BERGISCHE
UNIVERSITÄT
WUPPERTAL

Exact and Heuristic Methods for Dial-a-Ride Problems

Dissertation
zur Erlangung des Doktorgrades (Dr. rer. nat.)
Fakultät für Mathematik und Naturwissenschaften
Bergische Universität Wuppertal

vorgelegt von
Daniela Gaul

Wuppertal, Dezember 2023

Acknowledgments

I would like to thank my supervisors Kathrin Klamroth and Michael Stiglmayr for their guidance throughout the last couple of years. Thank you for supporting me with numerous ideas, rich discussions and valuable feedback. During times of frustration, it always kept me motivated to know that I could talk to you about any problems I was facing. The latter is something I also highly appreciate about my colleagues. Thank you all for welcoming me so warmly here in Wuppertal, and for making it a supportive environment. One thing I really enjoyed during my time as a doctoral student is visiting conferences and meeting other researches from all over the world. In particular, I am glad to have met Arne Schulz and Christian Pfeiffer, who also work on the dial-a-ride problem and who have contributed to the results of this thesis with great ideas. Furthermore, I would like to thank Hanno Gottschalk and Hayk Asatryan for enriching this thesis with their excellence from the field of statistics and for the friendly cooperation during our joint work.

Besides fruitful discussions at work, it is at least just as important to have some people in your life that you can always count on. Therefore, I would like to thank my parents, for always being there for me with advice and for supporting me in the decisions I make. I would like to thank my brother Oliver, for convincing me that I should do a doctorate, and for discussing with me about my research too. Last but not least, dear Thomas, thank you for being by my side through all the years, and for your endless love, support and encouragement.

Since this work has been supported financially through bergisch.smart_mobility, I would like to thank the Federal Ministry for Economics, Innovation, Digitalization and Energy of North-Rhine Westphalia for their funding. Moreover, many thanks go to WSW for the cooperation within bergisch.smart_mobility and for providing us with *Hol mich! App* and WSW-LAN data.

Contents

1 Introduction

Recently, ridepooling services have emerged in many big cities and rural areas all around the globe, with a majority of ridepooling services launched in Europe (Foljanty, 2020). Ridepooling services are taxi-like services which are usually booked via a smartphone app. In contrast to taxi services, users may be pooled, i.e., they may have to share parts of their rides in the same vehicle if they head in a similar direction during the same period of time. Ridepooling bridges a gap between rather inflexible line-based public transport (e.g., buses and trains) and relatively expensive taxi services. It provides an alternative to motorized private transport and thus has the potential to reduce the number of vehicles in the cities (ITF, 2015). From an environmental perspective, the benefits of ridepooling services still have to be investigated. As stated in Anair et al. (2020), a ridepooling trip shared between two passengers is similar in emissions to a private vehicle trip, but about 33 percent better than a taxi trip.

To pool rides, service operators need to solve two interrelated tasks: the assignment of customers to vehicles and the computation of vehicle schedules. These tasks have to be solved under consideration of operating costs and customer satisfaction. In mathematical optimization, this combinatorial optimization problem is called the *dial-a-ride problem* (DARP). For a small number of requests the DARP could be solved by enumerating all feasible assignments of customers to vehicles and the corresponding vehicle routes, and by determining the lowest-cost solution subsequently. However, this is not very efficient and becomes impractical with a growing number of requests. Thus, efficient optimization algorithms are required to simultaneously plan routes and pool users in shared rides.

In this thesis, we consider exact and heuristic solution methods for the DARP inspired by the on-demand ridepooling service *Hol mich! App*[1] which was established in the city of Wuppertal (Germany) in 2019 in collaboration with the project bergisch.smart_mobility[2]. This ridepooling service is run by the local public transport provider and is designed to complement the bus service. The most important application of the DARP is probably on-demand ridepooling (Ho et al., 2018), but it has various applications in modern transportation systems, such as the transportation of patients, or in a related branch of research, the transportation of goods.

We consider two variants of the DARP. While in on-demand ridepooling services transport requests arrive throughout the day, we also consider the static case where all transport requests are known in advance. This results in the dynamic DARP and the static DARP, respectively.

The static DARP is expressed as an *optimization problem* $\min\{f(x)\colon x \in X\}$, i.e., given a set of feasible solutions $X \subseteq \mathbb{R}^n$ and an objective function $f\colon X \to \mathbb{R}$, find the optimal solution $x^\star \in X$ minimizing f, or state that no such solution exists. An optimization problem is called a *mixed-integer linear program (MILP)*, if the objective function is linear,

[1] `https://www.holmich-app.de/`
[2] `https://www.bergischsmartmobility.de/en/the-project/`

and if the feasible set X is a subset of $\mathbb{R}^p \times \mathbb{Z}^{n-p}$ that can be described using linear inequalities. To solve an MILP, one option is to pass it to a mathematical solver, which in its core uses an algorithm that iteratively partitions the feasible set into subsets. It avoids a complete enumeration of all feasible solutions by ruling out the existence of an optimal solution in certain subsets using upper and lower bounds on the objective function value. We discuss so-called *branch-and-bound algorithms* and other standard solution techniques for MILPs in Section 2.2. Nevertheless, in general, MILPs (and also the DARP) are complex problems, and the performance of the solver strongly depends on the formulation of the problem, i.e., on the linear inequalities describing the set X. The MILP formulations of the DARP known from the literature have in common that binary variables represent the decision whether two locations (i.e., the vehicle depot, pick-up or delivery locations) are visited directly after one another. We refer to these formulations as *location-based*.

In this thesis, we present a new perspective on modeling dial-a-ride problems. We introduce an *event-based (EB)* MILP formulation of the static DARP, where binary variables are based on tuples representing feasible allocations of users to a vehicle, additionally storing the location of the last picked up or dropped off user in the first component. These tuples are referred to as *events*. The EB MILP constitutes the basis for the contributions of this thesis. We show that it outperforms state-of-the-art location-based formulations of the DARP. By using the best of both worlds, the EB model is combined with a location-based formulation into a new location-augmented-event-based MILP, whose superiority compared to state-of-the-art location-based formulations is demonstrated theoretically and computationally. The concept of event-based modeling is transferred to the dynamic DARP by introducing a dynamic event-based MILP which is solved in a rolling-horizon algorithm every time new requests arrive. The rolling-horizon algorithm stands out from existing solution methods for the dynamic DARP in that it finds optimal insertion positions for new requests in 99.5% of all iterations given a time limit of 30 seconds. Hence, there is hardly any need for re-optimization in the idle time between new requests. The rolling-horizon algorithm is then used in a case-study investigating the replacement of buses by ridepooling services during the late evening hours and w.r.t. service criteria.

We now shortly summarize the contents of each chapter of this thesis.

1.1 Structure of this work

The contributions described above are divided into four main chapters. Furthermore, this thesis includes two additional chapters, reviewing basic mathematical concepts and providing an overview of the DARP, respectively.

In Chapter 2, we review concepts from graph theory, mixed-integer optimization and multi-objective optimization.

The existing literature on the DARP, distinguishing between static and dynamic DARP as well as exact and heuristic solution methods, is described in Chapter 3. Each solution method is briefly explained and the relevant literature is summarized. Furthermore, two paragraphs focus on multi-objective optimization, and simulations and case studies, respectively. After that, real-life applications of the DARP are described, followed by a formal definition of the DARP. We present two standard MILP formulations from the

literature. In the next section, we briefly discuss different objectives by which the DARP is characterized. We conclude this chapter by a section on related problems such as vehicle routing problems and the minimum cost flow problem.

In Chapter 4, two event-based MILP formulations are presented, which can be distinguished by their representation of time window and ride time constraints. The event-based formulations have the advantage that capacity, pairing and precedence constraints, needed in location-based formulations to ensure the feasibility of the solution, are implicitly encoded. This comes at the cost of significantly more variables and constraints compared to compact location-based models. In computational tests, we show that the event-based MILP formulations outperform the state-of-the-art location-based formulations. The superior of the two proposed models lays a foundation for the following chapters of this thesis and is referred to as the event-based (EB) MILP. In a second part of our computational study, we consider weighted-sums of optimization criteria and investigate the respective trade-offs between different optimization goals.

In Chapter 5, we combine advantages from location-based and event-based models into a *location-augmented-event-based* (LAEB) and an *aggregated location-augmented-event-based* (ALAEB) formulation, which can be distinguished by the type of binary variables (i.e., location- or event-based). We show that the location-augmented-event-based models are tighter than state-of-the-art location-based formulations. Moreover, they yield an integral polyhedron if the time windows fulfill the condition that they induce a unique pairwise ordering of locations. The new models are strengthened even further by novel valid inequalities tailored to the event-based approach, and by preprocessing techniques which detect infeasible events. By using the preprocessing techniques before passing the LAEB formulation to a solver, computational times are reduced by more than half compared to the EB MILP.

The (location-augmented-)event-based formulations are easy to implement and can be solved by standard MILP solvers. Instances with up to 100 requests can be solved within a few seconds. Other exact approaches for the static DARP (e.g., Gschwind and Irnich, 2015; Rist and Forbes, 2021) are usually based on more complex solution methods such as branch-and-cut or column generation. Hence, the (location-augmented-)event-based formulations are an efficient and low-maintenance tool for service providers since they generate high-quality solutions in a short amount of time.

In the second part of this thesis, we turn our attention to the dynamic DARP. Solution methods for the dynamic DARP are mainly heuristic approaches combining two steps (e.g., Berbeglia et al., 2012; Häll and Peterson, 2013): First, an incoming request is inserted into the current solution by using a simple and fast insertion heuristic. Then, in the idle time between two incoming requests, the solution is re-optimized using a more complex heuristic or meta-heuristic. In Chapter 6, a rolling-horizon algorithm for the dynamic DARP is presented, which is based on a dynamic event-based model and iteratively solves a reduced event-based MILP formulation every time new requests arrive. In numerical tests, the rolling-horizon algorithm computes optimal insertion positions w.r.t. the current schedule in 99.5% of all iterations, which cannot be guaranteed in general by two-phase heuristics.

In Chapter 7 we investigate the effects of replacing line-based public transport services during the late evening hours by ridepooling services with regard to service quality. To-

wards this end, artificial ridepooling requests are generated based on real-life data from bus passengers and *Hol mich! App* passengers by using a predictive simulation. We use the rolling-horizon algorithm to model the routing decisions of the ridepooling service, enhancing it by a feasible-path heuristic which compensates for request peaks and reduces computation time.

The thesis ends with Chapter 8, where we summarize and conclude the main contributions and make some remarks on future research.

A list of parameters and variables can be found in the nomenclature after Chapter 8.

1.2 Publications

The content of this thesis has been published or submitted to scientific journals:

- H. Asatryan et al. (2023). "Ridepooling and public bus services: A comparative case-study". *Submitted to EURO Journal on Transportation and Logistics.* DOI: 10.48550/ARXIV.2302.01709
- D. Gaul et al. (2021). "Solving the dynamic dial-a-ride problem using a rolling-horizon event-based graph". In: *21st Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2021)*. Ed. by M. Müller-Hannemann and F. Perea. Vol. 96. Open Access Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 8:1–8:16. DOI: 10.4230/OASIcs.ATMOS.2021.8
- D. Gaul et al. (2022). "Event-based MILP models for ridepooling applications". In: *European Journal of Operational Research* 301.3, pp. 1048–1063. DOI: 10.1016/j.ejor.2021.11.053
- D. Gaul et al. (2023). "A tight formulation for the dial-a-ride problem". *Submitted to European Journal of Operational Research.* DOI: 10.48550/ARXIV.2308.11285

Chapter 4 is published in Gaul et al. (2022). The content of Chapter 5 is based on Gaul et al. (2023). Chapter 6 is published in Gaul et al. (2021), while Chapter 7 is aligned to Asatryan et al. (2023). Contents of all of the above articles appear in Chapter 3.

Furthermore, the C++ implementation of the presented MILPs and algorithms is published in the following git repositories:

- D. Gaul (2022b). "Rolling-horizon algorithm for the dynamic DARP". Git repository. URL: https://git.uni-wuppertal.de/dgaul/rolling-horizon-algorithm-for-dynamic-darp
- D. Gaul (2022a). "Event-based MILP for the DARP". Git repository. URL: https://git.uni-wuppertal.de/dgaul/event-based-milp-for-darp
- D. Gaul (2023). "A tight formulation for the DARP". Git repository. URL: https://git.uni-wuppertal.de/dgaul/a-tight-formulation-for-the-darp

2 Basic Concepts

This chapter introduces basic mathematical concepts and notation and is divided into three sections: graph theory, which is discussed in Section 2.1, followed by mixed-integer linear optimization in Section 2.2 and multi-objective optimization in Section 2.3. With these three sections, we provide the main prerequisites required to formulate the dial-a-ride problem as a mixed-integer linear program based on a directed graph, and to consider it from a multi-objective perspective.

2.1 Graph Theory

The graph theoretical concepts reviewed in this section are mainly based on Ahuja et al. (1993). Basic concepts such as directed graphs, paths and cycles are defined. Moreover, we present the all-pairs shortest path problem and the Floyd-Warshall algorithm for its solution. We begin with several elementary definitions.

Definition 2.1. *A* directed graph $G = (V, A)$ *consists of a finite set* $V \neq \emptyset$ *of nodes and a set* $A \subseteq V \times V$ *of arcs whose elements are ordered pairs of distinct nodes. A directed graph whose nodes and arcs have associated numerical values (e.g., demands or costs) is also referred to as* directed network*.*

Definition 2.2. *Let* $(v, w) \in A$ *be a directed arc. We refer to node* v *as the* tail *of arc* (v, w) *and to node* w *as its* head*. Then, the arc* (v, w) *is* incident *to nodes* v *and* w*. The arc* (v, w) *is an* outgoing arc *of node* v *and an* ingoing arc *of node* w*. We denote the set of outgoing arcs of node* v *as* $\delta^{out}(v) \coloneqq \{(v, w) \in A\}$ *and the set of ingoing arcs of node* v *as* $\delta^{in}(v) \coloneqq \{(w, v) \in A\}$*. Whenever an arc* $(v, w) \in A$*, we say that node* v *is* adjacent *to node* w*.* Multiarcs *are two or more arcs with the same tail and head nodes.*

Definition 2.3. *A directed graph* $G' = (V', A')$ *is a* subgraph *of* $G = (V, A)$ *if* $V' \subseteq V$ *and* $A' \subseteq A$*. We say that* $G' = (V', A')$ *is the subgraph of* G induced *by* V' *if* A' *contains each arc of* A *with both endpoints in* V'*, i.e.,* $A' = \{(v, w) \in A \colon v, w \in V'\}$*.*

With the definition of a subgraph at hand, we define further structures in a graph.

Definition 2.4. *A* directed path *in a directed graph* $G = (V, A)$ *is a subgraph of* G *consisting of a sequence of nodes and arcs* $(v^1, a^1, v^2, a^2, \ldots, a^{t-1}, v^t)$ *such that* $a^i = (v^i, v^{i+1}) \in A$ *for all* $i \in \{1, \ldots, t-1\}$ *and* $v^i \neq v^j$ *for all* $i, j \in \{1, \ldots, t\}$ *with* $i \neq j$*,* $\{i, j\} \neq \{1, t\}$*. If the considered graph does not contain multiarcs, we sometimes write* $v^1 \to v^2 \to \ldots \to v^t$ *without explicitly mentioning the arcs.*

Definition 2.5. *A* dicycle *is a directed path* $v^1 \to \ldots \to v^t$ *in a directed graph together with the arc* $(v^t, v^1) \in A$*.*

Definition 2.6. *A dicycle in a directed graph $G = (V, A)$ which visits each node $v \in V$ exactly once is called a* Hamiltonian cycle *or* Hamiltonian tour*.*

In derogation of this definition, in this thesis we often use the term *(vehicle) tour* to refer to a dicycle which starts and ends at a distinguished node, the vehicle depot. In the case of one node being associated with the start, and another node being associated with the end depot, we also use the term "vehicle tour" to describe a directed path starting at the start depot and ending at the end depot.

Definition 2.7. *A directed graph is called* simple *if it does not contain multiarcs. A* complete *directed graph is a simple directed graph in which all nodes $v \neq w \in V$ are adjacent, i.e., $(v, w), (w, v) \in A$ for all $v \neq w \in V$. We say that a directed graph is* connected*, if it contains a directed path between every pair of nodes.*

Remark 2.8. *In this thesis, we usually refer to a simple and connected directed graph, or to a simple and connected directed network, as* graph*. We usually refer to a directed path as* path *and to a directed cycle as* cycle *or* dicycle*.*

In the following, we take a look at the computation of shortest paths in a graph and define the *shortest path problem* and the *all-pairs shortest path problem.*

Definition 2.9. *Let $G = (V, A)$ be a graph with an* arc length *(or* arc cost*) c_a associated with each arc $a = (v, w) \in A$. We define the* length of a directed path *as the sum of the lengths of the arcs in the path. Let $s \in V$ be a distinguished node, called the* source*. The* shortest path problem *is to determine for every non-source node $v \in V$ a shortest path from s to v. The* all-pairs shortest path problem *is to determine for every pair of nodes $(v, w) \in V \times V$ a shortest path from v to w.*

The literature classifies solution approaches for shortest path problems in two groups: label-setting and label-correcting algorithms. They assign distance labels, which are upper bounds on the shortest path lengths, to nodes at each step. Label-setting algorithms nominate one label as permanent in every iteration, while in label-correcting algorithms all labels are temporary until the final step of the algorithm. The main difference is that label-setting algorithms are only applicable to cycle-free graphs or graphs with non-negative arc lengths. Label-correcting algorithms either identify a negative cycle when one exists, or solve the shortest path problem if not. The shortest path problem with non-negative arc lengths can be solved using *Dijkstra's algorithm*, see, e.g., Ahuja et al. (1993), which is a label-setting algorithm. Since in this thesis we consider real-world street networks it is not restrictive to assume non-negative arc lengths representing the physical length of the street segment or the corresponding travel time. For the same reason, we may assume that a graph does not contain a negative cycle, i.e., a dicycle of negative length, which is a requirement for the label-correcting *Floyd-Warshall algorithm* to solve the all-pairs shortest path problem: The Floyd-Warshall algorithm uses distance labels $\text{dist}(v, w)$, $v, w \in V$, which represent, if finite, the length of a path between nodes v and w. During the run of the algorithm $\text{dist}(v, w)$ is an upper bound on the shortest path length from node v to node w. Furthermore, let $\text{pred}(v, w)$ denote the predecessor index of the last node prior to node w in the tentative shortest path from v to w. Then, Algorithm 1 describes the pseudo-code for the Floyd-Warshall algorithm.

Algorithm 1: Floyd-Warshall

input : graph $G = (V, A)$ with arc lengths $c_{(v,w)}$ for $(v, w) \in A$ and no negative cycle

output: shortest path for every pair of nodes $(v, w) \in V \times V$, defined by predecessor labels $\text{pred}(v, w)$

```
1  for all node pairs (v, w) ∈ V × V do
2      dist(v, w) := ∞ and pred(v, w) := NULL
3  for all nodes v ∈ V do
4      dist(v, v) := 0
5  for each arc (v, w) ∈ A do
6      dist(v, w) := c_(v,w) and pred(v, w) := v
7  for each u ∈ V do
8      for each v ∈ V do
9          for each w ∈ V do
10             if dist(v, w) > dist(v, u) + dist(u, w) then
11                 dist(v, w) := dist(v, u) + dist(u, w)
12                 pred(v, w) := pred(u, w)
```

Using the predecessor indices, a shortest path from v to w can be reconstructed as follows: Starting at node w, the node $u = \text{pred}(v, w)$ is the node prior to w in the path. The node prior to u is given by $\text{pred}(v, u)$, and so on, until we reach node v. Algorithm 1 performs $\mathcal{O}(|V|^3)$ computations.

To the end of this section, we present a type of shortest path problem that arises frequently in practice.

Definition 2.10. *Consider a graph $G = (V, A)$ with two attributes c_a and t_a associated with each arc $a \in A$, representing arc lengths and a second resource which is consumed by traveling on arc $a \in A$, for example travel time. The* constrained shortest path problem *is to find the shortest path from a given source node s to a given sink node t, under the restriction that not more than T units of the second resource are used for the path.*

Hence, we need to solve the shortest path problem with the additional side constraint, that not more than T units of, e.g., time are consumed along the path. Constrained shortest path problems arise frequently in column generation approaches for vehicle routing problems (see Section 3.6). An overview of constrained shortest path problems and solution methods is given, for example, in Irnich and Desaulniers (2005).

2.2 Mixed-Integer Optimization

In this section, we introduce the main concepts of mixed-integer linear programming used in this thesis. Most parts of this section are based on the books Papadimitriou and Steiglitz (1998), Schrijver (1998) and Wolsey (1998). After defining a general optimization

problem and in particular mixed-integer linear programs, we discuss the basics of complexity theory in Section 2.2.1. An introduction to polyhedral theory follows in Section 2.2.2, where we present the important result, that under certain conditions the optimization of a linear function over a mixed-integer set described by a rational polyhedron equals the optimization over the convex hull of this set, i.e., the solution of a mixed-integer linear program becomes equivalent to the solution of a linear program. Next, we discuss exact solution techniques: Section 2.2.3 covers branch-and-bound algorithms, Section 2.2.4 deals with preprocessing techniques and Section 2.2.5 reviews branch-and-cut algorithms.

Throughout this chapter and this thesis, we use the following notation: To number vectors sequentially, we use upper indices, while we use lower indices to refer to different components of a vector. Given two vectors $x, y \in \mathbb{R}^n$, the symbols $\leq, \geq, <$ and $>$ are used to compare the vectors componentwise, i.e., $x \leq y$ if and only if $x_i \leq y_i$ for all $i = 1, \ldots, n$ and likewise for the other symbols. Moreover, given two sets A and B, we write $A \subset B$ if A is a proper subset of B, and $A \subseteq B$ if A is a subset of B.

Definition 2.11. *Let $X \subseteq \mathbb{R}^n$ and let $f\colon X \to \mathbb{R}$ be a function. Then,*

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & x \in X \end{aligned} \tag{2.1}$$

is called an optimization problem*. The function f is called* objective function *and the set X is called* feasible set*.*

In the following, we often assume that an optimization problem is feasible (i.e., $X \neq \emptyset$), bounded (i.e., there is a $\lambda \in \mathbb{R}$ such that for all $x \in X$ it holds that $f(x) \geq \lambda$) and that it has at least one optimal solution (i.e., a feasible solution $x \in X$ in which the minimum is attained). The optimization problems examined in this thesis belong to a specific class of optimization problems:

Definition 2.12. *Let $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ be vectors, let $A \in \mathbb{R}^{m \times n}$ be a matrix and let $p \in \{0, 1, \ldots, n\}$. A* mixed-integer linear program (MILP) *is an optimization problem of the form*

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & Ax \leq b \\ & x \geq 0 \\ & x \in \mathbb{R}^p \times \mathbb{Z}^{n-p}. \end{aligned} \tag{2.2}$$

The objective function $f(x) = c^\top x$ is a linear function and the feasible set X is equal to $\{x \in \mathbb{R}^p \times \mathbb{Z}^{n-p} \colon Ax \leq b,\, x \geq 0\}$. If $p = n$, (2.2) is called a *linear program (LP)*, and if $p = 0$, (2.2) is called an *integer linear program.*

2.2.1 Basic Notations of Complexity Theory

In this section, we introduce the concept of a decision problem and different complexity classes. This section is based on Papadimitriou and Steiglitz (1998) and Wolsey (1998). A detailed introduction to the subject is also given in Garey and Johnson (1979).

A decision problem is a question that can be answered by *yes* or *no*. In this section, instead of the optimization problem (2.1) we consider its corresponding decision problem.

To start with, we begin with the definition of the length of an input to an algorithm. The input I, e.g., a problem instance, is represented as a sequence (or *string*) of symbols (say binaries) on a computer. Then the length of the input is the number of binaries in the sequence:

Definition 2.13. *Given an input I to an algorithm, the* length of the input $L = L(I)$ *is the length of the binary representation of a "standard" representation of I.*

Definition 2.14. *Given a problem P, an algorithm A for the problem, and an input I, let $f_A(I)$ be the number of elementary calculations required to run algorithm A on the input. Then, $f_A^\star(\ell) = \sup_I\{f_A(I)\colon L(I) = \ell\}$ is the worst-case* running time *of the algorithm A for a given input length ℓ. An algorithm A is* polynomial *for a problem P if $f_A^\star(\ell) = \mathcal{O}(\ell^k)$ for some positive integer k.*

Now, we define the decision problem corresponding to an optimization problem.

Definition 2.15. *Given an optimization problem $\min\{f(x)\colon x \in X\}$ and a constant $k \in \mathbb{Z}$ the corresponding* decision problem *is:*

$$\textit{Is there a feasible solution } x \in X \textit{ with value } f(x) \leq k\textit{?}$$

An instance of the above decision problem consists of f, a "standard" representation of X and an integer k. A solution to a decision problem is the correct answer "yes" or "no".

In the following, we classify decision problems according to their complexity.

Definition 2.16. *The class $\mathcal{P}$ is the class of decision problems that can be solved by a polynomial-time algorithm.*

Next we define the class $\mathcal{NP}$. For a problem to be in $\mathcal{NP}$ we do not require that the decision problem can be answered in polynomial time. Rather than that, we require that given an instance I for which the decision problem is answered by *yes*, there exists a *concise* (i.e., bounded in length by a polynomial in $L(I)$) *certificate* for I, which can be checked in polynomial time for validity.

Definition 2.17. *We say that a decision problem P is in the class $\mathcal{NP}$ if there exists a polynomial q and an algorithm A (the* certificate-checking algorithm*) such that the following is true: The string I is a* yes *instance of P if and only if there exists a string of symbols, the certificate $c(I)$, with $L(c(I)) \leq q(L(I))$ and the property that A, if supplied with input I and $c(I)$, reaches the answer* yes *after at most $q(L(I))$ steps.*

Notice that $\mathcal{P}$ is a subset of $\mathcal{NP}$. To see this, suppose that A is a polynomial time algorithm for problem P. Given any instance I of P for which the decision problem is answered by *yes*, algorithm A will operate on I in polynomial time and answer *yes*. The record of this operation is a certificate $c(I)$, which can be checked in polynomial time by checking that it is a valid execution of A. So $\mathcal{P} \subseteq \mathcal{NP}$. We consider a second subclass of $\mathcal{NP}$ in the remainder of this subsection. First, we need the concept of polynomial reductions.

Definition 2.18. *Let P_1 and P_2 be two decision problems. We say that P_1* reduces in polynomial time *to P_2 if and only if there exists a polynomial-time algorithm A_1 for P_1 that uses several times, as a subroutine* at unit cost*, a (hypothetical) algorithm A_2 for P_2. We call A_1 a* polynomial-time *reduction of P_1 to P_2.*

By *at unit cost* we mean that algorithm A_2 is considered as a single instruction, taking unit time to execute.

