.3: Visual representation of retrieving a new solution using the *2-opt* operator

r-4-opt

In Figure A.4, a visual representation of the *r-4-opt* operator is given. In the first step, 3 vertices of the route of the second vehicle are selected. For each order of these 3 vertices, the feasibility is checked. In the last row, one of the possible routes for the second vehicles is chosen. The resulting route set in the last row of the figure is one example of a route set that can be created using this operator. The entire neighborhood can consist of more routes.

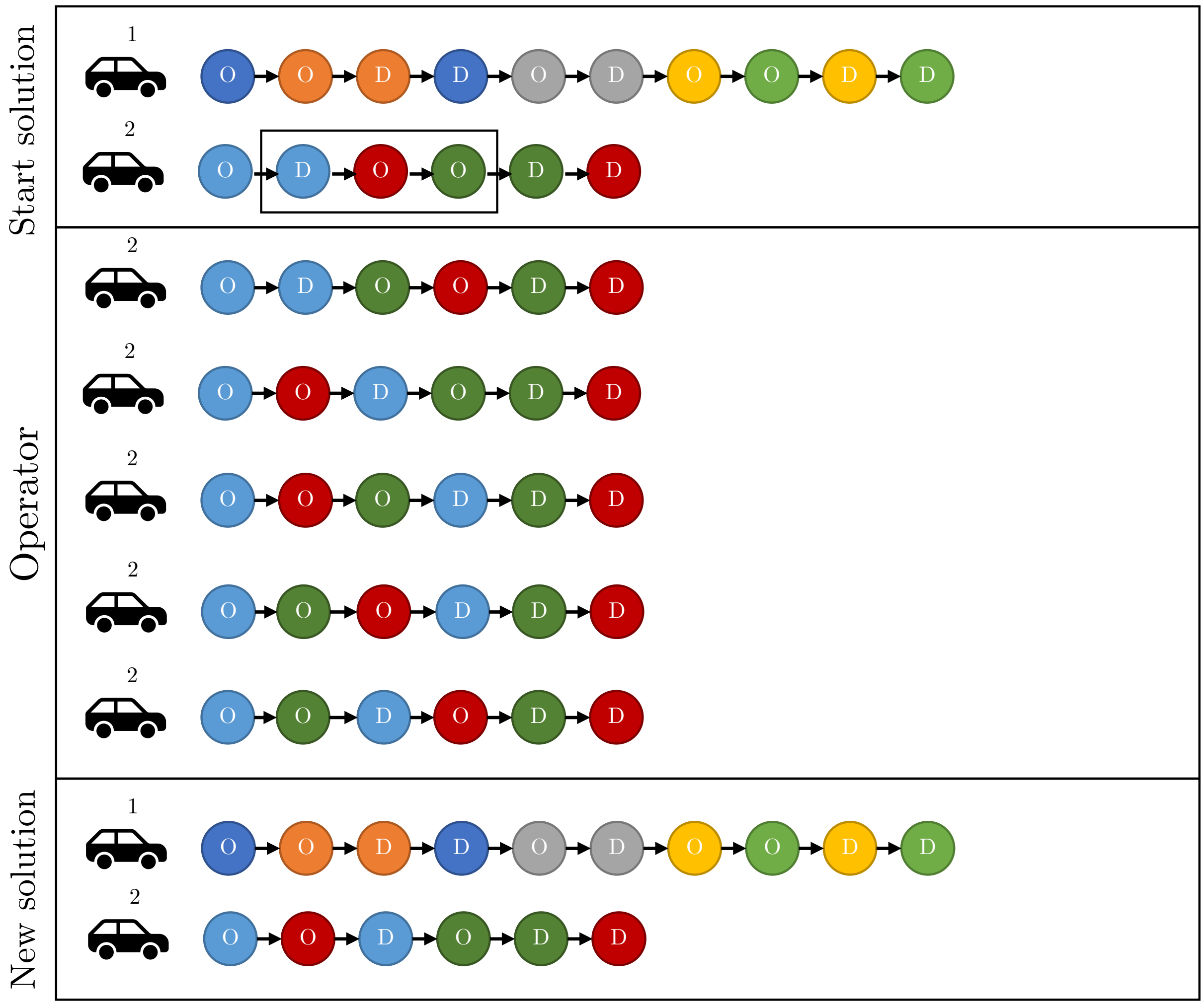


Figure A.4: Visual representation of retrieving a new solution using the *r-4-opt* operator

A.4 Detailed result descriptions

The effect of preprocessing steps and valid inequalities

Only looking at the experiments plotted in Figure 5.1, it can be observed that the run times are highest when only applying V2. When also incorporating P1, the run times already drop significantly. When using P1 in combination with V1, the run time drops even more, especially for the instances with lower numbers of meeting points. For the next experiment with P1, P2 and V1 incorporated in the model, the pattern of the run times looks similar to the experiment when only using P1 and V1. For 9 instances, the run time is somewhat faster and for the other 8 instances the run time is somewhat slower. When using the combination of P1, P2 and V2, the run time for the instances with a smaller number of meeting points is higher than in the previous experiment, whereas the run time for the instances with more meeting points is lower compared to the previous described experiments. In the experiment in which P1, V1 and V2 are used, the run times decrease even more for all numbers of meeting points. When applying all the preprocessing steps and valid inequalities at the same time, the pattern of the run times looks similar to the experiment with P1, V1 and V2 applied, but the run times are lower for almost every instance.

Evaluation framework

When looking at Figure 5.3a it can be observed that the absolute extra travel time for the DARP and the DARPmp have the same pattern. For the first three bins with smaller direct travel times, the extra time is also smaller for both models. For the next bin, with direct travel times ranging from 10 to 12.5, the extra time is larger for both models. From that point, for the bins with larger direct travel times the extra time seems to decrease again. Overall, the average extra time is larger for the DARPmp than for the original DARP for each bin, except for bin 10-12,5 where the extra time is on average slightly larger for the DARP model. In Figure 5.3b, the maximum extra travel times for each bin are displayed. This figure shows that the maximum extra time, decreases as the direct travel time increases. For the smaller direct distances, the maximal extra travel time time is larger for the DARP than for the DARPmp in the first two bins. For the other bins, the maximum extra travel time is larger for the DARPmp.

For TS1 the average extra travel time is plotted in Figure 5.3c. The average extra travel time shows the same pattern as the B&C VP, except for bin 5-7,5, which shows a slightly higher average time than bin 7,5-10. For each bin, the extra travel time is higher for the DARPmp, except for bin 10-12,5. In Figure 5.3d, the maximum found extra travel time is displayed. For the DARPmp, the maximal extra time is decreasing as the direct travel time increases. This is the same for the DARP, except for bin 10-12,5. The maximum found extra travel time is larger for the DARP in bin 0-5 and 10-12,5. For the other bins, the extra travel time is larger for the DARPmp.

The average extra time for Tabu Search algorithm 2 is displayed in Figure 5.3e. The pattern looks the same as in 5.3c, where the average extra time is first increasing and then again decreasing, with for bin 5-7,5 a slightly higher value than for bin 7,5-10. The average extra travel time is larger for the DARPmp than for the DARP, except for bin 10-12,5. In Figure 5.3f, the maximum extra time for Tabu Search algorithm 2 is displayed. Again, the maximum extra time seems to decrease as the direct travel time decreases. The maximum extra time for bin 0-5, 5-10 and 10-12,5 is larger for the DARP and for the other bins the extra time is larger for the DARPmp.