Proposition 2.19. *If P_1 polynomially reduces to P_2 and if there is a polynomial-time algorithm for P_2, then there is a polynomial-time algorithm for P_1.*

Proof. See Papadimitriou and Steiglitz (1998). □

The following reduction will be of particular interest in the following.

Definition 2.20. *We say that a decision problem P_1* polynomially transforms *to another decision problem P_2, if, given any string I_1, we can construct a string I_2 within polynomial time in $L(I_1)$ such that I_1 is a* yes *instance of P_1 if and only if I_2 is a* yes *instance of P_2.*

Hence, a polynomial-time transformation is equivalent to a polynomial-time reduction with just one call of the subroutine for P_2.

Definition 2.21. *A decision problem $P \in \mathcal{NP}$ is said to be $\mathcal{NP}$-*complete *or in the class $\mathcal{NPC}$ if all other problems in $\mathcal{NP}$ polynomially transform to P.*

By Proposition 2.19, if a problem P is $\mathcal{NP}$-complete then the following holds: if there is an efficient algorithm for P then there is an efficient algorithm for every problem in $\mathcal{NP}$. A proof that $\mathcal{NPC} \neq \emptyset$ is given in Cook (1971) by showing that the satisfiability problem is in $\mathcal{NPC}$. Further $\mathcal{NP}$-complete problems based on that proof are listed in Garey and Johnson (1979). An important corollary is as follows:

Corollary 2.22. *If $\mathcal{P} \cap \mathcal{NPC} \neq \emptyset$, then $\mathcal{P} = \mathcal{NP}$.*

Proof. See Wolsey (1998). □

Unfortunately, the question whether $\mathcal{P} = \mathcal{NP}$ or $\mathcal{P} \neq \mathcal{NP}$ is still open. Optimization problems, for which the associated decision problem is in $\mathcal{NPC}$ are also called *$\mathcal{NP}$-hard*. We now turn the page from decision to optimization problems again.

2.2.2 Polyhedral Theory

In this section we take a closer look at the feasible set of an MILP, which (without integrality conditions) defines a polyhedron. We present the *decomposition theorem for polyhedra*, and show that for a rational polyhedron P, the convex hull of its mixed-integer points, i.e., the set $\text{conv}(P \cap (\mathbb{R}^p \times \mathbb{Z}^{n-p}))$, is again a polyhedron. This result is then used together with the *fundamental theorem of linear programming* to show that if an optimal solution of the MILP exists, solving the MILP is equivalent to solving an LP over the convex hull of the feasible set of the MILP. This section is mainly based on Wolsey (1998) and Schrijver (1998). The results presented in this chapter are analogous to the

well-known results for integer linear programs (see e.g. Schrijver (1998)) where instead of $\text{conv}(P \cap (\mathbb{R}^p \times \mathbb{Z}^{n-p}))$, the set $\text{conv}(P \cap \mathbb{Z}^n)$ is shown to be a polyhedron if P is rational. For the sake of completeness, we state them for the mixed-integer case here.

Definition 2.23. *A set P of vectors in $\mathbb{R}^n$ is called a* (convex) polyhedron*, if $P = \{x \in \mathbb{R}^n \colon Ax \leq b\}$ for some matrix $A \in \mathbb{R}^{m \times n}$ and vector $b \in \mathbb{R}^m$.*

Thus, a polyhedron is the intersection of finitely many *affine half-spaces*, i.e., of sets of the form $\{x \colon w^\top x \leq \delta\}$ for non-zero vectors $w \in \mathbb{R}^n$ and numbers $\delta \in \mathbb{R}$.

Definition 2.24. *The* convex hull *of a set $X \subseteq \mathbb{R}^n$ is the smallest convex set containing X and is denoted by* $\text{conv}(X)$*. Then:*

$$\text{conv}(X) = \left\{ \sum_{i=1}^{t} \lambda_i x^i \colon t \geq 1,\, x^1, \ldots, x^t \in X,\, \lambda_1, \ldots, \lambda_t \geq 0,\, \sum_{i=1}^{t} \lambda_i = 1 \right\}.$$

The constraint set of an MILP without integrality restrictions defines a polyhedron. This polyhedron is also called a *formulation* of the MILP:

Definition 2.25. *A polyhedron $P \subseteq \mathbb{R}^n$ is a* formulation *of a set $X \subseteq \mathbb{R}^p \times \mathbb{Z}^{n-p}$ if and only if $X = P \cap (\mathbb{R}^p \times \mathbb{Z}^{n-p})$.*

Note that, a formulation of the feasible set of the MILP (2.2) is given by the polyhedron $P = \{x \in \mathbb{R}^n \colon Ax \leq b,\, x \geq 0\}$.

Definition 2.26. *A formulation P for the set $X \subseteq \mathbb{R}^p \times \mathbb{Z}^{n-p}$ is said to be* compact*, if P is described by a number of linear inequalities which is polynomial in n.*

The next definition helps us to distinguish the quality of a formulation.

Definition 2.27. *Given a set $X \subseteq \mathbb{R}^p \times \mathbb{Z}^{n-p}$ and two formulations P_1 and P_2 of X, we say that P_1 is a* tighter *formulation than P_2 if $P_1 \subset P_2$.*

Next, we introduce the concept of valid inequalities.

Definition 2.28. *An inequality $\pi^\top x \leq \pi_0$ is a* valid inequality *for a set $X \subseteq \mathbb{R}^p \times \mathbb{Z}^{n-p}$ if $\pi^\top x \leq \pi_0$ for all $x \in X$.*

Definition 2.29. *If $\pi^\top x \leq \pi_0$ and $\mu^\top x \leq \mu_0$ are two valid inequalities for $X \subseteq (\mathbb{R}^p \times \mathbb{Z}^{n-p})_{\geq 0}$, then the inequality $\pi^\top x \leq \pi_0$* dominates *$\mu^\top x \leq \mu_0$ if there exists $\lambda > 0$ such that $\pi \geq \lambda\mu$ and $\pi_0 \leq \lambda\mu_0$, and $(\pi, \pi_0) \neq (\lambda\mu, \lambda\mu_0)$.*

Definition 2.30. *Let $X \subseteq (\mathbb{R}^p \times \mathbb{Z}^{n-p})_{\geq 0}$ and let P be a formulation of X. A valid inequality $\pi^\top x \leq \pi_0$ is* redundant *in the description of P, if there exist $k \geq 1$ valid inequalities $(\pi^i)^\top x \leq \pi_0^i$, $i \in \{1, \ldots, k\}$ for X and weights $\lambda_i > 0$, $i \in \{1, \ldots, k\}$ such that $\sum_{i=1}^k \lambda_i (\pi^i)^\top x \leq \sum_{i=1}^k \lambda_i \pi_0^i$ dominates $\pi^\top x \leq \pi_0$.*

The following principle is used to lift a valid inequality into a higher dimension, see, e.g., Conforti et al. (2014):

Definition 2.31. *Consider the feasible set* $X = \{x \in \mathbb{R}^p \times \mathbb{Z}^{n-p} \colon Ax \leq b,\, x \geq 0\}$ *of an MILP. Let* $N \coloneqq \{1, \ldots, n\}$ *and* $C \subset N$*, and let* $\sum_{j \in C} \alpha_j x_j \leq \beta$ *be a valid inequality on a subset of variables, namely* $\operatorname{conv}(X) \cap \{x \in \mathbb{R}^n \colon x_j = 0,\, j \in N \setminus C\}$*. An inequality* $\sum_{j=1}^{n} \alpha_j x_j \leq \beta$ *is called a* lifting *of* $\sum_{j \in C} \alpha_j x_j \leq \beta$ *if it is valid for* $\operatorname{conv}(X)$*.*

In the following, we introduce further convex sets such as *polytopes* and *cones*, and show that a polyhedron can be written as the Minkowski sum of a polytope and a cone.

Definition 2.32. *A set of vectors is a* (convex) polytope*, if it can be written as the convex hull of finitely many vectors.*

Definition 2.33. *A set* $C \neq \emptyset$ *in* $\mathbb{R}^n$ *is called a* convex cone *if* $\lambda x + \mu y \in C$ *for all* $x, y \in C$ *and* $\lambda, \mu \geq 0$*. A convex cone* C *is* polyhedral *if* $C = \{x \in \mathbb{R}^n \colon Ax \leq 0\}$ *for some matrix* $A \in \mathbb{R}^{m \times n}$*. The convex cone generated by a set* $\mathcal{X}$ *of vectors is denoted as* conic hull of $\mathcal{X}$*:*

$$\operatorname{cone}(\mathcal{X}) = \left\{ \sum_{i=1}^{t} \lambda_i x^i \colon \lambda_1, \ldots, \lambda_t \geq 0,\, x^1, \ldots, x^t \in \mathcal{X} \right\}.$$

A convex cone generated by a finite set of vectors $\mathcal{X} = \{x^1, \ldots, x^t\}$ *is called* finitely generated*.*

Remark 2.34. *The conic hull* $\operatorname{cone}(\mathcal{X})$ *generated by the set* $\mathcal{X} = \{x^1, \ldots, x^t\}$ *of vectors is the smallest convex cone containing* $\mathcal{X}$*.*

In this thesis, we usually refer to a convex cone simply as *cone*.

Theorem 2.35 (Farkas-Minkowski-Weyl)**.** *A cone is polyhedral if and only if it is finitely generated.*

Proof. See Schrijver (1998). □

Theorem 2.36 (Decomposition theorem for polyhedra)**.** *A set* $P \subseteq \mathbb{R}^n$ *is a polyhedron, if and only if* $P = Q + C$ *for some polytope* Q *and some polyhedral cone* C*.*

Proof. See Schrijver (1998). □

The next corollary follows immediately from Theorem 2.36.

Corollary 2.37. *A set* P *is a polytope if and only if* P *is a bounded polyhedron.*

Definition 2.38. *Let* $P = \{x \in \mathbb{R}^n \colon Ax \leq b\}$ *be a polyhedron,* $P \neq \emptyset$*. The* characteristic cone *of* P *is the polyhedral cone*

$$\mathit{char.cone}P \coloneqq \{y \in \mathbb{R}^n \colon x + y \in P \ \forall x \in P\} = \{y \colon Ay \leq 0\}.$$

It is straightforward, that if $P \neq \emptyset$ and $P = Q + C$ with Q a polytope and C a polyhedral cone, then $C = \operatorname{char.cone}P$.

Definition 2.39. *Let $P \subseteq \mathbb{R}^n$ be a polyhedron. Then, $P_I := \operatorname{conv}(P \cap (\mathbb{R}^p \times \mathbb{Z}^{n-p}))$ denotes the convex hull of the mixed-integer points in P.*

We will be specifically interested in the following polyhedra for the remaining part of this subsection:

Definition 2.40. *A polyhedron $P \subseteq \mathbb{R}^n$ is called* rational *if there exist $A \in \mathbb{Q}^{m \times n}$ and $b \in \mathbb{Q}^m$ such that $P = \{x \in \mathbb{R}^n \colon Ax \leq b\}$.*

Rational polyhedra have the nice property, that the convex hull of the mixed-integer points is again a polyhedron. This well-known result is first formulated below for the special case of a rational polytope and a rational polyhedral cone, see, e.g., Schrijver (1998). Before we extend it to the mixed-integer case, we need the following auxiliary result.

Lemma 2.41 (cf. Rockafellar (1997))**.** *Let $x^{i,1}, \dots, x^{i,t_i} \in \mathbb{R}^n$ for all $i \in \{1, \dots, k\}$. Then*

$$\operatorname{conv}\left(\bigcup_{i=1}^{k} \operatorname{conv}(x^{i,1}, \dots, x^{i,t_i})\right) = \operatorname{conv}\left(\bigcup_{i=1}^{k} \{x^{i,1}, \dots, x^{i,t_i}\}\right).$$

Proof. Let $x \in \operatorname{conv}\left(\bigcup_{i=1}^{k} \operatorname{conv}(x^{i,1}, \dots, x^{i,t_i})\right)$. Hence, $x = \sum_{j=1}^{t} \lambda_j y^j$ with $\sum_{j=1}^{t} \lambda_j = 1$, $\lambda_j \geq 0$ and $y^j \in \operatorname{conv}(x^{i,1}, \dots, x^{i,t_i})$ for some $i \in \{1, \dots, k\}$. We denote this i by i_j. Then

$$x = \sum_{j=1}^{t} \lambda_j \sum_{\ell=1}^{t_{i_j}} \mu_{i_j,\ell} x^{i_j,\ell}$$

with $\sum_{\ell=1}^{t_{i_j}} \mu_{i_j,\ell} = 1$ and $\mu_{i_j,\ell} \geq 0$ for all $i \in \{1, \dots, k\}$, $j \in \{1, \dots, t\}$. It follows, that $x \in \operatorname{conv}\left(\bigcup_{i=1}^{k} \{x^{i,1}, \dots, x^{i,t_i}\}\right)$, since

$$x = \sum_{j=1}^{t} \sum_{\ell=1}^{t_{i_j}} \lambda_j \mu_{i_j,\ell} x^{i_j,\ell}$$

with

$$\sum_{j=1}^{t} \sum_{\ell=1}^{t_{i_j}} \lambda_j \mu_{i_j,\ell} = \sum_{j=1}^{t} \lambda_j \sum_{\ell=1}^{t_{i_j}} \mu_{i_j,\ell} = \sum_{j=1}^{t} \lambda_j = 1$$

and $\lambda_j \mu_{i_j,\ell} \geq 0$ for all $i \in \{1, \dots, k\}$, $j \in \{1, \dots, t\}$.

For the reverse inclusion, let $x \in \operatorname{conv}\left(\bigcup_{i=1}^{k} \{x^{i,1}, \dots, x^{i,t_i}\}\right)$. Hence, $x = \sum_{j=1}^{t} \lambda_j y^j$ with $\sum_{j=1}^{t} \lambda_j = 1$, $\lambda_j \geq 0$ and $y^j \in \{x^{i,1}, \dots, x^{i,t_i}\}$ for some $i \in \{1, \dots, k\}$ for all $j \in \{1, \dots, t\}$. Thus, $y^j \in \operatorname{conv}(x^{i,1}, \dots, x^{i,t_i})$ for some $i \in \{1, \dots, k\}$. It follows that $x \in \operatorname{conv}\left(\bigcup_{i=1}^{k} \operatorname{conv}(x^{i,1}, \dots, x^{i,t_i})\right)$. □

The following three results are usually formulated explicitly for the integer case, i.e., instead of the set $P_I = \operatorname{conv}(P \cap (\mathbb{R}^p \times \mathbb{Z}^{n-p}))$ the set $\operatorname{conv}(P \cap \mathbb{Z}^n)$ is considered. We refer to Schrijver (1998) for the integer case and consider them for the mixed-integer case here.

Theorem 2.42. *If P is a rational polytope, then P_I is a polytope.*

Proof. Let π denote the projection on the last $n-p$ components (the integer components), i.e., $\pi(x_1, \ldots, x_n) = (x_{p+1}, \ldots, x_n)$. Since P is bounded, $\pi(P)$ contains only finitely many integer points. Let $\pi(P) = \{z^1, \ldots, z^k\}$, with $z^i \in \mathbb{Z}^{n-p}$ for $i \in \{1, \ldots, k\}$ be this set of integer points. Then

$$P \cap (\mathbb{R}^p \times \mathbb{Z}^{n-p}) = \bigcup_{i=1}^{k} P \cap (\mathbb{R}^p \times \{z^i\}).$$

Since P is a bounded polyhedron, for each $i \in \{1, \ldots, k\}$ the set $P \cap (\mathbb{R}^p \times \{z^i\})$ is also a bounded polyhedron, i.e., a polytope. Thus, there exist $x^{i,1}, \ldots, x^{i,t_i} \in \mathbb{R}^p \times \{z^i\}$ such that $P \cap (\mathbb{R}^p \times \{z^i\}) = \operatorname{conv}(x^{i,1}, \ldots, x^{i,t_i})$. Thus,

$$\operatorname{conv}(P \cap (\mathbb{R}^p \times \mathbb{Z}^{n-p})) = \operatorname{conv}\left(\bigcup_{i=1}^{k} \operatorname{conv}(x^{i,1}, \ldots, x^{i,t_i})\right) = \operatorname{conv}\left(\bigcup_{i=1}^{k} \{x^{i,1}, \ldots, x^{i,t_i}\}\right),$$

where we use Lemma 2.41 in the last equality, so P_I is the convex hull of finitely many points. □

Lemma 2.43. *If C is a rational polyhedral cone, then $C = C_I$.*

Proof. Since C is a convex cone, $C_I = \operatorname{conv}(C \cap (\mathbb{R}^p \times \mathbb{Z}^{n-p})) \subseteq \operatorname{conv}(C) = C$. For the reverse inclusion, note that since C is a rational polyhedral cone, it can be written as $C = \operatorname{cone}(x^1, \ldots, x^t)$ with $x^1, \ldots, x^t \in \mathbb{Q}^n$. By scaling, we can assume $x^1, \ldots, x^t \in \mathbb{Z}^n$. Let $c \in C$. Then, there exist vectors $\tilde{x}^1, \ldots, \tilde{x}^{\tilde{t}} \in C \cap (\mathbb{R}^p \times \mathbb{Z}^{n-p})$, such that $c \in \operatorname{conv}(\tilde{x}^1, \ldots, \tilde{x}^{\tilde{t}})$. Hence, $c \in C_I$. □

In fact, it even holds that for rational polyhedral cones $C = \operatorname{conv}(C \cap \mathbb{Z}^n)$ using a similar argumentation. Next, we prove the statement for general rational polyhedra.

Theorem 2.44. *For any rational polyhedron P, the set P_I is again a polyhedron.*

Proof. The proof for this result is based on the proof for the convex hull of integer points in P given in Schrijver (1998).

If $P = \emptyset$, then $P_I = \emptyset$, and hence the statement holds. We assume $P \neq \emptyset$ for the remaining part of the proof. Let $P = Q + C$, where Q is a polytope and C is the characteristic cone of P (see Theorem 2.36 and the remark after Definition 2.38). Since C is a rational polyhedral cone, $C = C_I = \operatorname{conv}(C \cap \mathbb{Z}^n)$. Using Theorem 2.35, let C be generated by the vectors $x^1, \ldots, x^t \in \mathbb{Z}^n$ and let B be

$$B \coloneqq \left\{ \sum_{i=1}^{t} \lambda_i x^i \colon 0 \leq \lambda_i \leq 1 \text{ for } i \in \{1, \ldots, t\} \right\}.$$

Hence, B is a polytope generated by the vectors $\{\sum_{i \in N} x^i \colon N \subseteq \{1, \ldots, t\}\}$ and the vector $(0, \ldots, 0)^\top$. We show that $P_I = (Q + B)_I + C$, which implies the theorem, as $Q + B$ is a rational polytope, and hence it follows from Lemma 2.42, that $(Q + B)_I$ is a polytope.

To show that $P_I \subseteq (Q+B)_I + C$, let $\rho \in P \cap (\mathbb{R}^p \times \mathbb{Z}^{n-p})$. Then, $\rho = q + c$, with $q \in Q$ and $c \in C$. Let $c = \sum_{i=1}^t \mu_i x^i$ with $\mu_1, \dots, \mu_t \geq 0$. Define $c' = \sum_{i=1}^t \lfloor \mu_i \rfloor x^i$ and $b = \sum_{i=1}^t (\mu_i - \lfloor \mu_i \rfloor) x^i$, then, $c = b + c'$ with $b \in B$ and $c' \in C$. Hence, $\rho = (q+b) + c'$ and $(q+b) \in (Q+B)_I$, as $q + b = \rho - c'$ and $\rho \in \mathbb{R}^p \times \mathbb{Z}^{n-p}$ and c' is integral. So $\rho \in (Q+B)_I + C$. Since $(Q+B)_I + C$ is a polyhedron (using Theorem 2.36), $(Q+B)_I + C$ is convex, hence $P_I = \operatorname{conv}(P \cap (\mathbb{R}^p \times \mathbb{Z}^{n-p})) \subseteq (Q+B)_I + C$.

The reverse inclusion follows from

$$\begin{aligned}
(Q+B)_I + C &\subseteq P_I + C \\
&= P_I + C_I \\
&= \operatorname{conv}(P \cap (\mathbb{R}^p \times \mathbb{Z}^{n-p})) + \operatorname{conv}(C \cap (\mathbb{R}^p \times \mathbb{Z}^{n-p})) \\
&\overset{\star}{=} \operatorname{conv}(p + c \colon p \in P \cap (\mathbb{R}^p \times \mathbb{Z}^{n-p}),\, c \in C \cap (\mathbb{R}^p \times \mathbb{Z}^{n-p})) \\
&\subseteq \operatorname{conv}(p + c \colon p \in P,\, c \in C,\, p + c \in \mathbb{R}^p \times \mathbb{Z}^{n-p}) \\
&= (P + C)_I = P_I,
\end{aligned}$$

where for the equality marked with "$\star$" we use the fact that for $V, W \subseteq \mathbb{R}^n$ it holds that $\operatorname{conv}(V + W) = \operatorname{conv}(V) + \operatorname{conv}(W)$. □

Note, that in general it is not a restriction to assume rational data, since irrational numbers can not be represented on a computer such that they have to be converted into rational numbers. We are going to combine the previous result with another important theorem, the *fundamental theorem of linear programming.* First, we need the following definition given, e.g., in Narici and Beckenstein (1985).

Definition 2.45. *A point x of a convex set $K \subseteq \mathbb{R}^n$ is an* extreme point *of K if there do not exist $\tilde{y}, \bar{y} \in K$, $\tilde{y} \neq \bar{y}$ and $0 < \lambda < 1$, such that $x = \lambda \tilde{y} + (1-\lambda)\bar{y}$.*

Theorem 2.46 (Fundamental theorem of linear programming)**.** *If $P = \{x \in \mathbb{R}^n \colon Ax \leq b,\, x \geq 0\} \neq \emptyset$, and $\min\{c^\top x : x \in P\}$ is finite, then there is an optimal solution that is an extreme point.*

Proof. Cf. Nemhauser and Wolsey (1988). □

The following well-known result is given for example in Wolsey (1998). We give a proof for the sake of completeness.

Proposition 2.47. *Let $X \subseteq \mathbb{R}^n$. The extreme points of* $\operatorname{conv}(X)$ *all lie in X.*

Proof. Let $x \in \operatorname{conv}(X)$ be an extreme point of $\operatorname{conv}(X)$. Since $x \in \operatorname{conv}(X)$, there exist $x^i \in X$, $\lambda_i \geq 0$ for $i \in \{1, \dots, t\}$ with $\sum_{i=1}^t \lambda_i = 1$ such that $x = \sum_{i=1}^t \lambda_i x^i$. If $t = 1$, then $x \in X$. So we assume the contrary $t \geq 2$. Moreover, we assume that $\lambda_i > 0$, otherwise we can choose t smaller. Then,

$$x = \lambda_1 x^1 + (1 - \lambda_1) \sum_{i=2}^{t} \frac{\lambda_i}{1 - \lambda_1} x^i.$$

Since $\sum_{i=2}^t \lambda_i = 1 - \lambda_1$, it holds that $x' \coloneqq \sum_{i=2}^t \frac{\lambda_i}{1-\lambda_1} x^i \in \operatorname{conv}(X)$. Hence, $x = \lambda_1 x^1 + (1-\lambda_1)x'$ with $0 < \lambda_1 < 1$ and $x^1, x' \in \operatorname{conv}(X)$. If $x^1 \neq x'$, this is a contradiction to x being an extreme point. If $x^1 = x'$, $x = x^1 \in X$. □

With the previous results of this section, we have worked towards the following theorem.

Theorem 2.48. *Let $P = \{x \in \mathbb{R}^n \colon Ax \leq b,\, x \geq 0\}$ be a rational polyhedron, such that $X = P \cap (\mathbb{R}^p \times \mathbb{Z}^{n-p}) \neq \emptyset$ and $\min\{c^\top x \colon x \in X\}$ is finite. Then,*

$$\min\{c^\top x \colon x \in X\} = \min\{c^\top x \colon x \in \operatorname{conv}(X)\}.$$

Proof. Since $X \subseteq \operatorname{conv}(X)$, $\min\{c^\top x \colon x \in X\} \geq \min\{c^\top x \colon x \in \operatorname{conv}(X)\}$. From Theorem 2.44, we know that $\operatorname{conv}(X) = P_I$ is a polyhedron. By Theorem 2.46 there exists an extreme point x' of $conv(X)$, such that $\min\{c^\top x \colon x \in \operatorname{conv}(X)\} = c^\top x' \geq \min\{c^\top x \colon x \in X\}$, where the last inequality follows from Proposition 2.47. □

Hence, assuming rationality of the data, solving the MILP $\min\{c^\top x \colon x \in X\}$ with $X = \{x \in \mathbb{R}^p \times \mathbb{Z}^{n-p} \colon Ax \leq b, x \geq 0\}$ is equivalent to solving the LP $\min\{c^\top x \colon x \in \operatorname{conv}(X)\}$. Unfortunately, in general, it is not easy to find a description of $\operatorname{conv}(X)$ as there might be an (exponential) number of linear inequalities needed (see, e.g., Rothvoß, 2012). The advantage of LPs is that they are solvable in polynomial time, while this is in general not true for MILPs. In most practical applications the simplex method (based on Dantzig, 1963) is used to solve LPs efficiently, although there exist examples for which it has exponential running time in the worst case.

An upper bound on the objective function value of an MILP is given by any feasible solution $x \in X$. Usually heuristics are used to quickly find feasible solutions. Upper bounds (of a minimization problem) are also referred to as *primal bounds*. To compute lower or *dual* bounds, the most important strategy is to use a *relaxation*, see, e.g., Wolsey (1998).

Definition 2.49. *Let $f \colon T \to \mathbb{R}$ and $g \colon X \to \mathbb{R}$ be two functions. An optimization problem* (RP) $z^{rel} = \min\{f(x) \colon x \in T \subseteq \mathbb{R}^n\}$ *is a* relaxation *of* (P) $z = \min\{g(x) \colon x \in X \subseteq \mathbb{R}^n\}$ *if:*

- *$X \subseteq T$, and*
- *$f(x) \leq g(x)$ for all $x \in X$.*

Proposition 2.50. *If* (RP) *is a relaxation of* (P)*, $z^{rel} \leq z$.*

Proof. If $x^\star$ is an optimal solution of (P), then $x^\star \in X \subseteq T$ and $z = g(x^\star) \geq f(x^\star)$. As $x^\star \in T$, $f(x^\star)$ is an upper bound on z^{rel}, and so $z \geq f(x^\star) \geq z^{\text{rel}}$. □

A common relaxation for MILPs is the following:

Definition 2.51. *The* linear programming relaxation *of the MILP* (2.2) *is given by the LP*

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & Ax \leq b \\ & x \geq 0 \\ & x \in \mathbb{R}^n. \end{aligned}$$

In the next section, we present the concept of *branch-and-bound* algorithms, which involve the computation of lower and upper bounds to identify subsets of X, in which an optimal solution may exist, and which constitute a foundation for exact solution methods for MILPs.

2.2.3 Branch-and-Bound

A *branch-and-bound* (B&B) algorithm is an exact solution method for MILPs, which relies on two main principles: First, the iterative decomposition of the feasible set into smaller subsets, representing subproblems of the original optimization problem. The division of the original problem into smaller subproblems is typically represented by an enumeration tree. Second, the application of pruning rules to eliminate parts of the tree in which a better solution can not be found. This section is based on Wolsey (1998).

We consider the optimization problem

$$z = \min\{c^\top x \colon x \in X\}$$

with $c \in \mathbb{R}^n$ and $X \subseteq \mathbb{R}^p \times \mathbb{Z}^{n-p}$.

Proposition 2.52. *Let $X = X_1 \cup \ldots \cup X_H$ be a decomposition of X into smaller sets, and let $z^h = \min\{c^\top x \colon x \in X_h\}$ for $h \in \{1, \ldots, H\}$. Then $z = \min\{z^h \colon h \in \{1, \ldots, H\}\}$.*

The decomposition of a problem into smaller subproblems, which are easier to solve, is also referred to as *divide and conquer*. Typically, an enumeration tree is used to represent a divide and conquer approach. An example of an enumeration tree for $X \subseteq \{0,1\}^3$ is shown in Figure 2.1. Since a complete enumeration is impractical for most integer programs, pruning rules are used to cut parts of the tree. One of these rules is based on the computation of upper and lower bounds.

Proposition 2.53. *Let $X = X_1 \cup \ldots \cup X_H$ be a decomposition of X into smaller sets, and let $z^h = \min\{c^\top x \colon x \in X_h\}$ for $h \in \{1, \ldots, H\}$. Let $\underline{z}^h$ denote a lower bound, and $\bar{z}^h$ denote an upper bound on z^h. Then $\bar{z}^h = c^\top \bar{x}^h$ with $\bar{x}^h \in X_h$ the current best solution in subproblem X_h. Moreover, $\bar{z} = \min_h \bar{z}^h$ is an upper bound on z.*

We give an example for *pruning by bounds*. In Figure 2.2 a decomposition of a feasible set X with lower and upper bounds on the subproblems is shown. Here, $\bar{z} = \min\{20, 26\} = 20$ and $\underline{z} = \min\{18, 21\} = 18$. Hence, the optimal value is at most 20. Since $\underline{z}^2 = 21$, an optimal solution can not lie in X_2 and the branch X_2 of the tree can be pruned.

In total, there are three cases where a node of the enumeration tree can be pruned:

- Pruning by bounds: $\underline{z}^h \geq \bar{z}$.
- Pruning by optimality: $\bar{z}^h = \underline{z}^h$, i.e., $z^h = \min\{c^\top x \colon x \in X_h\}$ has been solved.
- Pruning by infeasibility: $X_h = \emptyset$.

In practice, upper bounds are usually obtained by feasible solutions and lower bounds are obtained by relaxations. A flowchart outlining the steps of a branch-and-bound algorithm using the LP relaxation to compute lower bounds and binary variable branching is

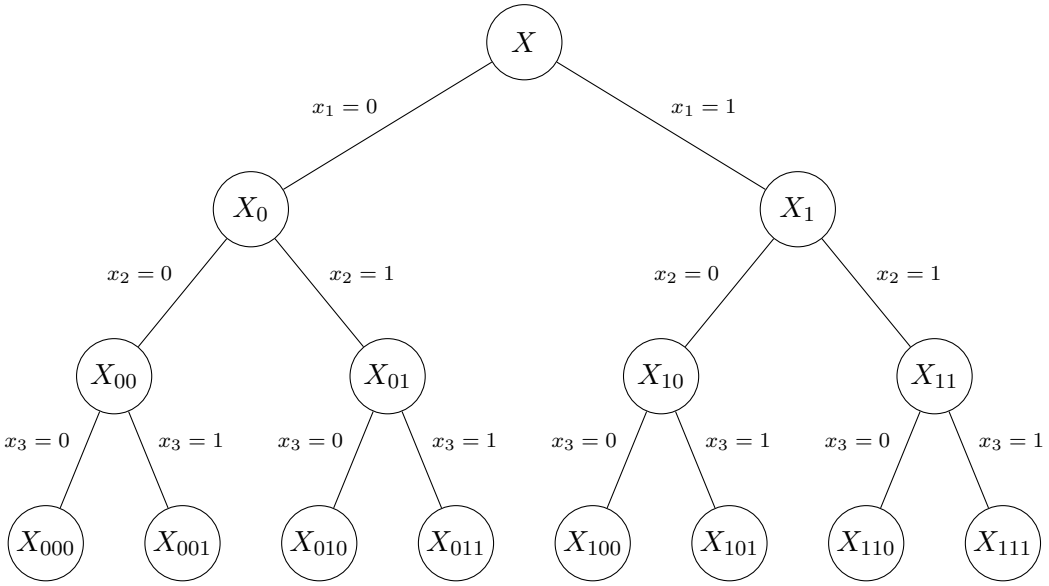


Figure 2.1: Binary enumeration tree.

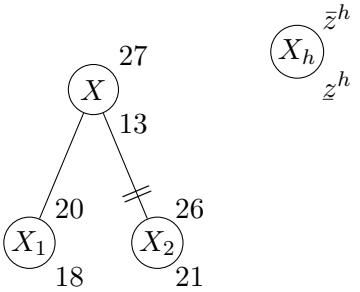


Figure 2.2: Pruning by bounds.

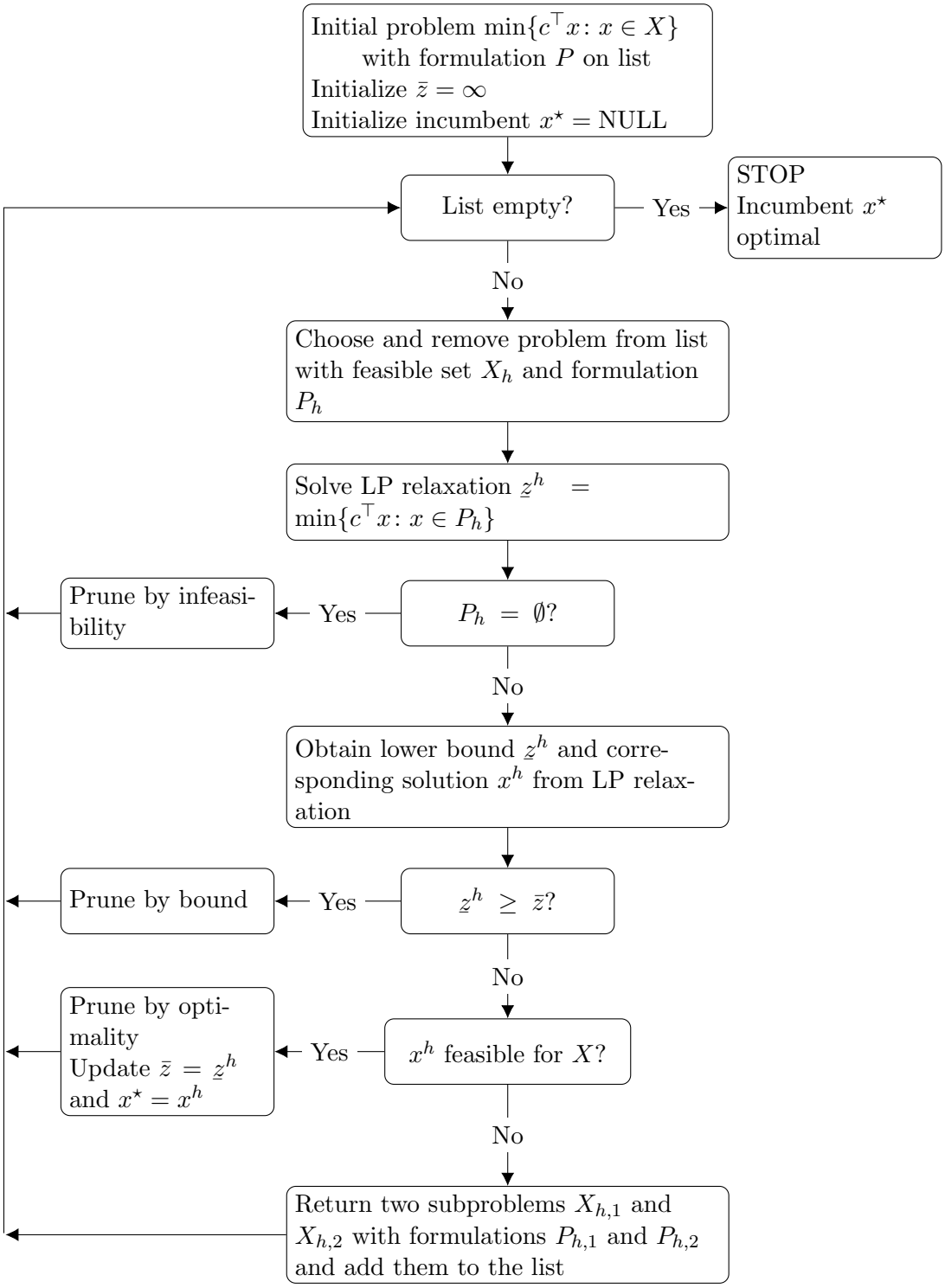


Figure 2.3: Example of a branch-and-bound flow chart.

presented in Figure 2.3. In an implementation of a branch-and-bound algorithm further details have to be specified, e.g., the order in which the branch-and-bound nodes are examined (e.g., best-first, depth-first, or breadth-first search), which relaxations to choose (e.g., linear programming relaxation, Lagrangian relaxation) or the branching, i.e., the division of the problem into subproblems. For example, if the solution $x^\star$ of the current subproblem is fractional in $x_j^\star$ with $p < j \leq n$, a common technique is to create two new subproblems with feasible sets $X \cap \{x\colon x_j \leq \lfloor x_j^\star \rfloor\}$ and $X \cap \{x\colon x_j \geq \lceil x_j^\star \rceil\}$. Note that B&B is in general not finite if the formulation P is not bounded. For a recent discussion of branch-and-bound algorithms we refer to Morrison et al. (2016).

Solvers for MILPs are often based on branch-and-bound algorithms. These are enriched by other methods some of which we will describe below: Preprocessing (see Section 2.2.4) is an important step to reduce the size of the MILP in order to strengthen the initial relaxation. A branch-and-cut algorithm (see Section 2.2.5) is an algorithm based on branch-and-bound in which *cuts*, i.e., valid inequalities, are used to cut-off a solution of a relaxation of the problem which is not feasible due to integrality violations. Typically, a range of cuts tailored to different problem types are implemented in the solver. Furthermore, heuristics are invoked to quickly find feasible solutions and can speed the proof of optimality of a given incumbent by providing upper bounds on the optimal objective value. For example the *feasibility pump* is a heuristic that is able to find an initial solution even in certain very hard MILPs, see Fischetti et al. (2005). Popular solvers are the commercial solvers CPLEX[1], Gurobi[2] and LocalSolver[3], or the non-commercial solvers SCIP[4], GLPK[5] and OR-Tools[6]. For more details, we refer to the respective solvers' manuals.

2.2.4 Preprocessing

Before solving an MILP using, e.g., a branch-and-bound algorithm within a solver, different strategies may be applied to make the optimization process easier. These methods are generally summarized under the term *preprocessing*. Several common strategies, described e.g. in Wolsey (1998), are discussed below.

First, heuristics may be used to construct an initial feasible solution. Passing a feasible solution as a starting solution to a solver may be advantageous to guide the remaining search process. An initial solution often provides a good upper bound on the objective function value: the better the initial solution, the more likely pruning is possible. Moreover, the time the solver needs to find an initial feasible solution can be saved.

Another important step in speeding up the solution process may be the addition of valid inequalities to the problem formulation to tighten it. The idea of the a-priori addition of valid inequalities $Qx \leq q$ for some $Q \in \mathbb{R}^{m' \times n}$, $q \in \mathbb{R}^{m'}$ to the initial formulation $P = \{x\colon Ax \leq b, x \geq 0\}$ with $X = P \cap (\mathbb{R}^p \times \mathbb{Z}^{n-p})$ is to obtain a tighter formulation $P' = \{x\colon Ax \leq b, Qx \leq q, x \geq 0\}$ with $\operatorname{conv}(P') \subset \operatorname{conv}(P)$. If the valid inequalities

[1] `https://www.ibm.com/docs/en/icos`
[2] `https://www.gurobi.com/solutions/gurobi-optimizer/`
[3] `https://www.localsolver.com/`
[4] `https://scipopt.org/`
[5] `https://www.gnu.org/software/glpk/`
[6] `https://developers.google.com/optimization`

are well chosen, this can significantly improve the performance of the branch-and-bound algorithm. However, the addition of a large number of valid inequalities to the MILP can also have undesired effects, and lead to a slow-down, because either the solver is not capable of processing as many constraints, or the MILP becomes too big to be solved in a reasonable amount of time. Hence, one has to consider the trade-off between the savings in computational time due to a tighter formulation and the increase in computational time due to the addition of a large number of constraints.

Furthermore, solvers usually check the MILP for redundant constraints and variables, and tighten bounds on variables if possible. This is formalized in the next proposition.

Proposition 2.54. *Consider the set*

$$X = \left\{ x \in \mathbb{R}^{n+1} \colon a_0 x_0 + \sum_{j=1}^{n} a_j x_j \leq b,\ \ell_j \leq x_j \leq u_j,\ \text{for } j \in \{0, 1, \dots, n\} \right\}.$$

- *Bounds on variables. If $a_0 > 0$, then*

$$x_0 \leq \frac{1}{a_0} \left(b - \sum_{\substack{j=1 \\ j\colon a_j > 0}}^{n} a_j \ell_j - \sum_{\substack{j=1 \\ j\colon a_j < 0}}^{n} a_j u_j \right),$$

  and if $a_0 < 0$, then

$$x_0 \geq \frac{1}{a_0} \left(b - \sum_{\substack{j=1 \\ j\colon a_j > 0}}^{n} a_j \ell_j - \sum_{\substack{j=1 \\ j\colon a_j < 0}}^{n} a_j u_j \right).$$

- *Redundancy. The constraint $a_0 x_0 + \sum_{j=1}^{n} a_j x_j \leq b$ is redundant if*

$$\sum_{\substack{j=0 \\ j\colon a_j > 0}}^{n} a_j u_j + \sum_{\substack{j=0 \\ j\colon a_j < 0}}^{n} a_j \ell_j \leq b.$$

- *Infeasibility. $S = \emptyset$ if*

$$\sum_{\substack{j=0 \\ j\colon a_j > 0}}^{n} a_j l_j + \sum_{\substack{j=0 \\ j\colon a_j < 0}}^{n} a_j u_j > b.$$

- *Variable fixing. For a minimization problem $\min\{c^\top x \colon x \in \mathbb{R}^n,\ Ax \leq b,\ l \leq x \leq u\}$, if $a_{ij} \geq 0$ for all $i \in \{1, \dots, m\}$ and $c_j > 0$, then $x_j = \ell_j$. Conversely, if $a_{ij} \leq 0$ for all $i \in \{1, \dots, m\}$ and $c_j < 0$, then $x_j = u_j$.*

Considering an MILP, the bounds on integer variables x_j can further be tightened as $\lceil \ell_j \rceil \leq x_j \leq \lfloor u_j \rfloor$. In the next subsection, we consider a branch-and-bound-based algorithm to solve MILP, which involves the addition of valid inequalities in the course of the algorithm.

2.2.5 Branch-and-Cut

A *branch-and-cut (B&C) algorithm* is a combination of a B&B and a cutting plane algorithm. In a B&C algorithm, whenever the solution x^h of the current subproblem is not feasible for the original problem, a so called *cutting plane algorithm* looks for valid inequalities, which are violated by x^h. These are then added to the subproblem and it is solved again, so that a new solution is hopefully "less fractional". In this section, we first describe a general cutting plane algorithm. After that, we present a flowchart of a B&C algorithm. This section is based on Wolsey (1998).

Cutting plane algorithm Suppose we know a family $\mathcal{F}$ of valid inequalities $\pi^\top x \leq \pi_0$, $(\pi, \pi_0) \in \mathcal{F}$ for X, but there are too many inequalities to add them all to a problem formulation, for example during preprocessing or in a node of a branch-and-cut tree. A cutting plane algorithm selects a subset of inequalities from $\mathcal{F}$. In every iteration of a cutting plane algorithm, the following problem is solved to select an inequality.

Definition 2.55. *The* separation problem *associated with* (2.2) *is the problem: Given* $x^\star \in \mathbb{R}^n$*, is* $x^\star \in \operatorname{conv}(X)$*? If not, find an inequality* $\pi^\top x \leq \pi_0$ *satisfied by all points in* X*, but violated by* $x^\star$*.*

Algorithm 2 describes a basic cutting plane algorithm for an MILP (2.2) with formulation $P = \{x \colon Ax \leq b,\, x \geq 0\}$.

Algorithm 2: Basic Cutting Plane Algorithm

input : An MILP (2.2) with formulation P, bound on the number of iterations I, family of valid inequalities $\mathcal{F}$
output: An optimal solution for (2.2) or a tighter formulation P^i

```
init
    i = 1 and P^1 = P
while i ≤ I do
    Solve the linear program min{c^⊤ x : x ∈ P^i}
    Let x^i be an optimal solution
    if x^i ∈ R^p × Z^{n−p} then
        STOP, x^i is an optimal solution for (2.2)
    else
        Solve the separation problem for x^i and the family F
        if there is an inequality (π^i, π_0^i) ∈ F with (π^i)^⊤ x^i > π_0^i then
            P^{i+1} = P^i ∩ {x : (π^i)^⊤ x ≤ π_0^i}
            i = i + 1
        else
            STOP
```

If the algorithm terminates without finding an optimal solution for (2.2), the formulation $P^i = P \cap \{x \colon (\pi^k)^\top x \leq \pi_0^k,\, k \in \{1, \ldots, i\}\}$ is an improved formulation. Note that, in

practice, it is often better to add more than one cut in every iteration.

Branch-and-cut algorithm The integration of a cutting plane algorithm into a B&B framework yields a B&C algorithm. Figure 2.4 illustrates the procedure of a B&C algorithm.

In contrast to a B&B algorithm, where the focus lies on solving many nodes fast to quickly narrow down the search region, in a B&C algorithm the focus is on finding good (i.e., close to the true optimal solution) solutions in a smaller number of nodes of the tree by tightening the formulations of the examined nodes. To assess the quality of an incumbent found during the execution of a B&B or B&C algorithm, the following measure is commonly used.

Definition 2.56. *The* relative gap *is defined as the relative difference between the lower and upper bound of the optimal objective function value:*

$$\textit{relative gap} = \frac{\bar{z} - \underline{z}}{\bar{z}}.$$

We usually refer to the relative gap as *gap*.

The MILPs in this thesis are solved using a state-of-the-art commercial optimization software like CPLEX that implements a branch-and-cut algorithm.

2.3 Multi-Objective Optimization

This section gives a brief summary of the concepts of multi-objective optimization used in this thesis and is based on Ehrgott (2005). First we define different notions of optimality. In the second part of this section, we mainly focus on two scalarizations: the weighted-sum method and the ε-constraint method.

In the following, we use the symbol $\leqslant$ which is often used in the field of multi-objective optimization: Let $x, y \in \mathbb{R}^q$ be two vectors, then $x \leqslant y$ if and only if $x \leq y$ and $x \neq y$.

Definition 2.57. *Let $X \subseteq \mathbb{R}^n$ and $f = (f_1, \ldots, f_q) \colon X \to \mathbb{R}^q$ a function. Then,*

$$\begin{aligned} \min \quad & f(x) = (f_1(x), \ldots, f_q(x))^\top \\ \text{s.t.} \quad & x \in X \end{aligned} \tag{2.3}$$

is called a multi-objective optimization problem (MOP)*. Here, f is an* objective function vector *comprised of q optimization* criteria *$f_i \colon X \to \mathbb{R}$, $i \in \{1, \ldots, q\}$. The set X is called the feasible set. The image of the feasible set under the mapping f is denoted as $Y \coloneqq f(X)$.*

If $q = 2$, we refer to (2.3) as a *bi-objective optimization problem*. The following definitions provide us with notions of optimality in multi-objective optimization.

Definition 2.58. *A feasible solution $x^\star \in X$ is called* efficient *or* Pareto optimal *if there is no other $x \in X$ such that $f(x) \leqslant f(x^\star)$. If $x^\star$ is efficient, then $f(x^\star)$ is called* non-dominated*. If $f(x^1) \leqslant f(x^2)$ with $x^1, x^2 \in X$ we say that x^1* dominates *x^2 and $f(x^1)$*

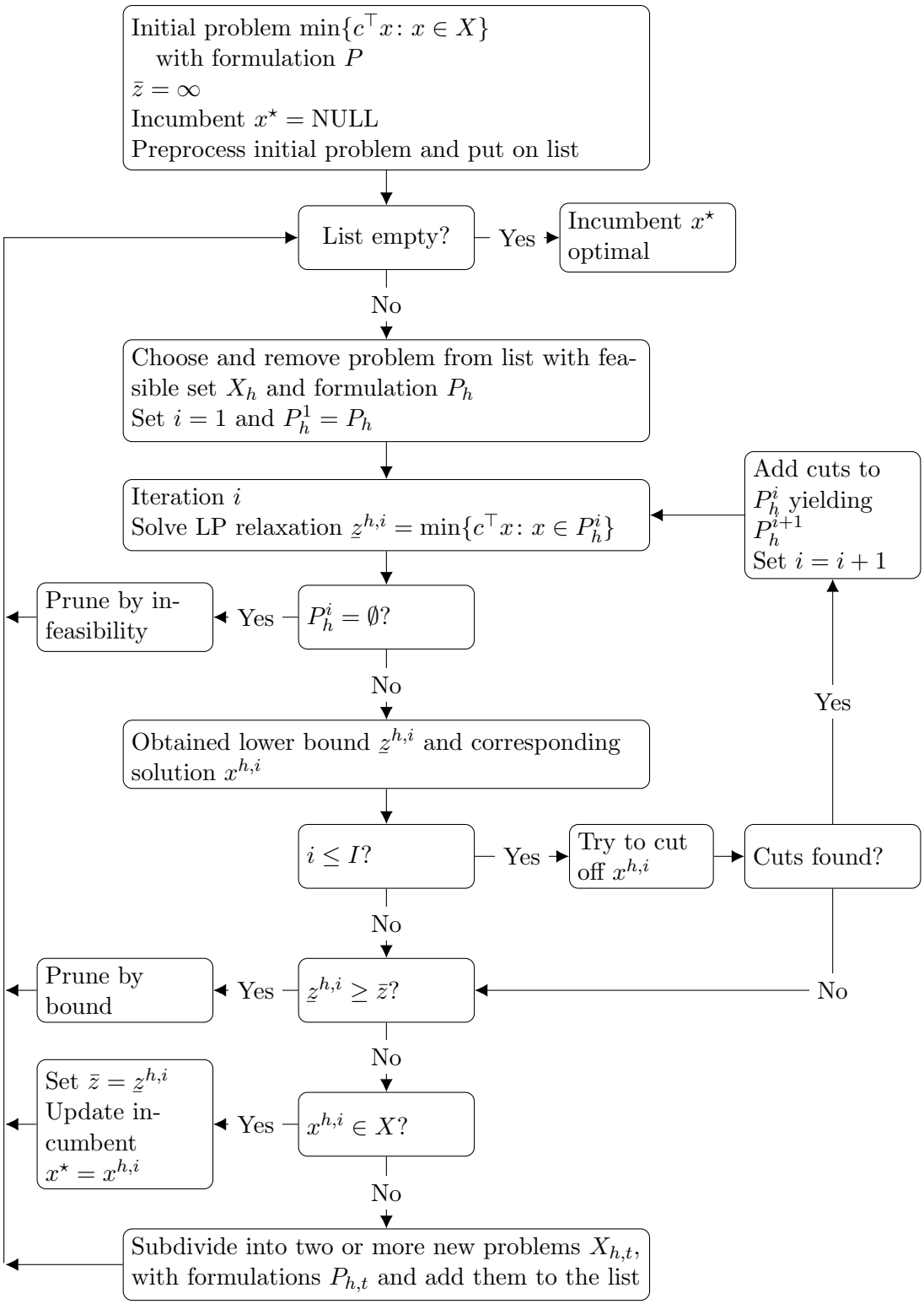


Figure 2.4: Example of a branch-and-cut flow chart.

dominates $f(x^2)$. *The set of all efficient solutions* $x^\star \in X$ *is denoted by* X_E *and called the* efficient set. *The set of all non-dominated points* $y^\star = f(x^\star) \in Y$, *where* $x^\star \in X_E$, *is denoted by* Y_N *and called the* non-dominated set.

Definition 2.59. *A feasible solution* $x^\star \in X$ *is called* weakly efficient *if there is no* $x \in X$ *such that* $f(x) < f(x^\star)$. *The corresponding point* $y^\star = f(x^\star)$ *is called* weakly non-dominated. *The weakly efficient set and the weakly non-dominated set are denoted by* X_{wE} *and* Y_{wN}, *respectively.*

Another concept of optimality arises from applications of multi-objective optimization, where an ordering of the optimization criteria in terms of their importance is given. For example, in the context of ridepooling, a service provider might decide that the reduction of routing costs or driven kilometers is the most important optimization goal, and only for the respective optimal solution the time users wait to be picked up should be minimized as a secondary goal. This so called *lexicographic optimization problem* is written as

$$\begin{aligned} \text{lexmin} \quad & f(x) = (f_1(x), \dots, f_q(x))^\top \\ \text{s.t.} \quad & x \in X. \end{aligned}$$

We now formally define the notion of lexicographic optimality.

Definition 2.60. *A feasible solution* $x^\star \in X$ *is* lexicographically optimal *or a* lexicographic solution *if there is no* $x \in X$ *such that* $f(x) <_{lex} f(x^\star)$, *where* $y^1 <_{lex} y^2$ *for* $y^1, y^2 \in \mathbb{R}^q$, *if* $y^1 \neq y^2$ *and* $y^1_k < y^2_k$ *with* $k = \min\{i \in \{1, \dots, q\} \colon y^1_i \neq y^2_i\}$.

In other words, $x^\star \in X$ is lexicographically optimal, if $f(x^\star) \leq_{\text{lex}} f(x)$ for all $x \in X$, i.e., $f(x^\star) = f(x)$ or $f_k(x^\star) < f_k(x)$ with $k = \min\{i \in \{1, \dots, q\} \colon f_i(x^\star) \neq f_i(x)\}$. While the concept of Pareto optimality stresses the independent and equitable optimization of all objectives, the concept of lexicographic optimality explicitly ranks the objective functions in the sense that criterion f_i is only considered when $f_1, \dots, f_{i-1}$ have been optimized. We have the following relation between efficient and lexicographically optimal solutions:

Lemma 2.61. *Let* $x^\star \in X$ *such that* $f(x^\star) \leq_{lex} f(x)$ *for all* $x \in X$. *Then* $x^\star \in X_E$.

Proof. A proof is given in Ehrgott (2005). □

2.3.1 Scalarizations

A common approach to solve multi-objective optimization problems is by *scalarization*, which is the formulation of one or a series of single-objective optimization problems related to the multi-objective optimization problem. In a scalarized problem, the objective function is a real-valued function, typically depending on the optimization criteria and/or the constraints of (2.3) and/or additional variables and parameters. The feasible set may be restricted by additional constraints corresponding to the optimization criteria of the MOP. A well-known scalarization technique is the following:

Definition 2.62. *The* weighted-sum scalarization *of the MOP* (2.3) *is given by*

$$\begin{aligned} \min \quad & \sum_{i=1}^{q} \omega_i f_i(x) \\ \text{s.t.} \quad & x \in X, \end{aligned} \tag{2.4}$$

where $\omega \geqslant 0$ is a non-negative weighting vector.

Definition 2.63. *Let $Y \subseteq \mathbb{R}^q$ and $\omega \geqslant 0$. We denote by*

$$\mathcal{S}(\omega, Y) \coloneqq \{y^\star \in Y : \omega^\top y^\star = \min_{y \in Y} \omega^\top y\}$$

the set of optimal points of Y with respect to ω, and define:

$$\mathcal{S}(Y) \coloneqq \bigcup_{\omega > 0} \mathcal{S}(\omega, Y) = \bigcup_{\{\omega > 0 : \|\omega\|_1 = 1\}} \mathcal{S}(\omega, Y)$$

and

$$\mathcal{S}_0(Y) \coloneqq \bigcup_{\omega \geqslant 0} \mathcal{S}(\omega, Y) = \bigcup_{\{\omega \geqslant 0 :\ \|\omega\|_1 = 1\}} \mathcal{S}(\omega, Y).$$

Note that, the assumption $\|\omega\|_1 = 1$ is not restrictive, since it normalizes the weight, but does not affect the set of optimal outcome vectors. Using the previous definition, we get the following important results:

Theorem 2.64. *For any set $Y \subseteq \mathbb{R}^q$ we have $\mathcal{S}_0(Y) \subseteq Y_{wN}$.*

Proof. See Ehrgott (2005). □

Theorem 2.65. *Let $Y \subseteq \mathbb{R}^q$. Then $\mathcal{S}(Y) \subseteq Y_N$.*

Proof. A proof is given in Ehrgott (2005). □

Hence, the optimal points of a weighted-sum scalarization with strictly positive weights are non-dominated for (2.3), while they are weakly non-dominated if they are generated by non-negative weights $\omega \geqslant 0$. Under additional convexity assumptions, also the reverse inclusion can be shown:

Definition 2.66. *A set $Y \subseteq \mathbb{R}^q$ is called $\mathbb{R}^q_{\leq}$-convex if $Y + \mathbb{R}^q_{\leq}$ is convex.*

Theorem 2.67. *If Y is $\mathbb{R}^q_{\leq}$-convex, then $Y_{wN} = \mathcal{S}_0(Y)$.*

Proof. See Ehrgott (2005). □

Theorem 2.68. *If Y is an $\mathbb{R}^q_{\leq}$-convex set, then $Y_N \subseteq \mathcal{S}_0(Y)$.*

Proof. For a proof we refer to Ehrgott (2005). □

Hence, if Y is $\mathbb{R}^q_{\geq}$-convex, then the set of non-dominated points can be generated by weighted-sum scalarizations . Here, an appropriate selection of weighting vectors is needed. A problem that occurs within this context is that an even distribution of weights does not necessarily lead to an even distribution of non-dominated outcome vectors, see for example Das and Dennis (1997). In general, if Y is not $\mathbb{R}^q_{\geq}$-convex, weighted-sum scalarizations with non-negative weights $\omega \geqslant 0$ can generate only *supported non-dominated points*, that is, non-dominated points on the convex hull of $Y + \mathbb{R}^q_{\geq}$. We now introduce a scalarization technique that can generate the entire non-dominated set even if Y is not $\mathbb{R}^q_{\geq}$-convex.

For the ε-constraint scalarization, one of the optimization criteria is kept as an objective function of the new single-objective problem and the remaining criteria are added to the feasible set as additional constraints:

Definition 2.69. *Let $j \in \{1, \ldots, q\}$ and $\varepsilon \in \mathbb{R}^q$. The* ε-constraint scalarization *w.r.t. j and ε of the MOP* (2.3) *is given by*

$$\begin{aligned} \min \quad & f_j(x) \\ \text{s. t.} \quad & f_i(x) \leq \varepsilon_i, \\ & x \in X. \end{aligned} \tag{2.5}$$

For optimal solutions of ε-constraint scalarizations, we can only guarantee that they are weakly efficient:

Proposition 2.70. *Let $x^\star$ be an optimal solution of* (2.5) *for some $j \in \{1, \ldots, q\}$ and $\varepsilon \in \mathbb{R}^q$. Then $x^\star$ is weakly efficient.*

Proof. See Ehrgott (2005). □

Under appropriate assumptions, the optimal solutions of ε-constraint scalarizations can be shown to be efficient:

Theorem 2.71. *The feasible solution $x^\star \in X$ is efficient if and only if there exists $\varepsilon \in \mathbb{R}^q$ such that $x^\star$ is an optimal solution of* (2.5) *for all $j \in \{1, \ldots, q\}$.*

Proof. See Ehrgott (2005). □

However, the difficulty here is to find an appropriate choice of ε, as the proof in Ehrgott (2005) relies on the ε_j values that equal to the actual objective function values of the efficient solution one would like to find, so that the construction from the proof of this theorem can be used more as a check of efficiency.

The following result provides us with a relation between weighted-sum and ε-constraint scalarizations.

Theorem 2.72 (Chankong and Haimes, 1983)**.**

1. *Let $j \in \{1, \ldots, q\}$ and suppose that $x^\star$ is an optimal solution of* (2.4)*. If $\omega_j > 0$ then there exists an $\varepsilon^\star \in \mathbb{R}^q$ such that $x^\star$ is an optimal solution of* (2.5) *too.*
2. *Suppose that X is a convex set and $f_i \colon \mathbb{R}^n \to \mathbb{R}$ are convex functions for all $i \in \{1, \ldots, q\}$. If $x^\star$ is an optimal solution of* (2.5) *for some $j \in \{1, \ldots, q\}$, there exists $\omega^\star \geqslant 0$, such that $x^\star$ is optimal for $\min\{(\omega^\star)^\top f(x) \colon x \in X\}$.*

Proof. See Ehrgott (2005). □

3 The Dial-a-Ride Problem

The *dial-a-ride problem* (DARP) is an optimization problem where a finite set of transportation requests either have to be assigned to vehicle routes, or rejected. The transportation requests are submitted by users and are defined by origin, destination, load (i.e., number of passengers), time windows and ride time (i.e., the total time on board of a vehicle). The rides may be shared, meaning that multiple users may be in the same vehicle at the same time.

This chapter introduces the DARP and is structured as follows: First, we review the existing literature on the DARP in Section 3.1. In doing so, we also describe different problem variants and solution approaches. Then, since the DARP is motivated by real-life applications, a variety of these are described in Section 3.2. A formal problem description of the DARP is given in Section 3.3. Moreover, we present two standard MILP formulations from the literature in Section 3.4. We cover common objective functions and approaches towards the multi-objective aspect of this problem in Section 3.5. Finally, related optimization problems are reviewed in Section 3.6. Large parts of this section are taken from the papers Asatryan et al. (2023) and Gaul et al. (2023, 2021, 2022).

3.1 Related Work on DARP Models and Algorithms

In the first part of this section, we consider the *static* DARP, where all user requests are known in advance, i.e., before the vehicles routes and schedule is computed. This is in contrast to the *dynamic* DARP where requests are received during the day and vehicle routes are updated whenever a new request arrives. The dynamic DARP will be the topic of the second part of this section. While for the static DARP the section is divided into exact, heuristic and hybrid solution methods, the dynamic DARP is only solved by heuristic approaches. Within the categories of exact, heuristic and hybrid solution methods, each solution method is described briefly and the relevant literature is summarized. Furthermore, for the static DARP, multi-objective solution approaches, and, for the dynamic DARP, simulation studies are covered specifically. For a broad review of the DARP the reader is referred to Cordeau and Laporte (2007) which covers research on the DARP up to 2007, and to Molenbruch et al. (2017a) or Ho et al. (2018) for more recent in-depth surveys.

Before we proceed with the literature review, we need some further definitions: Besides the distinction between static and dynamic DARP we *distinguish* between the following main variants: The *heterogeneous* DARP refers to both heterogeneous user requests (e.g., w.r.t. load or special needs like the usage of a wheelchair) and/or vehicles (e.g., w.r.t vehicle capacity or equipment). In the *homogeneous* DARP all transportation requests and vehicles have the same characteristics. While in the *deterministic* DARP, all information, when received, is known with certainty, in the *stochastic* DARP, information may be uncertain at the time decisions are made, e.g., the show-up of users at their designated

pickup location, or travel times between locations are taken as uncertain into account. In the DARP with *multiple depots*, there are, as the name suggests, multiple vehicle depots instead of a single depot from which the vehicles start and end their tour. Often, the considered DARP is a combination of the above variants. Besides that, various additional features related to certain practical applications may be studied. The following review, in its respective subsections, is sorted by solution methods rather than DARP variants, and for each solution method we summarize different DARP variants solved with the respective method.

This review also contains some references dealing with the closely related pickup and delivery problem with time windows (PDPTW), see also Section 3.6, which is a special case of the DARP where ride time is neither bounded by constraints nor reflected in the objective function.

3.1.1 The Static DARP

We distinguish exact and heuristic solution approaches for the static DARP. Furthermore, we provide a separate short overview on multi-objective solution methods.

Exact Solution Methods Exact solution methods for the DARP are frequently based on B&C frameworks. In Cordeau (2006) a B&C algorithm is first applied to the DARP. In addition to valid inequalities derived from the related pickup and delivery problem, the vehicle routing problem, and the traveling salesman problem (compare Section 3.6), new valid inequalities are added as cuts to an MILP formulation of the DARP. The MILP model proposed in Cordeau (2006) is based on binary three-index variables, indicating wether a certain vehicle k drives from location i to location j. It has become a standard formulation of the DARP that is also used for problem extensions (see Ho et al., 2018). In Ropke et al. (2007) the model is reformulated (for the homogeneous DARP) by omitting the index for the vehicles. Now, a binary variable indicates whether any vehicle drives directly from location i to location j. We refer to model formulations where binary variables indicate whether two locations are visited directly after each other as *location-based* formulations. Moreover, further valid inequalities are introduced in Ropke et al. (2007): The so-called fork constraints and reachability constraints adapted from Lysgaard (2006), which both turn out to be very effective in their B&C algorithm. We discuss the three-index formulation according to Cordeau (2006) and the two-index formulation proposed in Ropke et al. (2007) in more detail in Section 3.4.

There exist several other exact solution approaches that are based on branch-and-cut frameworks. In a majority of these approaches, B&C is applied to a location-based formulation of the DARP. For example, in Cortés et al. (2010) a pickup and delivery problem that allows users to swap vehicles at specific locations during a trip is solved by combining B&C and *Benders decomposition*. The latter is based on the idea that if the discrete variables of an MILP are fixed the remainder is just an LP subproblem. It remains to determine the optimal values of the discrete variables. Benders decomposition involves the decomposition of the problem into a first-stage master problem with the original discrete variables and only one continuous variable and a second-stage subproblem which is an LP, see, e.g., Schrijver (1998). If the solution of the subproblem indicates that the solu-

tion of the first-stage problem is infeasible, *Benders cuts* are generated and added to the master problem. Benders decomposition is also applied in two papers (Riedler and Raidl, 2018; Rist and Forbes, 2022), where instead of the minimization of costs, as common in single-objective DARPs, the maximization of accepted requests is considered as optimization goal. In Liu et al. (2015) a B&C algorithm is devised and two different models for a DARP with multiple trips, heterogeneous vehicles and configurable vehicle capacity are formulated. The authors introduce eight families of valid inequalities to strengthen the models. In Bongiovanni et al. (2019), a problem variant of the DARP with electric vehicles, E-DARP, is examined. The problem includes special features such as battery management and detours to charging stations. It is solved using a B&C-algorithm with valid inequalities tailored to the new features. Due to the increased environmental sustainability, the introduction of features of electric vehicles is a very relevant research direction, and is also considered in Masmoudi et al. (2018) and Su et al. (2023). A B&C algorithm, as well as new valid inequalities aiming at the detection of infeasible vehicle routes based on time windows, precedence constraints and the capacity of the vehicles are proposed in Morapitiye and Kis (2022). In Schulz and Pfeiffer (2024) a branch-and-cut algorithm implementing a fixed-path procedure for the PDPTW (see Section 3.6) is developed. These fixed paths are extended every time another arc variable is fixed in the branch-and-cut tree. They can be used to improve the lower bounds of the linear programming relaxation based on a location-based model.

A different exact solution approach is *branch-and-price*, see, e.g., Wolsey (1998), where the DARP is split up into two problems: The *master problem* where vehicle routes are selected (typically modeled as a set partitioning problem) and the *pricing problem* where routes are generated (typically modeled as a constrained shortest path problem). The master problem typically contains a larger number of variables as compared to the original MILP modeling the DARP, so that its solution is first restricted to a subset of variables, i.e., only a subset of columns of the constraint matrix is used. To validate the solution found in the so-called *restricted master problem*, the pricing subproblem is solved. If its optimal value is negative, a corresponding column is added to the restricted master problem and it is solved again, else the solution of the restricted master problem is checked for integrality and a branch is created if applicable.

In the following, we discuss *branch-cut-and-price* (BC&P) algorithms, which are combinations of B&C and B&P algorithms. The majority of the following papers use location-based formulations. In Ropke and Cordeau (2009) a BC&P approach is used for the PDPTW where two different formulations of a constrained shortest path problem are compared for the pricing problem. Additionally, the authors introduce valid inequalities for the set partitioning problem. Also in Gschwind and Irnich (2015) a BC&P algorithm is applied. However, the considered pricing problem explicitly includes ride time constraints, making it more difficult to solve. For this reason, the authors add new dominance criteria to a labeling algorithm used to solve the pricing problem, making it more tractable. A heterogeneous vehicle fleet is considered in Qu and Bard (2015), where a BC&P algorithm for a PDPTW is presented. The authors introduce additional dominance criteria and valid inequalities to increase the performance of their algorithm. In Luo et al. (2019) a branch-and-cut-and-price approach is used in the slightly different context of patient transportation. Unlike standard DARP formulations, the considered problem includes

heterogeneous vehicles, manpower constraints, and an objective involving the profit per customer request. Motivated by new challenges brought up by the Covid-19 pandemic, an extension of the DARP with minimum disease transmission risk is considered in Guo et al. (2022): The authors introduce a maximum cumulative risk exposure constraint and minimize a weighted-sum objective of travel cost and disease-transmission risk exposure. The problem is solved using a BC&P-algorithm, where the pricing problem is a resource constrained elementary shortest path problem with minimized maximum risk.

Comparing both approaches, branch-and-cut and branch-and-price, w.r.t. to the number of variables and constraints, the location-based formulation used in the B&C approach requires only a quadratic number of binary variables in the number of locations while in the worst case all feasible tours need to be constructed for the set partitioning formulation in the B&P approach. However, the location-based formulation requires a potentially exponential number of constraints to ensure that corresponding pickup and delivery nodes are visited in the same tour. This is not the case in the pricing problem, as it can be formulated as a resource constrained shortest path problem. In conclusion, different formulations trade-off between the number of binary variables and the number of required constraints. In recent years, new approaches correspond to different compromises between the location-based formulation and the set partitioning formulation. In Rist and Forbes (2021) route fragments which are segments of a vehicle tour such that the vehicle is empty at the beginning and at the end of a fragment are introduced. Thus, vehicle tours can be compounded by fragments. The authors use the fact that time windows typically strongly reduce the number of possible fragments. In their approach, binary decision variables indicate whether a fragment is used in the solution or not and whether fragments are scheduled in direct succession within a tour. Rist and Forbes use a branch-and-cut framework to solve their model. In Gaul et al. (2022) an *event-based formulation* is introduced, where an event represents the current occupancy of the vehicle, together with the location of the last picked up or dropped off user. In this formulation, binary variables indicate whether two events occur in direct succession in a vehicle's tour. The authors rely on the fact that the vehicle capacity is typically rather small in practice, as the number of events strongly depends on the vehicle capacity. This formulation is introduced and discussed in Chapter 4 of this thesis. Both aforementioned formulations from Rist and Forbes (2021) and Gaul et al. (2022) have the advantage that the vehicle capacity constraints are enforced by the choice of fragments and events, respectively. Moreover, precedence constraints are either ensured by the choice of fragments or can be implemented implicitly by the selection of arcs connecting events. This considerably reduces the number of constraints. In Figure 3.1 a classification of the discussed formulations for the DARP with regard to the amount of information encoded in a binary variable is shown. Note that, a survey on formulations for the related capacitated vehicle routing problem (see Section 3.6) and a systematic study of how these formulations relate to each other is given in Letchford and Salazar-González (2005).

The DARP is an $\mathcal{NP}$-hard problem, as it is a generalization of the vehicle routing problem, see Section 3.6. Since the computational time to solve an instance of the DARP strongly depends on the instance size and parameter values, heuristic methods are a useful complementary approach to exact algorithms.

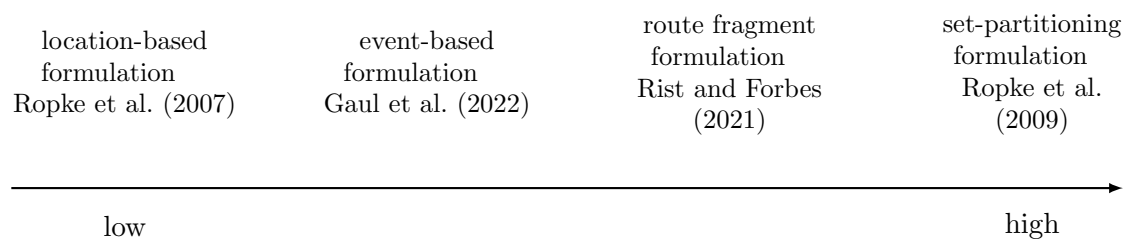


Figure 3.1: Classification of formulations according to the amount of information encoded in a binary variable with value 1

Heuristic Solution Methods Early work is carried out in Jaw et al. (1986), who develop one of the first heuristics for the DARP. Depending on the users' earliest possible pickup times, the heuristic determines the cheapest insertion position in an existing route in terms of user satisfaction and operator costs. In Desrosiers et al. (1991), Dumas et al. (1989) and Ioachim et al. (1995) groups of users to be served within the same area and time are first identified. In a second step, the clusters are combined to obtain feasible vehicle routes. The authors present different techniques to build these clusters. This frequently-applied decomposition approach, however, leads in general to suboptimal solutions.

In Cordeau and Laporte (2003) *tabu search* is first applied to the DARP. Tabu search is introduced by Glover, see Glover and McMillan (1986), and is a meta-heuristic solution method, where the algorithm moves to the best solution in a neighborhood of the incumbent in every iteration. To avoid cycling, certain solutions are declared *tabu* for a given number of iterations. Furthermore, a diversification mechanism allows the algorithm to move to solutions that are worse than the best known solution to avoid being trapped in local optima. A simple neighborhood generation technique is used in Cordeau and Laporte (2003) by shifting requests from one route to another. The reported computational results on real-life data as well as on randomly generated instances validate the efficiency of the heuristic. For more recent work on the static DARP using variants of tabu search based on Cordeau and Laporte (2003), see, for example, Detti et al. (2017), Guerriero et al. (2013), Kirchler and Wolfler Calvo (2013), and Paquette et al. (2013).

Variable neighborhood search (VNS) is a heuristic, which uses different types of local destroy-and-repair operators to create neighborhoods, and is introduced in Mladenović and Hansen (1997). Local search operators may include, for example, the shift of a request from one route to another, the swap of two requests between routes, or the reinsertion of a request in a route. Hence, local search operators make relatively small changes to the solution. First, a random solution is created in the current neighborhood of the incumbent. After applying local search to this solution, it is compared to the incumbent, and replaces it, if it improves the objective function or other acceptance criteria are met. In this case, the search restarts in the first neighborhood. If the solution does not improve the objective function or fulfills other acceptance criteria, the search moves on to the next neighborhood. Variable neighborhood search is first applied to the DARP in Parragh et al. (2009) which lays the foundation for the following publications: In Parragh et al. (2010b) VNS is used to solve a static single-depot DARP with homogeneous users and vehicles, in Detti et al.

(2017), Parragh (2011), and Parragh et al. (2010a) a VNS for a DARP with heterogeneous vehicles and users arising from the context of healthcare transportation is proposed, in Muelas et al. (2013, 2015) a large-scale DARP in a big city is considered, in Parragh et al. (2015) a DARP where transportation requests involving several persons may be split is solved, and in Souza et al. (2020) VNS is combined with a set-covering strategy to solve a DARP with heterogeneous vehicles.

In contrast to small changes made by the destroy-and-repair operators in the previously described meta-heuristics, in a *large neighborhood search* (LNS) the destroy-and-repair operators modify a substantial part of the solution. Large neighborhood search is proposed in Shaw (1998) for vehicle routing problems. For example, a number of k requests is removed from one route and reinserted into different other routes afterwards. In comparison to local search operators, which intensify the search in the neighborhood of a given solution, large neighborhood operators diversify the search. *Adaptive large neighborhood search* (ALNS) is introduced in Ropke and Pisinger (2006) and applied to the PDPTW, which is extended to DARP applications (e.g., Gschwind and Drexl, 2019; Masmoudi et al., 2019; Pfeiffer and Schulz, 2021). In an ALNS algorithm, the removal and insertion heuristics used in the destroy-and-repair operator are chosen based on their success in finding new best solutions in previous iterations. Thereby, the search algorithm adapts its performance to different instance types and achieves good results for diverse sets of instances. In adopting the ALNS algorithm proposed in Ropke and Pisinger (2006), a meta-heuristic solution method for the DARP is developed in Gschwind and Drexl (2019): It uses a constant-time feasibility check in the repair step of the ALNS, a local-search-based intraroute improvement of routes of promising solutions using the Balas–Simonetti neighborhood (see Balas and Simonetti, 2001) and the solution of a set covering model over a subset of all routes generated during the search. According to Ho et al. (2018), it is at the moment the most efficient heuristic method to solve the standard DARP. An (adaptive) LNS is applied to the pickup and delivery problem or the DARP with transfers in Masson et al. (2013, 2014) and Molenbruch et al. (2021). The *DARP with transfers* models the highly relevant problem of integrating on-demand services and other modes of transportation such as public transport, as on-demand systems are often considered as highly-efficient last-mile transportation services. An *adaptive variable neighborhood search* algorithm is proposed in Johnsen and Meisel (2022) to solve a DARP with interrelated trips. The latter refer to scenarios where several transport requests need to be synchronized, e.g., if two users would like to arrive at a certain location at the same time. The authors combine a VNS with the adaptive insertion and removal heuristic selection strategy introduced in Ropke and Pisinger (2006).

Another frequently applied class of meta-heuristics are *genetic algorithms*, which mimic the evolution of a finite population, representing a set of solutions, see, e.g., Wolsey (1998). The population is modified by operators inspired by biological evolution, such as selection, crossover or mutation. In Jorgensen et al. (2007) a genetic algorithm is used to assign passengers to vehicles. In a second step, routes are constructed sequentially by means of a nearest neighbor procedure. A genetic algorithm is also used in Cubillos et al. (2009) showing slightly better results than Jorgensen et al. (2007). A multi-objective genetic algorithm is devised in Atahran et al. (2014). In Belhaiza (2017) and Belhaiza (2019) genetic crossover operators are combined with variable neighborhood search and adaptive

large neighborhood search, respectively. A hybrid genetic algorithm for the heterogeneous DARP is developed in Masmoudi et al. (2017) by combining a genetic algorithm with local search operators. According to Ho et al. (2018) the best results on benchmark instances of the heterogeneous DARP are reported in Masmoudi et al. (2017).

In *simulated annealing*, a randomly chosen solution from the neighborhood of the incumbent is accepted immediately if it improves the objective value, and accepted with probability $0 < p < 1$ if it does not, see, e.g., Wolsey (1998). The latter helps to avoid getting stuck in local optima. The probability of acceptance is physically motivated and depends on an initial "temperature", a "cooling ratio", and the difference in objective values between the chosen solution and the incumbent. As the probability of acceptance depends on the difference in objective function values, slightly worse solutions have a higher probability of acceptance, while a significant deterioration will be accepted only rarely. The temperature decreases over time, so that with an increasing number of iterations, the probability of accepting worse solutions decreases. Simulated annealing is applied to the DARP in Reinhardt et al. (2013), Mauri et al. (2009) and Zidi et al. (2012). Moreover, in Braekers et al. (2014) a variant of simulated annealing, *deterministic annealing*, is applied, where worse solutions are only accepted as long as the difference in objective values does not fall below a given threshold. Additionally, more complicated destroy-and-repair operators than in the previous publications on simulated annealing are employed, and a restart strategy is used.

A recently proposed meta-heuristic is *iterative local search* (see Lourenço et al., 2018), which is a state-of-the-art algorithm for many computationally challenging problems. In iterative local search, a sequence of solutions is generated by an embedded heuristic, often local search, which in general leads to better solutions than random sampling. In Malheiros et al. (2021) iterative local search is applied to a DARP with single and multiple depots and heterogeneous users and vehicles. After each iteration of local search, the generated solutions are the input for a set partitioning procedure, where the best set of routes from a pool is chosen by solving a set partitioning formulation.

Hybrid Solution Methods There is a growing trend towards hybridizing exact and meta-heuristic methods to solve the DARP and other vehicle routing problems (see Dumez et al., 2021). For example, in Hu and Chang (2014) a branch-and-price (B&P) algorithm is applied to a DARP with time-dependent travel times, in which the pricing subproblem is solved by large neighborhood search. As a side result, the authors observe that the length of time windows can have a significant impact on the size of the fleet, the objective function value, the CPU time, the average ride time or the average pickup time delay. In Parragh et al. (2010a) and Parragh and Schmid (2013) a column-generation approach is presented, where the pricing problem is solved by a VNS, and in Tellez et al. (2018) a set-partitioning problem in an LNS for a dial-a-ride problem with heterogeneous requests and vehicles, which may be reconfigured en-route, is solved.

Multi-Objective Approaches While most of the literature on the DARP focuses on the single-objective DARP, many real-life applications are characterized by multiple conflicting objectives (see Section 3.5), which motivates the examination of the DARP from a multi-

objective optimization perspective. As stated in Ho et al. (2018), this line of research can be divided into three main categories: weighted-sum based approaches, lexicographic approaches, and the determination of (an approximation of) the set of all non-dominated outcome vectors.

Weighted-sum objectives are used, e.g., in Bongiovanni et al. (2019), Jorgensen et al. (2007), Kirchler and Wolfler Calvo (2013), Mauri et al. (2009), Melachrinoudis et al. (2007), and Su et al. (2023). In addition to the customers' total transportation time, which is a common objective in single-objective DARPs, the weighted-sum objective introduced in Jorgensen et al. (2007) is composed of the total regret (the difference in time compared to a user's earliest possible arrival time at his or her destination), the customers' waiting time, the drivers' work time as well as several penalty functions for the violation of constraints. The weights on these criteria are chosen with respect to the relative importance of the criteria from a user-perspective. While the methods to solve the DARP with a weighted-sum objective range from a genetic algorithm (Jorgensen et al., 2007), to tabu search (Kirchler and Wolfler Calvo, 2013; Melachrinoudis et al., 2007), simulated annealing (Mauri et al., 2009), deterministic annealing (Su et al., 2023) or B&C (Bongiovanni et al., 2019), most of the authors either adapt the weights introduced in Jorgensen et al. (2007) or choose them with respect to individual assessments of their relative importance.

In Luo et al. (2019) a lexicographic ordering of the objective functions w.r.t. their relative importance is considered. In this context, the authors propose a two-phase BC&P-algorithm. In the first phase a set of non-dominated trips is enumerated by a label-setting algorithm. Based on the set of non-dominated trips, a formulation of the DARP called "trip-based model" is introduced. In the second phase the model is solved using BC&P.

The non-dominated set is examined, e.g., in Atahran et al. (2014), Chevrier et al. (2012), Hu et al. (2019), Molenbruch et al. (2017b), Paquette et al. (2013), Parragh et al. (2009), Viana et al. (2019), and Zidi et al. (2012). A comparison of six multi-objective evolutionary algorithms to solve the multi-objective DARP is provided in Guerreiro et al. (2020).

While the static DARP is (practically) relevant on its own right and much research focuses on the static DARP, see, e.g., Ho et al. (2018), we note that static DARP models can also be extended to the dynamic scenario by using a rolling-horizon strategy, see, e.g., Gaul et al. (2021). This is discussed in more detail in Chapter 6.

3.1.2 The Dynamic DARP

In the first part of our literature review on the dynamic DARP we deal with solution methods. After that, we particularly review simulation studies concerned with the dynamic DARP.

Despite being a highly relevant topic of research, the dynamic DARP is less studied than its static counterpart (see Ho et al., 2018). Here, we focus on the dynamic while still deterministic DARP, i.e., we assume that all information, when received, is known with certainty. A broad review on the dynamic DARP is given in Ho et al. (2018), while a survey on the related dynamic pickup and delivery problem can be found in Berbeglia et al. (2010).

Solution strategies to the dynamic DARP are often motivated by the requirement to

immediately determine a feasible routing that includes the new requests. A frequently applied solution strategy to dynamic DARPs combines two approaches (see, e.g., Attanasio et al., 2004; Beaudry et al., 2008; Berbeglia et al., 2012; Carotenuto and Martis, 2017; Coslovich et al., 2006; Häll and Peterson, 2013; Lois and Ziliaskopoulos, 2017; Marković et al., 2015; Santos and Xavier, 2015; Vallee et al., 2020): On the one hand, a new request is inserted using fast and simple insertion heuristics. In the idle time between a pair of new requests, on the other hand, a more complex heuristic or meta-heuristic may be used to continually re-optimize the current solution. We give a brief overview on the variants of insertion and re-optimization heuristics used in the literature.

The first and most simple insertion heuristic tries to insert the new request in the current vehicle routes without relocating already assigned users. If a feasible insertion position is found, then the new request is inserted in the best insertion position in terms of incremental cost. Variants of this strategy are employed, for example, in Beaudry et al. (2008), Carotenuto and Martis (2017), Hanne et al. (2009), Häll and Peterson (2013), Lois and Ziliaskopoulos (2017), Madsen et al. (1995), Marković et al. (2015), Psaraftis (1980) and Santos and Xavier (2015). Especially in applications with a very large number of new requests per time unit (e.g., a ridepooling service in the city of New York), as considered in Pouls (2023), a fast insertion heuristic as described above is crucial. In addition to an algorithm for a large-scale dynamic DARP, a repositioning strategy for idle vehicles as well as a simulation-based framework to evaluate real-world scenarios are proposed in Pouls (2023).

The second variant of an insertion heuristic allows the relocation of already assigned users, thus leading two a higher number of possible insertion positions for the new request. For instance, in Attanasio et al. (2004) parallel heuristics are used to solve the dynamic DARP combining random insertion and tabu search. In Berbeglia et al. (2012) a tabu search heuristic is run in parallel with a constraint programming algorithm to determine whether a new request can be inserted feasibly in a given solution or not. In Luo and Schonfeld (2011) requests which are similar w.r.t. time windows and geographic locations are relocated whenever a simple insertion heuristic declares a new request to be infeasible. In Vallee et al. (2020) three different heuristics are proposed aiming at resorting already accepted requests if a new request's insertion is declared infeasible by a service provider's online system. In Coslovich et al. (2006) new unexpected requests may show up at a vehicle stop. In the idle time between two vehicle stops, a neighborhood of the current vehicle route is created. The insertion of the unexpected request is evaluated for all routes in the neighborhood of the current route. A maximum cluster algorithm, developed in Häme and Hakula (2015), that finds, for each set of users, a maximal subset of users that can be served by one vehicle can be used to quickly decide if new requests should be accepted or rejected. In Souza et al. (2021) a DARP with no rejects within the context of patient transportation is considered. Requests may be reallocated if emergency requests appear. Furthermore, the authors investigate the effects of dynamic requests on the solution and conclude that dynamism generally increases the number of vehicles needed and the violations of time windows (if allowed).

The second phase of a solution approach to the dynamic DARP often consists of a re-optimization phase. To improve the current solution in the idle time between a pair of new requests, different variants of local search are applied. For example, a reinsertion heuristic

is used to remove a request from its current route and evaluate the reinsertion of the request into all other routes, and/or a swap heuristic exchanges two requests with different routes, see, e.g., Lois and Ziliaskopoulos (2017), Luo and Schonfeld (2011), Marković et al. (2015), and Santos and Xavier (2015). In Carotenuto and Martis (2017) the quality of the solution is sought to be improved by reinserting the entire set of accepted and not yet picked up requests. In Häll and Peterson (2013) several destroy-and-repair heuristics are combined and compared; in particular ruin methods based on the removal of sequences of requests are proven to improve the quality of solutions. In Attanasio et al. (2004), Beaudry et al. (2008) and Berbeglia et al. (2012) (different variants) of tabu search are used in the improvement phase.

Case Studies and Simulations A review of simulation studies dealing with individual and agent-based demand-responsive transport systems can be found in Ronald et al. (2015). As the authors state, a majority of simulations are concerned with the optimization of trips, usually from an operator's perspective. In agent-based modeling, the interactions between operators and customers are studied by replicating the decision making process of individual travelers concerning the choice of destination, mode and route.

The first part of this paragraph deals with the former: studies which simulate changes in features of ridepooling services from an operator's perspective, i.e., changes in parameter settings or modes of operation such as the length of the time windows. In the second part of this paragraph, studies concerned with the impact of ridepooling on other modes of transport such as cars or public transport, and their interdependencies, are discussed. A majority of these studies is conducted using agent-based modeling.

Two case studies are conducted in Colorni and Righini (2001): On the one hand, the authors investigate a particular problem occurring in the city of Crema in Nothern Italy: At days with a farmers' market, every customer accepted by a dial-a-ride service has to be served twice, from home to the market and back, and all trips share a common origin or destination: the market. On the other hand, they study the feasibility of a mixed static-dynamic dial-a-ride system. The authors give insights on the dependency of the level of service on the number of customers (and the number of overlapping time windows), the planning horizon and the number of vehicles and their capacity. In Quadrifoglio et al. (2008) the effect of a zoning vs. no-zoning strategy and the length of time windows in dial-a-ride services are analyzed using data from a Los Angeles ridepooling service. According to the study, larger time windows and a centralized dispatching system reduce the number of vehicles and miles driven but also reduce the service quality for the users of ridepooling services. The effects of a partially dynamic environment are investigated in Wong et al. (2012): The authors introduce the *degree of dynamism* and investigate the influence of the ratio of dynamic requests on the system performance. Higher transportation costs and fewer accepted requests are the result of partially dynamic requests as compared to fully static or dynamic requests. In Häll et al. (2012) a graphical user interface is established to simulate dynamic dial-a-ride services with multiple fleets, different vehicle capacities, schedules and depots. As an illustration, the authors include costs for waiting time and users' regret in the service, and calculate the price of efficiency improvements in exchange for less user convenience. Another simulation study is conducted in Häll et al. (2015):

The authors analyze which changes in parameter settings in dynamic DARPs have a large impact on performance criteria such as customer satisfaction and operational costs and establish guidelines to service providers as a result of their study. In Lois and Ziliaskopoulos (2017) the trade-off between long-term highly optimized versus myopic optimization procedures is investigated and it is suggested that both optimization techniques should be used dependent on the load scenario. In Hungerländer et al. (2021) the pooling rates of an Austrian mobility provider in a rural area are aimed to be improved. In the study a large neighborhood search is implemented to solve the respective dynamic DARP and identify the most promising parameter settings to improve pooling and user convenience. We now summarize the literature on simulation studies which analyze the interdependencies of ridepooling with other modes of transport such as cars or public transport.

The integration of fixed public transport and dial-a-ride services is examined in Posada and Häll (2020). The purpose is to reduce costs of the often highly subsidized on-demand service, by allowing certain parts of the user's trips to be replaced by public transport. The authors compare the integrated with the non-integrated on-demand service and conclude that the driven distance can be reduced by 16% using the integrated service. The proposed meta-heuristic framework can help policy makers to gain insights into the effects of an integrated service. The substitution of all trips made by private cars and buses by autonomous shared vehicles in an urban setting is investigated in ITF (2015). The findings are that 9 out of 10 cars could become obsolete, resulting in a huge amount of freed space. As a negative effect, the total travel volume increases. Moreover, mixing a fleet of shared vehicles with private cars will not result in the same benefits as a pure system of autonomous shared vehicles and autonomous taxis. The simulation uses an agent-based model and synthetic trips are based on real trips generalized to a grid. The two following publications use MATSim (Horni et al. (2016)) as the simulation software. Its basic concept is the simulation of agents that make one or more trips a day using various transport modes (e.g., car, taxi, ridepooling or public transport). In Bischoff et al. (2017) the integration of shared rides into a simulation framework for non-shared taxi services is described. The simulation suggests that 15–20% of vehicle kilometers can be saved while travel time increases at most by 3% on average. The authors further remark that pooling works best in areas with a high taxi demand. The overall demand for pooled rides could increase even more with the introduction of autonomous vehicles since then lower fares could be offered. The complete replacement of public transport services in a mid-sized city of 100,000 inhabitants by ridepooling services is simulated in Bischoff et al. (2019). The authors distinguish between a stop-based transportation, where the remaining distance to the customer's location is walked on foot, and door-to-door transportation. Results suggest that the current public transport system could be replaced by 300 to 400 vehicles. In Wilkes et al. (2021) the travel demand model mobiTopp is used as an agent-based simulation system. The authors describe the integration of ridesourcing, i.e., services connecting drivers of shared and non-shared taxi services with users, into the travel demand model. They analyze the impact on service providers in terms of occupation rate or number of vehicles and the interdependencies with other modes of transport. The integration of autonomous taxi services into a microsimulation is described in Dandl et al. (2017). Travel times are modeled taking into account delays due to left turns or traffic lights. Moreover, taxi movements influence the flow in the street network and have the

potential to change travel times. The authors analyze the impact of these more realistic traffic conditions. A second focus is the impact of empty trips (i.e., movement of taxis without passengers on board). The simulation is conducted in the greater area of Munich, Germany. In Richter et al. (2019) characteristics of transportation with autonomous vehicles are integrated into a macroscopic four-stage model (trip generation, trip distribution, modal split, and route assignment). A framework for modeling the impacts of autonomous vehicles on the network performance and capacity and on travel demand is presented. Moreover, the framework is used to evaluate the impact of autonomous vehicles on empty trips and ridepooling services.

3.2 Applications

Dial-a-ride systems originated as door-to-door transportation services, primarily to complement or to replace public transport in areas or times with low customer requests, e.g., in rural areas or during night time, where public transport is rarely available with buses running only a couple of times a day. User requests were usually phoned in before the start of service, which corresponds to the static variant of the DARP.

In health care transport, dial-a-ride services are an appropriate alternative to expensive taxi services, as elderly, injured or disabled persons are often not able to drive on their own or use public transport. This application is modeled by a heterogeneous DARP, because usually specialized equipment is needed in the vehicles, e.g., extra space for a wheelchair, and users have individual needs.

A convenient alternative to public transport, but more affordable than a taxi, is the concept of ridepooling: a taxi-like service, typically operated by mini-buses, where users submit pickup and delivery locations and a desired pickup or delivery time via their smartphones. In contrast to taxi-services, where pooling is not allowed, customers with similar origin or destination are assigned to the same ride whenever economically and/or ecologically useful. Thus, ridepooling services are comparable to taxi-services but at a lower cost. They can also be used to substitute private car rides with the potential to reduce congestion and fine dust pollution in big cities. Since rides may be shared, users may have to accept longer ride times. In exchange, they can be offered lower fares. In the past couple of years, several so-called 'on-demand ridepooling services' have emerged. Prominent examples are Uber[1], DiDi[2] or moia[3].

This thesis is motivated by the ridepooling service *Hol mich! App*[4] which was launched in 2019 in the mid-sized city of Wuppertal in Germany. On-demand ridepooling services complement public transport and are usually established in urban areas. However, since they can be easily adapted to varying passenger demands, they provide an acceptable service quality also in suburban areas. On-demand ridepooling services are an application of the dynamic DARP.

While different variants of the deterministic DARP are interesting on their own right,

[1] `https://www.uber.com/de/en/ride/uberpool/`
[2] `https://web.didiglobal.com/`
[3] `https://www.moia.io`
[4] `https://www.holmich-app.de`

note, that all real-life applications of the DARP are uncertain because of external circumstances like traffic or unpredictable user behavior.

3.3 Problem Description and Notation

In this subsection, we consider the DARP in its basic variant, i.e., the deterministic and static DARP, where it is assumed that all requests have to be accepted and the objective is to minimize the routing costs, see Ho et al. (2018). Additionally, we assume that the vehicle fleet is homogeneous. Note, that this is a common setting for the static DARP, see, e.g., Gschwind and Drexl (2019) and Ropke et al. (2007). It is defined as follows:

Let n be the number of users submitting a transport request. Each request $i \in R \coloneqq \{1, \dots, n\}$ specifies a pickup location i^+ and a delivery location i^-. The sets of pickup and delivery locations are denoted by $P \coloneqq \{1^+, \dots, n^+\}$ and $D \coloneqq \{1^-, \dots, n^-\}$, respectively. At the beginning of service a homogeneous fleet of K vehicles with capacity Q each is situated at the vehicle depot, which is denoted by 0. A number of requested seats $q_i \geq 1$ and a service duration of $s_i \geq 0$, which is interpreted as the time for entering or leaving the vehicle, are associated with each request $i \in R$. We set $q_{i^+} = q_{i^-} \coloneqq q_i$, $s_{i^+} = s_{i^-} \coloneqq s_i$ and $q_0 \coloneqq 0$ as well as $s_0 \coloneqq 0$. The maximum ride time corresponding to request $i \in R$ is denoted by L_i. For the travel time $\bar{t}_{ij}$ and corresponding routing costs $\bar{c}_{ij}$ between each pair of locations $i, j \in J \coloneqq P \cup D \cup \{0\}$ we assume that $\bar{c}_{ij}$ and $\bar{t}_{ij}$ are non-negative and fulfill the triangle inequality. Let $\bar{t}_i$ denote the travel time for a direct ride from pickup to delivery location of request i, i.e., $\bar{t}_i = \bar{t}_{i^+i^-}$. W.l.o.g. we assume $\bar{t}_i + s_i > 0$. With each location $j \in J$ a time window $[e_j, \ell_j]$ is associated. The beginning of service is given by the lower bound e_0 of the time window at the depot, and there is a fixed duration of service T, so that the end of service is given by $\ell_0 \coloneqq e_0 + T$. We assume that $e_{i^+} \geq \bar{t}_{0i^+}$, i.e., each pickup location can be reached at the beginning of the time window when starting at time zero in the depot.

A feasible solution to the DARP consists of at most K vehicle routes which start and end at the depot. If a user is served by a vehicle the user's pickup and delivery location both have to be contained in this order in vehicle's route. The vehicle capacity of Q may not be exceeded at any time. The start of service at every location has to be within the time windows. It is possible to reach a location earlier than the start of service and wait. At each location j, a service duration of s_j minutes is needed for users to enter or leave the vehicle. In addition, the acceptable ride time of each user i is bounded from above by L_i. Vehicles have to return to the depot at least T minutes after the time of the overall start of service e_0. All users have to be served. The objective is to minimize the routing costs.

In this thesis, we consider the above problem variant of the DARP in Chapters 4 and 5. In Chapter 4, we additionally consider the option that users may be rejected and use a weighted-sum objective, where one optimization goal is to maximize the number of accepted requests. This setting is continued in Chapters 6 and 7, that both consider the dynamic DARP. For the purpose of this introductory chapter, we next present two standard MILP formulations from the literature for the above problem variant.

3.4 Location-based MILP Models

In this section, we present two standard MILP formulations for the DARP from the literature: the three-index formulation proposed in Cordeau (2006) and the two-index formulation according to Ropke et al. (2007). Both models are location-based formulations, i.e., binary variables indicate whether two locations are visited directly after each other.

The models are based on a complete graph, where the node set corresponds to the set of all locations $\bar{J} \coloneqq P \cup D \cup \{0^+, 0^-\}$. Note that, in this model the location of the vehicle depot is represented by two distinct nodes 0^+ and 0^-, providing the option to include a different start and end depot location in the model. However, in this thesis, 0^+ and 0^- always represent the same vehicle depot and are only used to stick to the notation from the original papers Cordeau (2006) and Ropke et al. (2007). We set $s_{0^+} \coloneqq s_{0^-} \coloneqq s_0$, $q_{0^+} \coloneqq q_{0^-} \coloneqq q_0$, $e_{0^+} \coloneqq e_{0^-} \coloneqq e_0$, $\ell_{0^+} \coloneqq \ell_{0^-} \coloneqq \ell_0$ as well as $\bar{c}_{0^+j} \coloneqq \bar{c}_{0^-j} \coloneqq \bar{c}_{0j}$, $\bar{c}_{i0^+} \coloneqq \bar{c}_{i0^-} \coloneqq \bar{c}_{i0}$, $\bar{t}_{0^+j} \coloneqq \bar{t}_{0^-j} \coloneqq \bar{t}_{0j}$ and $\bar{t}_{i0^+} \coloneqq \bar{t}_{i0^-} \coloneqq \bar{t}_{i0}$ for $i, j \in P \cup D$. Moreover, to distinguish between pickup and delivery locations when computing the vehicle load, let

$$\mathcal{I}_j \coloneqq \begin{cases} 1 & j \in P, \\ -1 & \text{else} \end{cases} \qquad \text{for } j \in \bar{J}$$

be an indicator parameter for pickup nodes.

3.4.1 Three-Index Formulation

We present the MILP model introduced in Cordeau (2006) for the standard DARP described in the prior Subsection 3.3. Thus, we assume that the vehicle fleet is homogeneous and adapt the MILP model from Cordeau (2006) in this aspect, i.e., we set the vehicle capacity for each vehicle $k \in \bar{K} \coloneqq \{1, \dots, K\}$ to Q, the duration of service of each vehicle to T, and assume that the routing costs for all vehicles are equal. Furthermore, in this standard model the only objective is to minimize the routing costs. This model is a standard compact three-index formulation of the DARP.

With each arc (i, j) and each vehicle k a binary variable $\bar{x}_{ij}^k$ is associated, where $\bar{x}_{ij}^k = 1$ if vehicle k drives directly from location i to location j, i.e., if the arc (i, j) is used, and $\bar{x}_{ij}^k = 0$ otherwise. Moreover, the variable Q_j^k represents the number of users in the vehicle after leaving location j, $\bar{L}_i^k$ denotes the ride time of user i on vehicle k, and $\bar{B}_j^k$ is the time when service of vehicle k at location j begins.

Then the DARP can be formulated as the following non-linear mixed-integer program:

$$\min \quad \sum_{k \in \bar{K}} \sum_{i \in \bar{J}} \sum_{j \in \bar{J}} \bar{c}_{ij} \, \bar{x}_{ij}^k \tag{3.1a}$$

$$\text{s. t.} \quad \sum_{k \in \bar{K}} \sum_{j \in \bar{J}} \bar{x}_{i^+j}^k = 1 \quad \forall i \in P, \tag{3.1b}$$

$$\sum_{j \in \bar{J}} \bar{x}_{i^+j}^k - \sum_{j \in \bar{J}} \bar{x}_{i^-j}^k = 0 \quad \forall i \in P, \, k \in \bar{K}, \tag{3.1c}$$

$$\sum_{j \in \bar{J}} \bar{x}_{0^+j}^k = 1 \quad \forall k \in \bar{K}, \tag{3.1d}$$

$$\sum_{j\in\bar{J}} \bar{x}^k_{ji} - \sum_{j\in\bar{J}} \bar{x}^k_{ij} = 0 \quad \forall i \in P \cup D,\, k \in \bar{K}, \tag{3.1e}$$

$$\sum_{i\in\bar{J}} \bar{x}^k_{i0^-} = 1 \quad \forall k \in \bar{K} \tag{3.1f}$$

$$\bar{B}^k_j \geq (\bar{B}^k_i + s_i + \bar{t}_{ij})\bar{x}^k_{ij} \quad \forall i,j \in \bar{J},\, k \in \bar{K}, \tag{3.1g}$$

$$Q^k_j \geq (Q^k_i + \mathcal{I}_j q_j)\bar{x}^k_{ij} \quad \forall i,j \in \bar{J},\, k \in \bar{K}, \tag{3.1h}$$

$$\bar{L}^k_i = \bar{B}^k_{i^-} - (\bar{B}^k_{i^+} + s_i) \quad \forall i \in P,\, k \in \bar{K}, \tag{3.1i}$$

$$e_j \leq \bar{B}^k_j \leq \ell_j \quad \forall j \in \bar{J}, k \in \bar{K}, \tag{3.1j}$$

$$\bar{t}_i \leq \bar{L}^k_i \leq L_i, \quad \forall i \in P,\, k \in \bar{K}, \tag{3.1k}$$

$$\max(0, \mathcal{I}_j q_j) \leq Q^k_j \leq \min(Q, Q + \mathcal{I}_j q_j) \quad \forall j \in \bar{J},\, k \in \bar{K}, \tag{3.1l}$$

$$\bar{B}^k_j \geq 0 \quad \forall j \in \bar{J},\, k \in \bar{K}, \tag{3.1m}$$

$$Q^k_j \geq 0 \quad \forall j \in \bar{J},\, k \in \bar{K}, \tag{3.1n}$$

$$\bar{L}^k_i \geq 0 \quad \forall i \in P,\, k \in \bar{K}, \tag{3.1o}$$

$$\bar{x}^k_{ij} \in \{0,1\}, \quad \forall i,j \in \bar{J},\, k \in \bar{K}. \tag{3.1p}$$

The objective function (3.1a) minimizes the routing costs. It is ensured by constraints (3.1b) and (3.1c), that each request is served once by exactly one vehicle and that the pickup and delivery node are contained in the same vehicle tour. Constraints (3.1d) and (3.1f) guarantee that each vehicle leaves the start depot and arrives at the end depot. Constraints (3.1e) are flow conservation constraints. The consistency of time and load variables is modeled by constraints (3.1g) and (3.1h). Note that indicator parameters are needed here to correctly calculate the change in vehicle load after entering or leaving a vehicle. The ride time of users is measured by constraints (3.1i) and it is ensured that the maximum ride time is not violated by constraints (3.1k). Moreover, the latter constraints make sure, that every user's pickup location is visited prior to its delivery location. The compliance with time windows and vehicle capacity is modeled in constraints (3.1j) and (3.1l).

As stated in Cordeau (2006), constraints (3.1g) and (3.1h) are non-linear, since they invoke products of the variables $\bar{x}^k_{ij}$ and $\bar{B}^k_i$, and $\bar{x}^k_{ij}$ and Q^k_i, respectively, but can be linearized in terms of the following big-M constraints:

$$\bar{B}^k_j \geq \bar{B}^k_i + s_i + \bar{t}_{ij} - \bar{M}_{ij}(1 - \bar{x}^k_{ij}) \quad \forall i,j \in \bar{J},\, k \in \bar{K}, \tag{3.2}$$

$$Q^k_j \geq Q^k_i + \mathcal{I}_j q_j - W_{ij}(1 - \bar{x}^k_{ij}) \quad \forall i,j \in \bar{J},\, k \in \bar{K}, \tag{3.3}$$

where $\bar{M}_{ij} \geq \max(0, \ell_i + s_i + \bar{t}_{ij} - e_j)$ and $W_{ij} \geq \min(Q, Q + \mathcal{I}_j q_j)$ are constants. Furthermore, the authors reduce the number of variables and constraints in this model by introducing aggregated variables $\bar{B}_j$, Q_j and $\bar{L}_i$ (at every location except the start and end depot). In this way, constraints (3.2) can be replaced by

$$\bar{B}_j \geq \bar{B}^k_{0^+} + s_0 + \bar{t}_{0^+j} - \bar{M}_{0^+j}(1 - \bar{x}^k_{0^+j}) \quad \forall j \in P \cup D,\, k \in \bar{K}, \tag{3.4}$$

$$\bar{B}_j \geq \bar{B}_i + s_i + \bar{t}_{ij} - \bar{M}_{ij}(1 - \sum_{k\in\bar{K}} \bar{x}^k_{ij}) \quad \forall i,j \in P \cup D, \tag{3.5}$$

$$\bar{B}^k_{0^-} \geq \bar{B}_i + s_i + \bar{t}_{i0^-} - \bar{M}_{i0^-}(1 - \bar{x}^k_{i0^-}) \quad \forall i \in P \cup D,\, k \in \bar{K}, \tag{3.6}$$

constraints (3.1i) can be replaced by

$$\bar{L}_i = \bar{B}_{i^-} - (\bar{B}_{i^+} + s_i) \quad \forall i \in P \tag{3.7}$$

and constraints (3.3) can be replaced by

$$Q_j \geq Q^k_{0^+} + \mathcal{I}_j q_j - W_{0^+j}(1 - \bar{x}^k_{0^+j}) \quad \forall j \in P \cup D,\, k \in \bar{K}, \tag{3.8}$$

$$Q_j \geq Q_i + \mathcal{I}_j q_j - W_{ij}(1 - \sum_{k \in \bar{K}} \bar{x}^k_{ij}) \quad \forall i, j \in P \cup D, \tag{3.9}$$

$$Q^k_{0^-} \geq Q_i + q_0 - W_{i0^-}(1 - \bar{x}^k_{i0^-}) \quad \forall i \in P \cup D,\, k \in \bar{K}. \tag{3.10}$$

Note that, we can omit the remaining constraints ensuring consistencies of time and load variables for traveling on arcs $(0^-, 0^+)$, $(j, 0^+)$ and $(0^-, j)$ because these arcs are not part of a feasible solution. We can omit these constraints for arc $(0^+, 0^-)$ as well because they would not impose any further restrictions on the variables $\bar{B}^k_{0^+}$, $\bar{B}^k_{0^-}$, $Q^k_{0^+}$ and $Q^k_{0^-}$. Ultimately, the authors state that as described in Desrochers and Laporte (1991), the constraints (3.9) can be lifted as follows:

$$Q_j \geq Q_i + \mathcal{I}_j q_j - W_{ij}(1 - \sum_{k \in \bar{K}} \bar{x}^k_{ij}) + (W_{ij} - \mathcal{I}_i q_i - \mathcal{I}_j q_j) \sum_{k \in \bar{K}} \bar{x}^k_{ji} \quad \forall i, j \in P \cup D. \tag{3.11}$$

The model (3.1a)–(3.1p) with aggregated variables $\bar{B}_j$, $\bar{L}_i$ and Q_j instead of variables $\bar{B}^k_j$, $\bar{L}^k_i$ and Q^k_j (at every location except the start and end depot) and the substitutions of constraints (3.1g)-(3.1i) by constraints (3.4)-(3.8), (3.10) and (3.11) will be used as a benchmark in Chapter 4.

3.4.2 Two-Index Formulation

In this section, we discuss the two-index formulation proposed in Ropke et al. (2007) (additionally, we assume the number of vehicles to be restricted). The model is a standard two-index formulation of the DARP described in Section 3.3.

A binary variable $\bar{x}_{ij}$ is associated with each arc and attains a value of 1 if a vehicle drives directly from location i to location j, i.e., the arc (i, j) is used, and 0 otherwise. Moreover, Q_j represents the number of customers in the vehicle after leaving location j and $\bar{B}_j$ is the time when service at location j begins. In the model, $\mathcal{S}$ is the set of all sets S with $0^+ \in S$, $0^- \notin S$, and there is at least one request i for which the delivery node is in S but not the pickup node. This means $\mathcal{S} = \{S \colon 0^+ \in S \wedge 0^- \notin S \wedge \exists i : (i^+ \notin S \wedge i^- \in S)\}$. Moreover, $\bar{M}_{ij}$ with $i, j \in \bar{J}$ is a sufficiently large constant and can be chosen as in constraints (3.2). Then, the model is formulated as:

$$Z_{LB} = \min \sum_{i,j \in \bar{J}} \bar{c}_{ij} \cdot \bar{x}_{ij} \tag{3.12a}$$

$$\text{s. t.} \sum_{i \in \bar{J}} \bar{x}_{ij} = 1 \qquad \forall j \in P \cup D, \tag{3.12b}$$

$$\sum_{j \in \bar{J}} \bar{x}_{ij} = 1 \qquad \forall i \in P \cup D, \tag{3.12c}$$

$$\sum_{j \in P} \bar{x}_{0^+ j} \leq K, \tag{3.12d}$$

$$\sum_{i,j \in S} \bar{x}_{ij} \leq |S| - 2 \qquad \forall S \in \mathcal{S}, \tag{3.12e}$$

$$\bar{B}_j \geq \bar{B}_i + s_i + \bar{t}_{ij} - \bar{M}_{ij}\,(1 - \bar{x}_{ij}) \qquad \forall i, j \in \bar{J}, \tag{3.12f}$$

$$Q_j \geq Q_i + \mathcal{I}_j q_j - Q\,(1 - \bar{x}_{ij}) \qquad \forall i, j \in \bar{J}, \tag{3.12g}$$

$$e_j \leq \bar{B}_j \leq \ell_j \qquad \forall j \in \bar{J}, \tag{3.12h}$$

$$\max\{0, \mathcal{I}_j q_j\} \leq Q_j \leq \min\{Q, Q + \mathcal{I}_j q_j\} \qquad \forall j \in \bar{J}, \tag{3.12i}$$

$$\bar{B}_{i^-} - \bar{B}_{i^+} - s_i \leq L_i \qquad \forall i \in P, \tag{3.12j}$$

$$\bar{B}_j \geq 0 \qquad \forall j \in \bar{J}, \tag{3.12k}$$

$$Q_j \geq 0 \qquad \forall j \in \bar{J}, \tag{3.12l}$$

$$\bar{x}_{ij} \in \{0, 1\} \qquad \forall i, j \in \bar{J}. \tag{3.12m}$$

Objective function (3.12a) minimizes the total routing costs. Constraints (3.12b) and (3.12c) ensure that each customer location is visited and left exactly once. Due to constraint (3.12d) the depot is left at most K times, i.e., up to K vehicles are used. Constraints (3.12e) ensure that the pickup and the delivery location of each customer are visited in the same tour and in the correct order. Constraints (3.12f) and (3.12h) take care that each location is visited within its time window. Constraints (3.12g) model the vehicle load which is not allowed to exceed the vehicle's capacity because of constraints (3.12i). No customer is allowed to be longer in the vehicle than their maximum ride time L_i, which is ensured by constraints (3.12j). Finally, constraints (3.12m) are the binary constraints for the sequence variables.

As can be seen the size of the set $\mathcal{S}$ grows exponentially in the number of customers n. Unlike the three-index formulation described previously, the two-index formulations has an exponential number of constraints, but contains fewer variables and provides tighter bounds (see Ropke et al., 2007). In Chapter 5, this model will be used as a reference model for location-based formulations. Hence, we call it the location-based (LB) model.

3.5 Multiple Objectives

In most of the research on the DARP only one objective is used, which is often the minimization of total routing costs. An excellent overview is given in Ho et al. (2018). Other popular objectives are, for example, the minimization of total route duration, number of vehicles used, users' waiting time, drivers' working hours, or deviation from the desired pickup and delivery times. Motivated by real-life applications, dial-a-ride problems are in fact characterized by multiple (and in general conflicting) objectives. In this thesis, we focus on

- economic efficiency, e.g., minimization of routing costs, and

- user experience, e.g., minimization of unfulfilled user requests, waiting time, ride time and transportation time (e.g., the time from booking the trip until the arrival at the destination),

and combine these objectives into an overall *weighted-sum objective*. Note that, this weighted-sum objective can be interpreted as a scalarization of a multi-objective model. Consequently, by variation of the weights every supported efficient solution can be determined as a solution of the weighted-sum scalarization, see Section 2.3. Other options would be to consider a lexicographic multi-objective model or to treat one objective function, e.g., economic efficiency, as primary objective and to impose constraints on the other objectives to ensure a satisfactory user experience (in particular, the acceptance of all or a certain number of requests) which corresponds to the ε-constraint scalarization.

3.6 Related Problems

This section reviews several related problems which are of great interest throughout the literature—due to their practical relevance, but also due to the fact that they are universal in the sense that many other problems are a generalization or specialization of these. One big class of problems are routing problems, which are concerned with the determination of minimum cost routes to visit a set of locations according to defined constraints. The first part of this section describes the similarities and differences of the DARP and several (generalizations of) the traveling salesperson problem and the vehicle routing problem. In the second part, we describe the minimum cost flow problem, in which we would like to determine a least cost shipment of commodity through a network, while satisfying demands at certain nodes with available supplies from other nodes.

Routing Problems The *traveling salesperson problem* (TSP) (Dantzig et al., 1954) and the *vehicle routing problem* (VRP) (Dantzig and Ramser, 1959) are among the most studied combinatorial optimization problems.

The TSP is easy to state (see Ahuja et al., 1993): Starting from his or her home base, node 1, the salesperson wishes to visit each of several cities, represented by nodes $2, \ldots, n$, exactly once and return home at lowest possible travel cost. This problem is part of many other routing problems, such as the VRP.

The following definition of the (asymmetric) VRP is based on Cordeau et al. (2007). Let $G = (V, A)$ be a complete and connected graph with node set $V = \{0, 1, \ldots, n\}$ and arc set $A = \{(i, j) \colon i, j \in V, i \neq j\}$. Each node $i \in V \setminus \{0\}$ represents a customer with demand $q_i \geq 0$, while the node 0 represents the vehicle depot. With each arc $(i, j) \in A$ a cost $\bar{c}_{ij}$ and travel time $\bar{t}_{ij}$ is associated. We assume that $\bar{c}_{ij}$ and $\bar{t}_{ij}$ are non-negative and fulfill the triangle-inequality. A fleet of K identical vehicles with capacity Q is situated at the vehicle depot. A solution to the VRP consists of K vehicle routes such that travel costs are minimized, the vehicle routes start and end at the depot, each customer is visited exactly once by one vehicle, and the vehicle capacity and a preset limit T on the length of the routes is not violated. Hence, after assigning a set of customers to each vehicle, the problem of finding an optimal route visiting all customer locations is a TSP. The VRP is an $\mathcal{NP}$-hard problem, as is shown in Lenstra and Kan (1981). There is a vast number of

contributions over the past decades. Recent reviews are given in Toth and Vigo (2014) or Braekers et al. (2016).

Frequently, real-life applications ask for the incorporation of additional features, such as time-related constraints: In the *vehicle routing problem with time windows* (VRPTW), additionally, a time window is associated with each customer location. Another vehicle routing problem where each customer request is associated with a pickup and a delivery location is the *pickup and delivery problem with time windows* (PDPTW). These locations have to be visited in the correct order by the same vehicle. By setting each customer's pickup location as the vehicle depot and using a suitable pickup time window, the PDPTW is a generalization of the VRPTW. Surveys on the static and dynamic pickup and delivery problem are given in Berbeglia et al. (2007) and Berbeglia et al. (2010), respectively. A special case of the PDPTW is the *taxi routing problem* considered in Bertsimas et al. (2019), where vehicles can only transport one customer at a time and delivery time windows are omitted. The fact that rides are not shared, and pickup time windows for taxi routing are typically much smaller than in applications of the PDPTW or the DARP, allows the authors to assume a cycle-free graph structure. This assumption reduces the number of possible vehicle routes, and thus facilitates the fast computation of solutions, as we will also see in Section 5.2 by making similar assumptions for the DARP leading to a cycle-free graph structure. Generalizing even further, the DARP is a generalization of the PDPTW and includes additional ride time constraints. While vehicle routing and pickup and delivery problems are often motivated by the transportation of goods, the dial-a-ride problem is motivated by the (simultaneous) transportation of persons, which explains the need for additional ride time constraints.

The Minimum Cost Flow Problem The *minimum cost flow problem* (MCFP) has a variety of applications, such as the distribution of products from manufacturers to warehouses, or the flow of raw materials through stages of a production line. The MCFP is a well-studied problem of fundamental interest because other relevant combinatorial and network optimization problems such as the shortest path problem, the maximum flow problem or the assignment problem can be reduced to an MCFP. In the following, we describe the problem as given in Ahuja et al. (1993), where an in-depth summary of the MCFP and its variants can be found.

Let $G = (V, A)$ be a connected graph. Each arc $(v, w) \in A$ has an associated *cost* $c_{(v,w)}$ that denotes the cost per unit flow on that arc. We also associate with each arc $(v, w) \in A$ a *capacity* $u_{(v,w)}$ that denotes the maximum amount that can flow on the arc and a *lower bound* $l_{(v,w)}$ that denotes the minimum amount that must flow on the arc. We assume $l_{(v,w)} \leq u_{(v,w)}$ for all $(v, w) \in A$. With each node $v \in V$ we associate an integer number $b(v)$ representing its supply or demand. If $b(v) > 0$, node v is a *supply node*, if $b(v) < 0$, node v is a *demand node* with a demand of $-b(v)$ and if $b(v) = 0$, node v is a *transshipment node*. We assume $\sum_{v \in V} b(v) = 0$. A solution to the MCFP is a least cost flow through the network which satisfies all demands and supplies.

The minimum cost flow problem can be formulated as the following linear program,

where the variables $x_{(v,w)}$ for $(v,w) \in A$ represent arc flows.

$$\begin{aligned} \min \quad & \sum_{(v,w)\in A} c_{(v,w)} x_{(v,w)} \\ \text{s. t.} \quad & \sum_{w\colon (v,w)\in A} x_{(v,w)} - \sum_{w\colon (w,v)\in A} x_{(w,v)} = b(v) \quad \forall v \in V, \\ & l_{(v,w)} \leq x_{(v,w)} \leq u_{(v,w)} \quad \forall (v,w) \in A, \end{aligned}$$

where $\sum_{v\in V} b(v) = 0$.

In this optimization problem, the objective function minimizes the total cost of the flow. The first set of constraints ensures that the difference in inflow and outflow at a node $v \in V$ equals its demand or supply (flow conservation). The second set of constraints ensures that the flow satisfies the lower bound and capacity restrictions.

In many applications, there is only one supply node, the *source*, and one demand node, the *sink*, and all other nodes are transshipment nodes. If all nodes are transshipment nodes, i.e., if there is no external inflow or outflow, all flow circulates in the network, so that the problem is called *minimum cost circulation problem*.

For integral input data, we have the following important result:

Theorem 3.1. *If all arc capacities and supplies or demands of nodes are integer, the minimum cost flow problem always has an integer minimum cost flow.*

Proof. See Ahuja et al. (1993). □

As we will see in Section 4.1, the dial-a-ride problem can be modeled as a minimum cost circulation problem with additional constraints, such as time window and ride time constraints. Furthermore, we show in Section 5.2 that if the time windows fulfill certain conditions, the LP relaxation of an MILP modeling the DARP (presented in the same section) has only integral solutions. For the proof, we show that under the said conditions, the DARP can be modeled as an MCFP and use the previous theorem to prove the integrality of the LP relaxation.

4 Event-Based MILP Models for the Static Dial-a-Ride Problem

In this chapter we suggest an event-based formulation of the static DARP which is based on an abstract graph model and, in contrast to other approaches, has the advantage of implicitly encoding vehicle capacities as well as pairing and precedence constraints. The event-based graph model is the basis for two alternative mixed-integer linear programming formulations that can be distinguished by the approach towards handling ride time and time window constraints. The MILP formulations are compact, that is, the number of linear inequalities in the formulations is polynomial w.r.t. the number of variables. The presented models can be used to solve small- to medium-sized benchmark instances in a short amount of time. In particular, we show that both suggested MILP formulations outperform the standard compact three-index formulation proposed in Cordeau (2006) presented in Section 3.4 and the compact two-index formulation of the closely related PDPTW from Furtado et al. (2017), which can be extended to an MILP formulation of the DARP by introducing additional constraints. Furthermore, conflicting optimization goals reflecting, e.g., economic efficiency and customer satisfaction are combined into weighted-sum objectives. In addition, we distinguish between average case and worst case formulations. The impact of these objective functions is tested using artificial request data for the city of Wuppertal, Germany.

This chapter deals with the deterministic, homogeneous and static DARP (see Section 3.3). We assume that all requests have to be accepted and that the objective is to minimize the total routing costs, but we describe the option of rejecting unfavorable requests as well.

The chapter is organized as follows. Based on the concept of the event-based graph, which is introduced in Section 4.1, two variants of an MILP formulation of the DARP are presented in Section 4.2. Different objective functions, based on user- and service-provider-related criteria are discussed in Section 4.3. The models are compared and tested using CPLEX, in Section 4.4. The chapter is concluded with a summary in Section 4.5. The contents of this chapter are published in the paper Gaul et al. (2022).

4.1 Event-Based Graph

Classical models for the DARP are based on a directed graph that can be constructed in a straight-forward way by identifying all pickup and delivery locations and the depot by nodes in a complete graph, see, e.g., the location-based models described in Section 3.4. In contrast to this, we suggest an *event-based graph* $G = (V, A)$ in which the node set V consists of Q-tuples representing feasible user allocations together with information on the most recent pickup or delivery. As will be explained later, this modeling approach has the advantage that many of the complicating constraints can be encoded directly in the

network structure: vehicle capacity, pairing (i.e., users' pickup and delivery locations need to be served by the same vehicle) and precedence constraints (i.e., users' pickup locations need to be reached before their delivery locations are reached) are implicitly incorporated in the event-based graph. Moreover, a directed arc $a \in A$ is only introduced between a pair of nodes if the corresponding sequence of events is feasible w.r.t. the corresponding user allocation. Thus, a tour in G is feasible if it does not violate constraints regarding time windows or ride times. By identifying the depot with the source and sink of a circulation flow, the DARP can be modeled as a variant of a minimum cost flow problem with additional constraints.

The allocation of transport requests to a vehicle with capacity Q can be written as a Q-tuple. If less than Q requests are allocated to the vehicle, the remaining places are marked by zeros. Hence, a zero either indicates an empty seat, or that a single request contains more than one passenger. If, for example, $Q = 3$ and two requests, request 1 and request 2 with $q_1 = q_2 = 1$, are assigned to the same vehicle, the user allocation may be, for instance, represented by the tuple $(2, 1, 0)$. Accordingly, an empty vehicle is represented by $(0, 0, 0)$. To additionally incorporate information about the most recent pickup or delivery location visited, we write

$$(2^+, 1, 0) \quad \text{or} \quad (2^-, 1, 0),$$

if user 1 is seated and user 2 has just been picked up or dropped off, respectively. Note that this encoding implicitly specifies the location of the vehicle, since, in this example, user 2 is picked up at his or her pickup location 2^+, i.e., the 3-tuple $(2^+, 1, 0)$ can be associated with the location 2^+, and user 2 is dropped off at his or her delivery location 2^-, i.e., the 3-tuple $(2^-, 1, 0)$ can be associated with the location 2^-. Since all permutations of such a Q-tuple specify the same user allocation, we order the components of the Q-tuple such that the first component contains the information regarding the last pickup or delivery stop and the remaining $Q - 1$ components are sorted in descending order. Users can only be placed together in a vehicle if the total number of requested seats does not exceed the vehicle capacity. This constraint limits the possible combinations of users in the vehicle and hence the set of possible Q-tuples representing user allocations. As stated by Cordeau, 2006, time window and ride time constraints can be used to identify incompatible user pairs. For example, given users $i, j \in R$, the following reductions are possible:

- If it is infeasible to visit locations i^+, i^-, j^+, j^- both in the order $j^+ \to i^+ \to j^- \to i^-$ and $j^+ \to i^+ \to i^- \to j^-$, the set $\{(v_1, v_2, \ldots, v_Q) \in V \colon v_1 = i^+, j \in \{v_2, \ldots, v_Q\}\}$ can be removed from the node set.

- If it is infeasible to visit locations i^+, i^-, j^+, j^- both in the order $j^+ \to i^+ \to i^- \to j^-$ and $i^+ \to j^+ \to i^- \to j^-$, the set $\{(v_1, v_2, \ldots, v_Q) \in V \colon v_1 = i^-, j \in \{v_2, \ldots, v_Q\}\}$ can be removed from the node set.

The number of possible Q-tuples can be reduced significantly by adapting this strategy to identify incompatible user allocations. For this purpose, given requests $i, j \in R$, let $f^1_{i,j}$ and $f^2_{i,j}$ encode the feasibility of the paths $j^+ \to i^+ \to j^- \to i^-$ and $j^+ \to i^+ \to i^- \to j^-$,

respectively, w.r.t. ride time and time window constraints. To simplify the notation, let $f^1_{i,0} = f^1_{0,i} = f^2_{i,0} = f^2_{0,i} = 1$.

Now the DARP can be represented by a directed graph $G = (V, A)$, where the node set V represents events rather than geographical locations. The set of event nodes corresponds to the set of all feasible user allocations. The set of all nodes that represent an event in which a request (or user) $i \in R$ is picked up is called the *set of pickup nodes* and is given by

$$V_{i^+} \coloneqq \Bigg\{ (v_1, v_2, \ldots, v_Q) \colon v_1 = i^+,\ v_j \in R \cup \{0\} \setminus \{i\},\ f^1_{i,v_j} + f^2_{i,v_j} \geq 1\ \forall j \in \{2, \ldots, Q\},$$
$$\Big(v_j > v_{j+1} \vee v_{j+1} = 0 \Big)\ \forall j \in \{2, \ldots, Q-1\},\ \sum_{j=1}^{Q} q_{v_j} \leq Q \Bigg\}.$$

Similarly, the *set of delivery nodes* corresponds to events where a request (or user) $i \in R$ is dropped off and is given by

$$V_{i^-} \coloneqq \Bigg\{ (v_1, v_2, \ldots, v_Q) \colon v_1 = i^-,\ v_j \in R \cup \{0\} \setminus \{i\},\ f^1_{v_j,i} + f^2_{i,v_j} \geq 1\ \forall j \in \{2, \ldots, Q\},$$
$$\Big(v_j > v_{j+1} \vee v_{j+1} = 0 \Big)\ \forall j \in \{2, \ldots, Q-1\},\ \sum_{j=1}^{Q} q_{v_j} \leq Q \Bigg\}.$$

Note, that for each request $i \in R$ there is only one pickup and one delivery location, but in general more than one potential pickup node and delivery node. Hence, there is a unique mapping of nodes to locations, while a location may be associated with many different nodes. The depot-node $(0, \ldots, 0)$ is associated with the location of the depot, and we write $V_0 \coloneqq \{(0, \ldots, 0)\}$. The overall set of nodes V is then given by

$$V = V_0 \cup \bigcup_{i=1}^{n} V_{i^+} \cup \bigcup_{i=1}^{n} V_{i^-}.$$

The arc set A of G is defined by the set of possible transits between pairs of event nodes in V. It is composed of six subsets, i.e.,

$$A = \bigcup_{i=1}^{6} A_i,$$

where the subsets A_i, $i = 1, \ldots, 6$ are defined as follows:

- Arcs that describe the transit from a pickup node in a set V_{i^+}, i.e., from a user i's pickup location, to a delivery node in V_{j^-}, i.e., to the delivery location of a user j, where $j = i$ is possible, but where j may also be another user from the current passengers in the vehicle:

$$A_1 \coloneqq \Big\{ \big((i^+, v_2, \ldots, v_Q), (j^-, w_2, \ldots, w_Q) \big) \in V \times V \colon$$
$$\{j, w_2, \ldots, w_Q\} = \{i, v_2, \ldots, v_Q\} \Big\}.$$

- Arcs that describe the transit from a pickup node in a set V_{i^+}, i.e., from a user i's pickup location, to another pickup node from a set V_{j^+} with $j \neq i$, i.e., to another user j's pickup location:

$$A_2 \coloneqq \Big\{\big((i^+, v_2, \dots, v_{Q-1}, 0), (j^+, w_2, \dots, w_Q)\big) \in V \times V : \{i, v_2, \dots, v_{Q-1}\} = \{w_2, \dots, w_Q\}\Big\}.$$

- Arcs that describe the transit from a delivery node in a set V_{i^-}, i.e., from a user i's delivery location, to a pickup node in a set V_{j^+}, $j \neq i$, i.e., to another user j's pickup location:

$$A_3 \coloneqq \Big\{\big((i^-, v_2, \dots, v_Q), (j^+, v_2, \dots, v_Q)\big) \in V \times V : i \neq j\Big\}.$$

- Arcs that describe the transit from a delivery node in a set V_{i^-}, i.e., from a user i's delivery location, to a node in V_{j^-}, $j \neq i$, i.e., to another user j's delivery location:

$$A_4 \coloneqq \Big\{\big((i^-, v_2, \dots, v_Q), (j^-, w_2, \dots, w_{Q-1}, 0)\big) \in V \times V : \{v_2, \dots, v_Q\} = \{j, w_2, \dots, w_{Q-1}\}\Big\}.$$

- Arcs that describe the transit from a delivery node in a set V_{i^-}, i.e., from a user i's delivery location, to the depot:

$$A_5 \coloneqq \Big\{\big((i^-, 0, \dots, 0), (0, \dots, 0)\big) \in V \times V\Big\}.$$

- Arcs that describe the transit from the depot to a pickup node in a set V_{i^+}, i.e., to a user i's pickup location:

$$A_6 \coloneqq \Big\{\big((0, \dots, 0), (i^+, 0, \dots, 0)\big) \in V \times V\Big\}.$$

Example 4.1. *We give an example of the event-based graph $G = (V, A)$ with three users and vehicle capacity $Q = 3$. Let $R = \{1, 2, 3\}$, $q_1 = q_2 = 1$ and $q_3 = 3$. We omit time windows, travel and ride times in this example. By the above definitions we obtain the graph illustrated in Figure 4.1. Note that there are no nodes $v \in V$ that simultaneously contain users 1 (i.e., 1^+, 1^- or 1) and 3 (i.e., 3^+, 3^- or 3) as the total number of requested seats specified by these users exceeds the vehicle capacity. Similarly, the seats requested by users 2 and 3 together exceed the vehicle capacity of three. Two feasible tours for a vehicle in G are given, for example, by the dicycles*

$$(0,0,0) \to (1^+,0,0) \to (2^+,1,0) \to (2^-,1,0) \to (1^-,0,0) \to (0,0,0)$$

and

$$(0,0,0) \to (3^+,0,0) \to (3^-,0,0) \to (0,0,0).$$

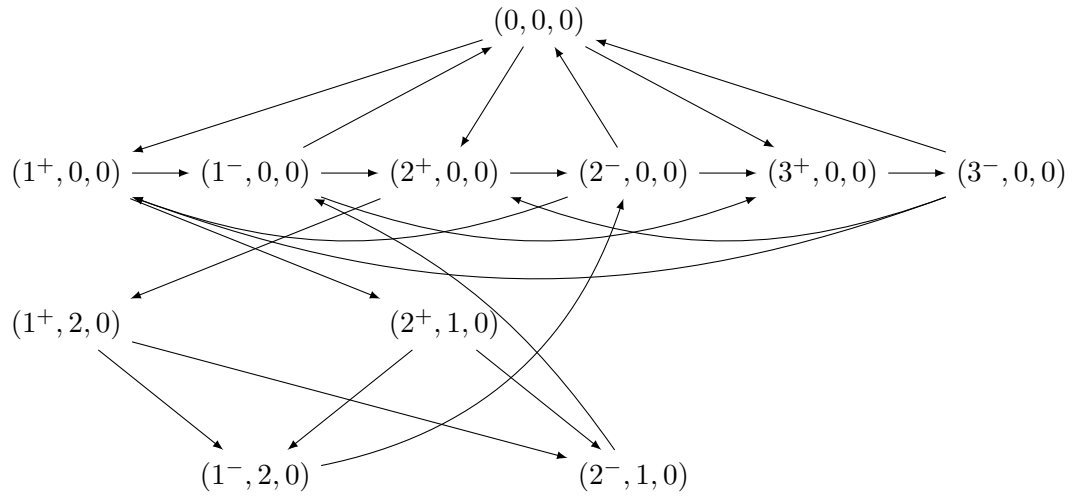


Figure 4.1: Graph representation of an example with three users.

In order to evaluate the worst case complexity of this event-based graph representation of the DARP, we first evaluate the respective cardinalities of the node set V and of the arc set A. Note that, the number of nodes and arcs in the event-based graph model depends on the vehicle capacity, the number of users and the number of requested seats per user and can potentially be reduced based on, e.g., precedence or time window constraints. Given Q and n, the number of nodes and arcs is maximal if all users request only one seat, i.e., if $q_i = 1$ for all $i \in R$. Thus, we obtain the worst case bound

$$|V| \leq 1 + 2\,n \sum_{j=0}^{Q-1} \binom{n-1}{j},$$

where

- "1" corresponds to the depot node,
- the factor $2n$ corresponds to the first component of the nodes being equal to i^+ or i^- with $i \in R = \{1, \ldots, n\}$,
- and the sum corresponds to the allocation of users to the remaining seats $v_2, \ldots, v_Q$, of which $j \in \{0, \ldots, Q-1\}$ are occupied.

For the arc set, we obtain the worst case bound

$$|A| \leq\ 2\,n + n \sum_{j=0}^{Q-1} \binom{n-1}{j}(j+1) + 3\,n\,(n-1) \sum_{j=0}^{Q-2} \binom{n-2}{j} + \frac{n(n-1)\cdot\ldots\cdot(n-Q)}{(Q-1)!}.$$

This can be seen by counting the number of outgoing arcs for each node with $Q-j+1$ zero entries and adding them up for $j \in \{0, \ldots, Q-1\}$, together with $2n$ arcs corresponding to the depot node. We use the convention that $\binom{m}{k} := 0$ when $k > m$. From the above formulas, we deduce that the number of nodes is bounded by $\mathcal{O}(n^Q)$ for $n \geq Q$ and the number

of arcs is bounded by $\mathcal{O}(n^{Q+1})$ for $n \geq Q + 1$. This is in general considerably more than what is obtained when a location-based DARP model on a complete graph is used, which has only $\mathcal{O}(n)$ nodes and $\mathcal{O}(n^2)$ arcs, see Section 3.4. However, in practice, ridepooling services are usually operated by taxis or mini-buses, so that $Q \in \{3, 6\}$. Moreover, the number of nodes and thereby the number of arcs, reduce substantially if we do not consider the "worst-case-scenario" $q_i = 1$ for all $i \in R$, in which all combinations of requests are possible user allocations to vehicles, and if we consider time windows, travel and ride times to determine infeasible user pairs. Besides that, the event-based formulation has the clear advantage that important constraints like vehicle capacity constraints, pairing constraints and precedence constraints that have to be formulated in classical models are implicitly handled using the event-based graph model, as will be seen in the next section.

4.2 Event-Based MILP Models

With the above definitions, the DARP can be modeled as a variant of a minimum cost flow problem with unit flows and with additional constraints. In particular, we consider circulation flows in G and identify the depot with the source and the sink of a minimum cost flow problem. Each dicycle flow in G then represents one vehicle's tour. We formulate corresponding MILP models in the subsequent subsections: The first event-based MILP is presented in Subsection 4.2.1. A reformulation of time window and ride time constraints described in Subsection 4.2.2 yields the second event-based MILP formulation, defined in Subsection 4.2.3.

The event-based MILP formulations are motivated by the work of Bertsimas et al. (2019) who propose an optimization framework for taxi routing, where only one passenger is transported at a time. Their algorithm can handle more than $25,000$ users per hour. Bertsimas et al. (2019) propose a graph-based formulation in which an arc (i, j) represents the decision to serve passenger j directly after dropping off passenger i.

Before we describe the first MILP formulation, we introduce some additional parameters and variables.

Since each node in V can be associated with a unique request location $j \in J$, we can associate a routing cost c_a and a travel time t_a with each arc $a = (v, w) \in A$. More precisely, both values c_a and t_a are calculated by evaluating the actual routing cost and travel time from the location associated with v to the location associated with w. We assume that all routing costs and all travel times are non-negative and satisfy the triangle inequality. Finally, let $\delta^{\text{in}}(v) \coloneqq \{(w, v) \in A\}$ and $\delta^{\text{out}}(v) \coloneqq \{(v, w) \in A\}$ denote the set of ingoing arcs of v and the set of outgoing arcs of v, respectively. Moreover, for $a \in A$ let the variable value $x_a = 1$ indicate that arc a is used by a vehicle, and let $x_a = 0$ otherwise. Thus, a vehicle tour is represented by a sequence of events in a dicycle C in G where $x_a = 1$ for all $a \in A(C)$, where $A(C)$ denotes the arc set of C. A request is matched with a vehicle if the vehicle's tour, i.e., the sequence of events in the corresponding dicycle, contains the event of picking-up and dropping-off of the corresponding user. Let the variable B_v store the information on the beginning of service time at node $v \in V$. Recall that the beginning of service time has to be within the associated time window of the respective location of

node v. Since we also consider the case that allows users' requests to be denied, let the variable value $p_i = 1$ indicate that request $i \in R$ is accepted, and let $p_i = 0$ otherwise.

Based on the event-based graph model, we are now ready to formulate our first MILP model for the DARP.

4.2.1 Basic Event-Based MILP for the DARP

In this subsection, we propose an event-based mixed integer linear program for the DARP. First, we present a non-linear mixed-integer programming formulation, which is based on the event-based graph model presented in Section 4.1. In a second step, this model is transformed into an MILP by a reformulation of time window and ride time constraints, i.e., constraints involving the variables B_v, $v \in V$.

The DARP can be formulated as the following non-linear mixed integer program:

$$\min \quad \sum_{a \in A} c_a \, x_a \tag{4.1a}$$

$$\text{s. t.} \quad \sum_{a \in \delta^{\text{in}}(v)} x_a - \sum_{a \in \delta^{\text{out}}(v)} x_a = 0 \quad \forall v \in V, \tag{4.1b}$$

$$\sum_{\substack{a \in \delta^{\text{in}}(v) \\ v \in V_{i^+}}} x_a = 1 \quad \forall i \in R, \tag{4.1c}$$

$$\sum_{a \in \delta^{\text{out}}((0,\dots,0))} x_a \leq K, \tag{4.1d}$$

$$B_w \geq (B_v + s_{v_1} + t_{(v,w)}) \, x_{(v,w)} \quad \forall (v,w) \in A, \, v \neq (0,\dots,0) \tag{4.1e}$$

$$B_w \geq e_0 + t_{(v,w)} \, x_{(v,w)} \quad \forall ((0,\dots,0),w) \in A, \tag{4.1f}$$

$$e_j \leq B_v \leq \ell_j \quad \forall j \in J, \, v \in V_j \tag{4.1g}$$

$$(B_w - B_v - s_{i^+}) \sum_{a \in \delta^{\text{in}}(v)} x_a \sum_{a \in \delta^{\text{in}}(w)} x_a \leq L_i \quad \forall i \in R, \, v \in V_{i^+}, \, w \in V_{i^-}, \tag{4.1h}$$

$$x_a \in \{0,1\} \quad \forall a \in A. \tag{4.1i}$$

The objective function (4.1a) minimizes the total routing cost. While constraints (4.1b) are flow conservation constraints, it is ensured by constraints (4.1c) that each request $i \in R$ is accepted and that exactly one node of all nodes which contain the request's pickup location is reached by exactly one vehicle. Together with constraints (4.1b) and (4.1c), the number of feasible dicycles in G is bounded from above by the number of vehicles in constraints (4.1d). Constraints (4.1e) and (4.1f) define the difference in time needed to travel from one node to another. For all nodes in V the start of service has to take place within the time window corresponding to the associated location of the node, which is handled by constraints (4.1g). An upper bound on the ride time is ensured by constraints (4.1h). Note that, we only impose a bound on the variables B_w, B_v, if both $v \in V_{i^+}$ and $w \in V_{i^-}$ are in fact the pickup and delivery nodes that are used to serve request i. Vehicle capacity, pairing and precedence constraints are ensured by the structure of the underlying network.

This formulation is non-linear due to constraints (4.1e) and (4.1h). In the following MILP these constraints are substituted by a linearized reformulation.

Model I.

$$\min \quad \sum_{a \in A} c_a \, x_a \tag{4.2a}$$

$$\text{s.t.} \quad \text{constraints (4.1b)} - \text{(4.1d)}$$

$$B_w \geq B_v + s_{v_1} + t_{(v,w)} - \tilde{M}_{v,w} \, (1 - x_{(v,w)}) \quad \forall (v,w) \in A,\, v \neq (0,\ldots,0), \tag{4.2b}$$

$$B_w \geq e_0 + t_{(v,w)} \, x_{(v,w)} \quad \forall ((0,\ldots,0),w) \in A, \tag{4.2c}$$

$$e_j \leq B_v \leq \ell_j \quad \forall j \in J,\, v \in V_j \tag{4.2d}$$

$$B_w - B_v - s_{i^+} \leq L_i + M_i \left(1 - \sum_{a \in \delta^{in}(v)} x_a + 1 - \sum_{a \in \delta^{in}(w)} x_a \right) \forall i \in R,\, v \in V_{i^+},\, w \in V_{i^-}, \tag{4.2e}$$

$$x_a \in \{0,1\} \quad \forall a \in A. \tag{4.2f}$$

where $M_i = \ell_{i^-} - e_{i^+} - L_i - s_{i^+}$ *and* $\tilde{M}_{v,w} = \ell_{v_1} - e_{w_1} + s_{v_1} + t_{(v,w)}$ *are sufficiently large constants.*

To include the option to deny requests, variables $p_i \in \{0,1\}$, $i \in R$ have to be added to Model I and constraints (4.1c) have to be changed to

$$\sum_{\substack{a \in \delta^{\text{in}}(v) \\ v \in V_{i^+}}} x_a = p_i \qquad \forall i \in R. \tag{4.3}$$

Hence, if a user is not picked up (i.e., if $p_i = 0$), then none of the nodes which contain his or her pickup location are traversed by any vehicle. Note that, in this case a reasonable objective function (see Section 4.3) has to penalize the denial of user requests since otherwise an optimal solution is given by $p = 0$, $x = 0$ and $B_v = e_{v_1}$ for all $v \in V$. In the computational experiments in Section 4.4 we consider both cases, i.e., the scenario that all users have to be served and the scenario that some requests may be denied.

Assuming $q_i = 1$ for all requests $i \in R$ and $n \geq Q + 1$ the total number of variables in Model I can be bounded by $\mathcal{O}(n^{Q+1})$ with $\mathcal{O}(n^{2Q-1})$ constraints, of which $\mathcal{O}(n^{2Q-1})$ constraints are ride time constraints (4.2e). If $q_i \in \{2,\ldots,Q\}$ for some requests, the number of variables and constraints decreases.

In each of the ride time constraints in Model I, the sums $\sum_{a \in \delta^{\text{in}}(v)} x_a$ and $\sum_{a \in \delta^{\text{in}}(w)} x_a$ are evaluated. This is computationally expensive, as will be demonstrated in the computational tests carried out in Section 4.3. By taking advantage of the relationship between the pickup and delivery time windows associated with request i, we show in the following how constraints (4.2e) can be reformulated without using big-M constraints, resulting in a second MILP formulation referred to as Model II.

4.2.2 Reformulation of Time-Related Constraints

In Model I, the ride time constraints are modeled as big-M constraints that are used to deactivate the respective constraints for pickup and delivery nodes that are not contained in a vehicle's tour. As shown in this subsection, by reformulating the time window constraints and using the relationship between earliest pickup and latest delivery times, the numerically unfavorable big-M constraints can be replaced by simpler constraints in the following model (Model II), see Subsection 4.2.3. This model is faster to solve which is verified by the numerical experiments presented in Section 4.4.

In Model II, the ride time constraints, which ensure that a user does not spend more than L_i minutes in the vehicle, are given by

$$B_w - B_v - s_{i^+} \leq L_i \qquad \forall i \in R, \forall v \in V_{i^+}, \forall w \in V_{i^-}. \tag{4.4}$$

To show that these constraints, together with a reformulation of constraints (4.2d), reflect the modeling assumptions, we first observe that in general applications of the DARP, as described for example in Cordeau (2006), users often formulate *inbound* requests and *outbound* requests. In the first case, users specify a desired departure time from the origin, while in the case of an outbound request, users specify a desired arrival time at the destination. In both cases a time window of a fixed length TW is constructed from the desired time, so that we end up with a pickup time window of length $TW = \ell_{i^+} - e_{i^+}$ for an inbound request and a delivery time window of length $TW = \ell_{i^-} - e_{i^-}$ for an outbound request. Now the remaining time window is constructed as follows (based on Cordeau, 2006): For an inbound request the delivery time window is given by the bounds

$$e_{i^-} = e_{i^+} + s_{i^+} + \bar{t}_i \qquad \text{and} \qquad \ell_{i^-} = \ell_{i^+} + s_{i^+} + L_i. \tag{4.5}$$

Similarly, for an outbound request the pickup time window is defined by

$$e_{i^+} = e_{i^-} - L_i - s_{i^+} \qquad \text{and} \qquad \ell_{i^+} = \ell_{i^-} - \bar{t}_i - s_{i^+}. \tag{4.6}$$

Secondly, we define the notion of an *active* node $v \in V$: We say that a node $v \in V$ is *active* if at least one of its ingoing arcs is part of a dicycle flow, i.e., if

$$\sum_{a \in \delta^{\text{in}}(v)} x_a = 1.$$

Otherwise, we call v *inactive*. Note that, due to constraints (4.1b)–(4.1c) and the structure of the event-based graph, for each request $i \in R$ we have exactly one associated active pickup node and one associated active delivery node. In case we include the option of denying user requests, i.e., we use constraints (4.3) instead of constraints (4.1c) and add variables $p_i \in \{0, 1\}$, $i \in R$ to the MILP, each request either has exactly one active pickup and delivery node (in this case we have $p_i = 1$), or no associated active node at all (this is the case when $p_i = 0$).

Now, if both v and w in constraints (4.4) are active nodes, inequalities (4.4) and (4.2e) coincide. In case both v and w are inactive, the values B_v, B_w can be ignored in an interpretation of an optimal solution, as v and w are not contained in any of the vehicle

tours. Hence, the critical two cases are the cases where one of the nodes is active and the other node is inactive. Let v^{off} and v^{on} denote the inactive and the active node from the set $\{v, w\}$, respectively. Then, we do not want that $B_{v^{\text{off}}}$ influences the value of $B_{v^{\text{on}}}$ in constraints (4.4):

Case 1: v is active, w is inactive. Resolving (4.4) for B_v we obtain $B_v \geq B_w - L_i - s_{i^+}$. Now, we do not want to impose any additional constraints on B_v. Thus, we demand $B_w - L_i - s_{i^+} \leq e_{i^+}$. Accordingly, it has to hold that

$$B_w \leq e_{i^+} + L_i + s_{i^+}.$$

Recall that here we assume w to be inactive. If w is active, the weaker constraint $B_w \leq \ell_{i^-}$ needs to hold. Putting these restrictions together for an inbound request, we get

$$\begin{aligned} B_w &\leq e_{i^+} + L_i + s_{i^+} + (\ell_{i^-} - (e_{i^+} + L_i + s_{i^+})) \sum_{a \in \delta^{\text{in}}(w)} x_a \\ &= e_{i^+} + L_i + s_{i^+} + (\ell_{i^+} + L_i + s_{i^+} - (e_{i^+} + L_i + s_{i^+})) \sum_{a \in \delta^{\text{in}}(w)} x_a \\ &= e_{i^+} + L_i + s_{i^+} + (\ell_{i^+} - e_{i^+}) \sum_{a \in \delta^{\text{in}}(w)} x_a \\ &= e_{i^+} + L_i + s_{i^+} + TW \sum_{a \in \delta^{\text{in}}(w)} x_a, \end{aligned} \tag{4.7}$$

using the reformulation of ℓ_{i^-} from equations (4.5). In the same manner, we use equations (4.6) to substitute $e_{i^+} = e_{i^-} - L_i - s_{i^+}$ and obtain that

$$B_w \leq e_{i^+} + L_i + s_{i^+} + (\ell_{i^-} - e_{i^-}) \sum_{a \in \delta^{\text{in}}(w)} x_a = e_{i^+} + L_i + s_{i^+} + TW \sum_{a \in \delta^{\text{in}}(w)} x_a \tag{4.8}$$

has to hold for an outbound request. Hence the two formulations (4.7) and (4.8) coincide.

Case 2: v is inactive, w is active. Resolving (4.4) for w we obtain $B_w \leq L_i + B_v + s_{i^+}$. In order for the latter inequality to be redundant for B_w, we demand that $L_i + B_v + s_{i^+} \geq \ell_{i^-}$. It follows that $B_v \geq \ell_{i^-} - L_i - s_{i^+}$ has to hold. For an inbound request, using the equivalence from (4.5), this can be resolved to

$$B_v \geq (\ell_{i^+} + L_i + s_{i^+}) - L_i - s_{i^+} = \ell_{i^+}.$$

Recall that we assumed that v is inactive. In case v is active, the less tighter constraint $B_v \geq e_{i^+}$ has to hold, so that we arrive at

$$B_v \geq e_{i^+} + (\ell_{i^+} - e_{i^+}) \left(1 - \sum_{a \in \delta^{\text{in}}(v)} x_a\right) = e_{i^+} + TW \left(1 - \sum_{a \in \delta^{\text{in}}(v)} x_a\right).$$

In a similar fashion, we obtain

$$B_v \geq e_{i^+} + (\ell_{i^-} - e_{i^-}) \left(1 - \sum_{a \in \delta^{\text{in}}(v)} x_a\right) = e_{i^+} + TW \left(1 - \sum_{a \in \delta^{\text{in}}(v)} x_a\right)$$

for an outbound request. We conclude that the two lower bounds on B_v coincide.

4.2.3 Event-Based MILP for DARP

As desired, by reformulating the constraints (4.2d) on the variables B_v, $v \in V$, we obtain a simpler version of the ride time constraints (4.2e). We put these results together in a second MILP formulation of the DARP.

Model II.

$$\min \quad \sum_{a \in A} c_a \, x_a \tag{4.9a}$$

$$\text{s. t.} \quad \text{constraints (4.1b)} - \text{(4.1d)}$$

$$B_w \geq B_v + s_{v_1} + t_{(v,w)} - \tilde{M}_{v,w} \, (1 - x(v,w)) \quad \forall (v,w) \in A, \, v \neq (0, \ldots, 0) \tag{4.9b}$$

$$B_w \geq e_0 + t_{(v,w)} \, x_{(v,w)} \quad \forall ((0, \ldots, 0), w) \in A, \tag{4.9c}$$

$$e_0 \leq B_{(0,\ldots,0)} \leq \ell_0, \tag{4.9d}$$

$$e_{i^+} + TW \left(1 - \sum_{a \in \delta^{in}(v)} x_a\right) \leq B_v \leq \ell_{i^+} \quad \forall i \in R, \, v \in V_{i^+}, \tag{4.9e}$$

$$e_{i^-} \leq B_v \leq e_{i^+} + L_i + s_{i^+} + TW \sum_{a \in \delta^{in}(v)} x_a \quad \forall i \in R, \, v \in V_{i^-}, \tag{4.9f}$$

$$B_w - B_v - s_{i^+} \leq L_i \quad \forall i \in R, \, v \in V_{i^+}, \, w \in V_{i^-}, \tag{4.9g}$$

$$x_a \in \{0,1\} \quad \forall a \in A. \tag{4.9h}$$

We substitute ride time constraints (4.2e) for a simpler version (4.9g). In return we use a more complex version of the time window constraints (4.9d) – (4.9f) instead of the short form (4.2d). Similar to Model I, variables $p_i \in \{0,1\}$, $i \in R$ may be added to Model II and constraints (4.1c) may be substituted by constraints (4.3) to include the option of denying user requests. In this case, the objective function should be modified to contain a term penalizing the denial of requests.

As there are $\mathcal{O}(n^{2Q-1})$ ride time constraints, for each of which two sums $\sum_{a \in \delta^{\text{in}}(v)} x_a$ have to be evaluated in the longer version (4.2e), but only $\mathcal{O}(n^Q)$ time window constraints, for each of which one sum of the above form has to be evaluated in the longer version (4.9d)-(4.9f), we obtain a more efficient new MILP formulation. Similar to Model I, for $n \geq Q + 1$ there are at most $\mathcal{O}(n^{Q+1})$ variables and at most $\mathcal{O}(n^{2Q-1})$ constraints, of which $\mathcal{O}(n^{2Q-1})$ constraints are ride time constraints (4.9g).

4.3 Objective Functions

As stated in Section 3.5, DARPs are characterized by multiple (and often conflicting) objectives. In the following, we focus on three prevalent criteria, namely the total routing cost, the total number of unanswered requests and the total users' regret or the maximum regret, and combine these three criteria into weighted-sum objective functions. The first and probably most important criterion is the total routing cost, which can be computed as

$$f_c(x) \coloneqq \sum_{a \in A} c_a \, x_a. \tag{4.10}$$

We refer to f_c as *cost-objective*. The second optimization criterion relates to customer satisfaction: We measure the response time to a service request by assessing a user's regret, which aims at penalizing overly long travel times as well as possibly delayed pick-up times. Let the variable $d_i \geq 0$, $i \in R$ measure the difference in time compared to a user's earliest possible arrival time. We refer to d_i as a user's *regret*. Moreover, let the variable $d_{\max} \geq 0$ measure the *maximum regret*. By introducing constraints

$$d_i \geq B_v - e_{i^-} \qquad \forall i \in R, \forall v \in V_{i^-}, \tag{4.11}$$

$$d_{\max} \geq d_i \qquad \forall i \in R, \tag{4.12}$$

we can now minimize the total or average regret, or the maximum regret, respectively. The total regret is thus given by the *regret-objective*

$$f_r(d) \coloneqq \sum_{i \in R} d_i, \tag{4.13}$$

while the *maximum-regret-objective* is given by

$$f_{r_{\max}}(d_{\max}) \coloneqq d_{\max}. \tag{4.14}$$

The discussion above highlights the fact that we have to consider different and generally conflicting objective functions that are relevant when solving the DARP. While the cost objective f_c aims at minimizing total travel cost and thus takes the perspective of the service provider, the quality of service which rather takes a user's perspective is better captured by objective functions like f_r and $f_{r_{\max}}$. We approach this technically bi-objective problem by using a weighted-sum approach with fixed weights, i.e., by combining the relevant objective functions into one weighted-sum objective. When using the total regret as quality criterion, we obtain

$$f_{cr}(x, d) \coloneqq \sum_{a \in A} c_a \, x_a + \alpha \sum_{i \in R} d_i \tag{4.15}$$

which will be referred to as *cost-regret-objective*, and when using the maximum regret as quality criterion, we get

$$f_{cr_{\max}}(x, d_{\max}) \coloneqq \sum_{a \in A} c_a \, x_a + \beta d_{\max} \tag{4.16}$$

which will be referred to as *cost-max-regret-objective*. The parameters $\alpha > 0$ and $\beta > 0$ are weighting parameters that can be selected according to the decision maker's preferences.

Last but not least, we consider a variant of the DARP in which it is allowed to deny certain user requests. This is accomplished by substituting constraints (4.1c) in Model I or Model II , respectively, by constraints (4.3) and adding variables $p_i \in \{0, 1\}$, $i \in R$ to Model I or II. In this case, the number of accepted requests has to be maximized or, equivalently, the number of unanswered requests has to be minimized. The objective function

$$f_n(p) \coloneqq n - \sum_{i \in R} p_i,$$

measures the total number of declined requests. At the same time, routing costs and regret should be as small as possible. The optimization of these opposing criteria is reflected by the *request-cost-regret-objective* given by

$$f_{rcr}(x, d, p) \coloneqq \sum_{a \in A} c_a \, x_a + \alpha \sum_{i \in R} d_i + \gamma \left(n - \sum_{i \in R} p_i \right), \tag{4.17}$$

where $\gamma > 0$ is an additional weighting parameter. While the third part of the objective refers to the number of unanswered requests and is equal to $\gamma \, n$ at maximum (i.e., if no requests are accepted), the values of the total routing costs and of the total regret strongly depend on the underlying network and request data. Note that, meaningful choices of the weighting parameters have to reflect this in order to avoid situations where one part of the objective overrides the others. This will be discussed in Section 4.4.

4.4 Numerical Results

This section is divided into two parts. In the first subsection, we compare the MILP formulations Model I and II to the MILP model from Cordeau (2006) presented in Section 3.4 and to a compact two-index formulation of the PDPTW by Furtado et al. (2017), which can be adapted to an MILP formulation of the DARP by using additional constraints. The computational performance is evaluated on a set of benchmark test instances. In the second part, we substitute the cost objective function in Model II with different objective functions as introduced in Section 4.3 and analyze the effect with respect to economic efficiency and customer satisfaction. For this purpose, a set of 60 artificial instances from the city of Wuppertal is generated. The computations are carried out on an Intel Core i7-8700 CPU, 3.20 GHz, 32 GB memory using CPLEX 12.10. The MILPs are programmed using C++ and the code can be found in the git repository Gaul (2022a). The time limit for the solution in all tests is set to 7200 seconds. In the following, the computational results are discussed in detail. We use the abbreviations listed in Table 4.1. When computing the average CPU times over several runs, and an instance was not solved to optimality within the time limit, then a CPU time of 7200 seconds is assumed.

4.4.1 Benchmark Data

We compare our MILP models to the model from Cordeau (2006), referred to as C-DARP for the remaining part of this section. Note that, the two-index formulation given by Ropke

Abbreviation	Explanation/ Reference
Inst.	Name of instance
Obj.	Objective value
CPU	Computational time in seconds
N/A	Not applicable, no integer solution found within the time limit
Gap	Gap obtained by CPLEX solution (in percent)
f_c	Total routing costs
f_r	Total regret
$f_{r_{\max}}$	Maximum regret
AR	Answered requests (in percent)
Δf_c	Change in total routing costs (in percent)
Δf_r	Change in total regret (in percent)
$\Delta f_{r_{\max}}$	Change in maximum regret (in percent)
Δ AR	Change in answered requests (in percent)

Table 4.1: List of abbreviations.

et al. (2007) is tighter (compare Section 3.4 for both models). However, this comes at the price of an exponentially growing number of constraints. It is thus better suited for a solution within a B&C framework and we did not include it in our comparison. Moreover, we compare our model to a compact two-index formulation of the PDPTW, introduced by Furtado et al. (2017), which can be adapted to an MILP formulation of the DARP by adding constraints on the passengers' maximum ride time, the maximum duration of service and the maximum number of vehicle tours. We refer to this model as F-DARP in the following. Both models, C-DARP and F-DARP, are based on a complete directed graph. The node set comprises all pickup and delivery locations and two additional nodes 0^+ and 0^- for the depot. Thus, the node set is equal to the set $\bar{J}$. The objective is to minimize the total routing costs. We do not add any additional valid inequalities described in Cordeau (2006) to any of the MILP formulations in this comparison. We use the following trivial variable fixings: $\bar{x}_{j,j} = 0$, $\bar{x}_{j,0^+} = 0$ and $\bar{x}_{0^-,j} = 0$ for all $j \in P \cup D$, $\bar{x}_{0^+,i^-} = 0$, $\bar{x}_{i^+,0^-} = 0$ and $\bar{x}_{i^-,i^+} = 0$ for all $i \in P$. Besides that, we use time window, capacity, and ride time constraints to fix the following variables: Let $i, j \in P$, then $\bar{x}_{i^+,j^+} = 0$ if $f^1_{j,i} = f^2_{j,i} = 0$, $x_{i^-,j^-} = 0$ if $f^1_{j,i} = f^2_{i,j} = 0$, and $\bar{x}_{i^+,j^-} = 0$ if $f^1_{i,j} = 0$.

The test instances are the two sets of benchmark instances[1], set a and set b, created in Cordeau (2006). The instances are denoted as aK-n and bK-n, where K indicates the number of vehicles and n denotes the number of requests. The characteristics of the instances are summarized in Table 4.2. In all test instances we tighten the remaining time windows, i.e., the time windows not given by the pickup time of inbound requests or by the delivery time of outbound requests, respectively, as described in Cordeau (2006): The bounds of the missing time windows can be calculated according to equations (4.5) and (4.6) stated earlier in Section 4.2.

A summary of the computational results can be found in Tables 4.3 and 4.4. For each of the considered models C-DARP, F-DARP, Model I and II, the objective value (Obj.)

[1]The instances are available at `http://neumann.hec.ca/chairedistributique/data/darp/branch-and-cut/`.

Inst.	Q	n	K	L_i	T
a2-16	3	16	2	30	480
a2-20	3	20	2	30	600
a2-24	3	24	2	30	720
a3-18	3	18	3	30	360
a3-24	3	24	3	30	480
a3-30	3	30	3	30	600
a3-36	3	36	3	30	720
a4-16	3	16	4	30	240
a4-24	3	24	4	30	360
a4-32	3	32	4	30	480
a4-40	3	40	4	30	600
a4-48	3	48	4	30	720
a5-40	3	40	5	30	480
a5-50	3	50	5	30	600
a5-60	3	60	5	30	720
a6-48	3	48	6	30	480
a6-60	3	60	6	30	600
a6-72	3	72	6	30	720
a7-56	3	56	7	30	480
a7-70	3	70	7	30	600
a7-84	3	84	7	30	720
a8-64	3	64	8	30	480
a8-80	3	80	8	30	600
a8-96	3	96	8	30	720

Inst.	Q	n	K	L_i	T
b2-16	6	16	2	45	480
b2-20	6	20	2	45	600
b2-24	6	24	2	45	720
b3-18	6	18	3	45	360
b3-24	6	24	3	45	480
b3-30	6	30	3	45	600
b3-36	6	36	3	45	720
b4-16	6	16	4	45	240
b4-24	6	24	4	45	360
b4-32	6	32	4	45	480
b4-40	6	40	4	45	600
b4-48	6	48	4	45	720
b5-40	6	40	5	45	480
b5-50	6	50	5	45	600
b5-60	6	60	5	45	720
b6-48	6	48	6	45	480
b6-60	6	60	6	45	600
b6-72	6	72	6	45	720
b7-56	6	56	7	45	480
b7-70	6	70	7	45	600
b7-84	6	84	7	45	720
b8-64	6	64	8	45	480
b8-80	6	80	8	45	600
b8-96	6	96	8	45	720

Table 4.2: Characteristics of the benchmark test instances.

	C-DARP			F-DARP			Model I		Model II	
Inst.	Obj.	Gap	CPU	Obj.	Gap	CPU	Obj.	CPU	Obj.	CPU
a2-16	294.3		0.08	294.3		0.04	294.3	0.04	294.3	0.10
a2-20	344.9		0.23	344.9		0.12	344.9	0.02	344.9	0.02
a2-24	431.1		0.68	431.1		0.28	431.1	0.06	431.1	0.05
a3-18	300.5		0.68	300.5		0.24	300.5	0.05	300.5	0.04
a3-24	344.9		5.53	344.9		1.93	344.9	0.10	344.9	0.10
a3-30	494.8		20.14	494.8		297	494.8	0.07	494.8	0.04
a3-36	583.2		28.70	583.2		6.65	583.2	0.09	583.2	0.11
a4-16	282.7		5.61	282.7		2.85	282.7	0.05	282.7	0.06
a4-24	375.0		11.13	375.0		24.95	375.0	0.07	375.0	0.04
a4-32	485.5		591	485.5	8	2h	485.5	0.12	485.5	0.15
a4-40	557.7		1903	557.7	4	2h	557.7	0.48	557.7	0.28
a4-48	669.4	6	2h	680.0	10	2h	668.8	0.67	668.8	0.50
a5-40	498.4	2	2h	498.4	11	2h	498.4	0.23	498.4	0.23
a5-50	696.3	17	2h	N/A	N/A	2h	686.6	4.02	686.6	2.47
a5-60	828.3	14	2h	N/A	N/A	2h	808.4	1.41	808.4	1.03
a6-48	604.1	18	2h	N/A	N/A	2h	604.1	1.33	604.1	0.89
a6-60	874.8	16	2h	N/A	N/A	2h	819.3	6.17	819.3	8.83
a6-72	N/A	N/A	2h	N/A	N/A	2h	916.1	17.88	916.1	13.87
a7-56	764.5	19	2h	N/A	N/A	2h	724.0	1.92	724.0	1.24
a7-70	N/A	N/A	2h	N/A	N/A	2h	875.7	4.77	875.7	6.93
a7-84	N/A	N/A	2h	N/A	N/A	2h	1033.3	35.70	1033.3	5.89
a8-64	N/A	N/A	2h	N/A	N/A	2h	747.5	15.31	747.5	7.19
a8-80	N/A	N/A	2h	N/A	N/A	2h	945.8	51.19	945.8	25.11
a8-96	N/A	N/A	2h	N/A	N/A	2h	1229.7	3593	1229.7	461

Table 4.3: Solution values and computing times for the benchmark test instances 'a'.

	C-DARP			F-DARP			Model I		Model II	
Inst.	Obj.	Gap	CPU	Obj.	Gap	CPU	Obj.	CPU	Obj.	CPU
b2-16	309.4		0.13	309.4		0.13	309.4	0.03	309.4	0.03
b2-20	332.7		0.06	332.7		0.02	332.7	0.02	332.7	0.01
b2-24	444.7		0.91	444.7		0.81	444.7	0.06	444.7	0.05
b3-18	301.6		0.83	301.6		0.73	301.6	0.05	301.6	0.04
b3-24	394.5		5.28	394.5		0.74	394.5	0.11	394.5	0.11
b3-30	531.4		1.68	531.4		0.22	531.4	0.07	531.4	0.06
b3-36	603.8		6.41	603.8		2.17	603.8	0.07	603.8	0.09
b4-16	297.0		0.23	297.0		0.03	297.0	0.04	297.0	0.02
b4-24	371.4		1.86	371.4		0.66	371.4	0.06	371.4	0.06
b4-32	494.9		3.01	494.9		0.57	494.9	0.04	494.9	0.04
b4-40	656.6		51.15	656.6		20.04	656.6	0.13	656.6	0.14
b4-48	673.8	2	2h	680.7	6	2h	673.8	1.14	673.8	0.94
b5-40	613.7	10	2h	619.1	6	2h	613.7	0.20	613.7	0.21
b5-50	763.0	7	2h	768.2	8	2h	761.4	0.54	761.4	0.45
b5-60	917.6	14	2h	N/A	N/A	2h	902.0	4.01	902.0	0.92
b6-48	714.8		2700	715.9	2	2h	714.8	0.27	714.8	0.26
b6-60	893.4	11	2h	N/A	N/A	2h	860.0	0.50	860.0	0.43
b6-72	1083.9	22	2h	N/A	N/A	2h	978.5	3.48	978.5	5.11
b7-56	850.7	14	2h	867.9	14	2h	824.0	6.99	824.0	3.62
b7-70	919.7	10	2h	N/A	N/A	2h	912.6	2.54	912.6	2.42
b7-84	N/A	N/A	2h	N/A	N/A	2h	1203.4	2.93	1203.4	2.72
b8-64	N/A	N/A	2h	N/A	N/A	2h	839.9	2.03	839.9	2.16
b8-80	1433.0	37	2h	N/A	N/A	2h	1036.4	1.34	1036.4	1.54
b8-96	N/A	N/A	2h	N/A	N/A	2h	1185.6	27.42	1185.6	24.26

Table 4.4: Solution values and computing times for the benchmark test instances 'b'.

of the cost objective and the computational time in seconds (CPU) are reported. For C-DARP and F-DARP we also report the relative gap (Gap) as some of the instances are not solved to optimality within the time limit of 7200 seconds. By taking a closer look at Tables 4.3 and 4.4, it becomes evident that Model II outperforms Model I in a majority of the instances, especially in the larger a-instances (in terms of number of users and vehicles used). Moreover, one can see clearly from the results that Models I and II yield a more efficient formulation than C-DARP and F-DARP: Starting from instance a4-48 (C-DARP) and instance a4-32 (F-DARP), the a-instances could not be solved to optimality within two hours (or even no integer solution was found). The computational time needed to solve these instances with Model II range from less than one second to 461 seconds and from less than one second to about 1 hour for Model I. In general, with Models I and II the b-instances are easier to solve than the a-instances, as a large majority of the instances are solved in less than one second. This reflects the fact that the size of the MILPs decreases tremendously when users request more than one seat, which can be traced back to the fact that the size of the event-based graph decreases in this case. With C-DARP and F-DARP, CPLEX was not able to solve instances b4-48 to b8-96 to optimality within the time limit of two hours, sometimes no integer solution was found. This clearly emphasizes the computational efficiency of the event-based graph as the underlying structure of an MILP formulation of the DARP.

4.4.2 Artificial Data – City of Wuppertal

Ridepooling services are usually operated by taxis or mini-buses, whose passenger capacity is often equal to three or six. Thus, in the artificially created instances we consider the case that $Q \in \{3, 6\}$. In the case that $Q = 3$, we assume that each user requests one seat each, while for $Q = 6$ the number of requested seats per user is chosen randomly from $\{1, \ldots, 6\}$. Moreover, the service time s_j associated with location j is set to be equal to the number of requested seats q_j. This is in accordance with the benchmark test instances for the DARP created in Cordeau (2006). The instance size is determined by the number of users. For both $Q = 3$ and $Q = 6$ and for each number of users $n \in \{20, 30, 40, 60, 80, 100\}$ we generate a set of 5 instances with n users each. We denote the instances by $Q3.n.m$ and $Q6.n.m$, indicating the m-th instance with n users, $m \in \{1, \ldots, 5\}$. Pickup and delivery locations are chosen randomly from a list of streets in Wuppertal, Germany. The taxi depot is chosen to be located next to the main train station in Wuppertal. The cost c_a for an arc a in the event-based graph is computed as the length of a shortest path from its tail to its head in an OpenStreetMap network corresponding to the city of Wuppertal using the Python API `OSMnx`[2]. The shortest path is calculated based on OpenStreetMap data using the shortest path method of the Python package `NetworkX`[3]. Due to slowly moving traffic within the city center, the average travel speed is set to 15 km/h, so that the travel time in minutes is equal to $t_a = 4\, c_a$. The maximum duration of service T is set to 240 minutes. Earliest pickup times are chosen randomly from five-minute intervals within the next 5–205 minutes (we consider inbound requests only) and the time window length is chosen to be equal to 15 minutes, as we assume that users of ridepooling services in a city,

[2] `https://github.com/gboeing/osmnx`
[3] `https://networkx.org/`

Instances	Q	n	K	Instances	Q	n	K
Q3.20.1-5	3	20	4-5	Q6.20.1-5	6	20	5-6
Q3.30.1-5	3	30	5-6	Q6.30.1-5	6	30	6-7
Q3.40.1-5	3	40	6-7	Q6.40.1-5	6	40	8-9
Q3.60.1-5	3	60	9-10	Q6.60.1-5	6	60	11-12
Q3.80.1-5	3	80	10-11	Q6.80.1-5	6	80	13-14
Q3.100.1-5	3	100	13-14	Q6.100.1-5	6	100	16-17

Table 4.5: Characteristics of the Wuppertal artificial test instances.

where public transport operates at high frequencies, want to be picked up without long waiting times. A user i's maximum ride time L_i is set to 1.5 times the ride time for a direct ride from the pickup to the delivery location. A summary of the artificial instances' remaining characteristics can be found in Table 4.5.

In general, the artificial instances are more complex than the benchmark instances: there is a smaller planning horizon (240–720 minutes for the benchmark instances and 240 minutes for the artificial instances) during which the same number of user requests have to be served, i.e., the ratio of user requests per time is higher. This is also reflected by the number of required vehicles. While there are only two to eight vehicles in the benchmark test instances to serve between 16 and 96 user requests, there are four to 17 vehicles required in the artificial instances.

It is shown in the previous subsection that Model II performs better than Model I. Therefore, we restrict the following tests to Model II. We compare the impact of employing different objective functions from Section 4.3 on the economic efficiency and the customer satisfaction of the final routing solution. The respective objective functions are used in Model II in the place of (4.9a). Moreover, for the objective function f_{rcr} we add variables p_i, $i \in R$ to Model II and replace constraints (4.1c) by constraints (4.3). In case of the objective functions f_r, f_{cr} and f_{rcr}, we add variables $d_i \geq 0$, $i \in R$ and constraints (4.11) to Model II. In case of the objective functions involving the users' *maximum regret*, i.e., $f_{r_{\max}}$ and $f_{cr_{\max}}$, we additionally add the variable $d_{\max} \geq 0$ and constraints (4.12) to Model II.

The weights in the objective functions involving more than one criterion are chosen from a user perspective and based on preliminary numerical tests. In the first part of the computational experiments we consider the single-objective functions f_c, f_r and $f_{r_{\max}}$. The weights in f_{cr} and $f_{cr_{\max}}$ are then chosen so that the values of total and maximum regret, respectively, in f_{cr} and $f_{cr_{\max}}$ deviate not more than 15% from the optimal objective values for f_r and $f_{r_{\max}}$ (in instances solved to optimality). After some preliminary testing the weights are set to $\alpha = 1$ and $\beta = \frac{n}{5}$, which yields

$$f_{cr} = \sum_{a \in A} c_a \, x_a + \sum_{i \in R} d_i \quad \text{and}$$
$$f_{cr_{\max}} = \sum_{a \in A} c_a \, x_a + \frac{n}{5} \, d_{\max}.$$

Note that, choosing the weighting parameter β as a function of n ensures that not only

	f_c			f_r			
n	f_c	f_r	CPU	f_c	f_r	Gap	CPU
Q3							
20	113.3	136.8	0.06	142.5	12.2	0	0.05
30	170.7	250.3	0.06	209.1	23.5	0	0.07
40	205.8	346.6	0.15	259.3	46.4	0	0.55
60	309.6	631.0	1.09	385.7	128.8	0	44.33
80	383.5	798.0	7.00	480.2[4]	193.6[4]	13[4]	4721[4]
100	445.5	1070.4	218	603.7[3]	193.0[3]	31[3]	6060[3]
Q6							
20	117.6	144.2	0.02	135.7	14.3	0	0.02
30	193.5	202.3	0.04	223.7	39.8	0	0.05
40	246.8	273.0	0.06	294.8	48.16	0	0.06
60	343.9	467.3	0.26	393.3	99.7	0	2.13
80	448.9	640.4	0.73	520.3	148.8	0	106
100	543.4	791.6	6.07	627.1	200.7	5	3236

[3] The average is built only over three instances, as two instances are not solved within 7200s.
[4] The average is built only over four instances, as one instance is not solved within 7200s.

Table 4.6: Average values on the Q3 and Q6 test instances solved with the objective functions f_c and f_r.

the first term in $f_{cr_{\max}}$ grows with the number of users.

By using the objective function f_{rcr} we can measure the impact of denying certain requests on the served users' regret and the total routing costs. The weighting parameter γ in f_{rcr} defines the trade-off between the general requirement of answering as many requests as possible on one hand, and the goal of cost and time efficient transport solutions on the other hand. After testing several weights, it turns out that $\gamma = 20$ is a reasonable choice, yielding

$$f_{rcr} = \sum_{a \in A} c_a \, x_a + \sum_{i \in R} d_i + 20 \left(n - \sum_{i \in R} p_i \right).$$

In our numerical experiments, on average at most 10% of the requests are rejected when using these weights. Note that, several authors using weighted-sum objectives base their choice of weights on Jorgensen et al. (2007) (see, e.g., Kirchler and Wolfler Calvo, 2013; Mauri et al., 2009). However, the total routing costs, the total/maximum regret and the number of unanswered requests depend strongly on the test instances and may vary considerably. Since in this section we create a new class of test instances, we refrain from using these predetermined weights.

In Tables 4.6, 4.7 and 4.8 the results are summarized. All figures reported are average values over five instances $Q3.n.m$, $m \in \{1, \ldots, 5\}$ and $Q6.n.m$, $m \in \{1, \ldots, 5\}$, respectively. The tables contain information about the total routing costs, the total regret (and where applicable the maximum regret and the percentage of answered requests) for each number of user requests $n \in \{20, 30, 40, 60, 80, 100\}$ of the instance sets $Q3$ and $Q6$. Fur-

	f_{cr}					f_{rcr}					
n	Obj.	f_c	f_r	Gap	CPU	Obj.	f_c	f_r	AR	Gap	CPU
Q3											
20	141.5	128.3	13.2	0	0.04	140.4	121.2	7.2	97	0	0.04
30	219.0	192.5	26.4	0	0.08	215.9	186.5	17.3	98	0	0.08
40	284.0	234.3	49.7	0	0.47	275.3	220.1	19.2	96	0	0.47
60	495.6	361.6	134.0	0	37.04	459.3	326.7	56.6	93	0	6.77
80	663.3[4]	454.3[4]	208.9[4]	3[4]	3680[4]	598.2	412.8	61.4	92	1.2	1705
100	762.7	533.8	228.9	12	6416	718.6	500.3	86.4	93	8.2	5796
Q6											
20	139.8	124.5	15.4	0	0.03	138.1	122.5	11.6	99	0	0.03
30	253.6	209.2	44.4	0	0.07	241.5	189.8	11.7	93	0	0.05
40	316.3	267.4	48.9	0	0.08	304.1	249.0	19.1	96	0	0.07
60	476.9	373.7	103.2	0	1.52	438.1	340.7	33.4	95	0	0.64
80	655.4	503.3	152.1	0	20.24	589.5	450.1	27.4	93	0	1.75
100	804.9	595.0	209.9	0	853	719.5	530.7	44.8	93	0	5.28

[4] The average is built only over four instances, as one instance is not solved within 7200s.

Table 4.7: Average values on the Q3 and Q6 test instances solved with the objective function f_{cr} and f_{rcr}.

	$f_{r_{\max}}$					$f_{cr_{\max}}$					
n	f_c	f_r	$f_{r_{\max}}$	Gap	CPU	Obj.	f_c	f_r	$f_{r_{\max}}$	Gap	CPU
Q3											
20	142.0	48.6	8.9	0	0.03	159.0	119.9	91.9	9.8	0	0.04
30	211.7	64.6	7.9	0	0.05	239.1	191.8	114.3	7.9	0	0.14
40	255.9	132.5	9.5	0	0.16	305.8	228.4	218.4	9.7	0	0.65
60	388.8	301.4	13.0	0	0.84	484.6	325.6	442.9	13.3	0	3.11
80	495.0	427.5	12.6	0	1443	634.2	431.3	557.4	12.7	1	1749
100	599.9	499.2	11.0	1	1496	738.0	506.5	709.7	11.6	2	5884
Q6											
20	137.4	42.2	5.9	0	0.02	149.8	125.9	50.6	6.0	0	0.04
30	223.8	118.6	11.2	0	0.11	274.3	206.8	149.9	11.2	0	0.05
40	293.5	114.0	12.3	0	0.06	355.9	255.0	232.5	12.6	0	0.11
60	406.8	246.7	11.5	0	0.28	502.4	363.8	332.1	11.6	0	0.84
80	531.2	365.9	11.0	0	1.52	663.0	485.7	431.9	11.1	0	7.15
100	646.3	478.8	12.0	0	15.93	815.3	573.2	588.3	12.1	0	71.12

Table 4.8: Average values on the Q3 and Q6 test instances solved with the objective functions $f_{r_{\max}}$ and $f_{cr_{\max}}$.

	f_r vs. f_c		f_{cr} vs. f_c		f_{cr} vs. f_r		$f_{r_{\max}}$ vs. f_c		$f_{cr_{\max}}$ vs. f_c		$f_{cr_{\max}}$ vs. $f_{r_{\max}}$	
n	Δf_c	Δf_r	Δf_c	ΔCPU	Δf_r	ΔCPU	Δf_c	Δf_r	Δf_c	Δ CPU	$\Delta f_{r_{\max}}$	Δ CPU
Q3												
20	26	-91	13	-36	8	-22	25	-64	6	-21	10	47
30	22	-91	13	32	12	24	24	-74	12	119	0	152
40	26	-87	14	216	7	-15	24	-62	11	336	2	309
60	25	-80	17	3286	4	-16	26	-52	5	184	2	269
80	25	-76	18	52539	8	-22	29	-46	12	24927	1	21
100	35	-82	20	2840	19	6	35	-53	14	2597	6	293
Avg.	27	-84	16	9813	10	-7	27	-59	10	4690	4	182
Q6												
20	15	-90	6	75	7	75	17	-71	7	150	1	82
30	16	-80	8	57	12	43	16	-41	7	29	0	-51
40	19	-82	8	36	2	23	19	-58	3	96	3	77
60	14	-79	9	489	4	-29	18	-47	6	224	1	203
80	16	-77	12	2664	2	-81	18	-43	8	877	0	372
100	15	-75	9	13948	5	-74	19	-40	5	1071	1	346
Avg.	16	-80	9	2878	5	-7	18	-50	6	408	1	172

Table 4.9: Comparison of the change in total routing costs, total regret and CPU time (all in percent) on the Q3 and Q6 test instances solved with the objective functions f_c, f_r, f_{cr}, $f_{r_{\max}}$ and $f_{cr_{\max}}$.

thermore, the relative gap and the CPU time returned by CPLEX are reported. If no relative gap is shown, all instances are solved to optimality.

The computational times show that Model II combined with a cost-objective function can be used to solve small to medium-sized artificial instances very efficiently. This is in accordance with the results indicated by the benchmark instances. The computational time increases when other objective functions are used, but most of the instances can still be solved within a few seconds. However, for the larger $Q3$-instances ($n \in \{80, 100\}$) the CPU time increases drastically. In general, the $Q6$-instances are easier to solve than the $Q3$-instances, reflecting the smaller number of possible user allocations in the vehicle when the number of requested seats per user is not limited to one. While the scenario that users may request any number of seats between one and six reflects the conditions under which ridepooling services operate in reality, a uniform distribution of q_i in the set $\{1, \ldots, 6\}$ is probably not realistic. In Section 7.2, the distribution of group sizes is discussed based on *Hol mich! App* data.

A comparison of the effects of different objective functions in Model II can be found in Tables 4.9 and 4.10. In the second and third column of Table 4.9 we illustrate the change (in percent) in total routing costs and total regret when replacing f_c by f_r. While the regret decreases by an average of 84% and 80% (Q3- and Q6-instances, respectively), the routing costs only increase by 27% and 16% on average. This shows that by including the criterion of the users' regret in the objective function we can spare users a large amount of unnecessary ride or waiting time. This comes at the cost of higher routing costs. However, even from a service provider's perspective it might be reasonable to accept a rather small

	$f_{cr_{\max}}$ vs. f_{cr}			f_{rcr} vs. f_{cr}			
n	Δf_c	Δf_r	ΔCPU	Δf_c	Δf_r	ΔCPU	ΔAR
Q3							
20	-7	594	22	-6	-46	6	-3
30	0	333	66	-3	-34	0	-2
40	-3	340	38	-6	-61	1	-4
60	-10	231	-92	-10	-58	-82	-7
80	-5	167	-52	-9	-71	-54	-8
100	-5	210	-8	-6	-62	-10	-7
Avg	-5	312	-4	-7	-55	-23	-5
Q6							
20	1	230	43	-2	-24	21	-1
30	-1	237	-18	-9	-74	-21	-7
40	-5	375	45	-7	-61	-5	-4
60	-3	222	-45	-9	-68	-58	-5
80	-3	184	-65	-11	-82	-91	-7
100	-4	180	-92	-11	-79	-99	-7
Avg	-2	238	-22	-8	-65	-42	-5

Table 4.10: Comparison of the change in total routing costs, total regret, CPU time and number of answered requests (all in percent) on the Q3 and Q6 test instances solved with the objective functions f_{cr}, $f_{cr_{\max}}$ and f_{rcr}.

loss in order to improve user convenience and to remain competitive. In column six of the same table we demonstrate that the weights chosen in the cost-regret objective f_{cr} indeed reflect a user perspective: On average (over the instances solved to optimality within the time limit) there is an increase of at most 15% in total regret compared to solely optimizing w.r.t. f_r. Moreover, there is an average increase in routing costs of 16% (Q3-instances) and 9% (Q6-instances) compared to the costs when using routing costs as the only optimization criterion, i.e., when using f_c as the objective function. Comparing f_{cr} to f_c, we observe an increase in CPU time for all instances but $Q = 3$ and $n = 20$, but CPU times for $Q = 3$ and $n \in \{20, 30, 40, 60\}$ and all Q6-instances are still reasonable as shown in Table 4.7. Comparing f_{cr} to f_r we observe an average decrease in CPU time of 7% for the Q3- and Q6-instances. Similar results are obtained for the objective functions f_c, $f_{r_{\max}}$ and $f_{cr_{\max}}$, although the average decrease in regret when comparing $f_{r_{\max}}$ to f_c is only 59% (Q3-instances) and 50% (Q6-instances). The meaningful choice of the weighting parameter in $f_{cr_{\max}}$ is demonstrated by the second last column in Table 4.9, as the increase in maximum regret is below 15% for all instances.

Since both of the weighted-sum objectives f_{cr} and $f_{cr_{\max}}$ improve user convenience and increase routing costs compared to f_c, we evaluate which of the objective functions is more effective in this respect. Columns two to four in Table 4.10 illustrate that when using $f_{cr_{\max}}$ instead of f_{cr} the average computational time decreases between 8% to 92% (Q3-instances) and between 45% and 92% (Q6-instances) in the larger instances, i.e., $n \in \{60, 80, 100\}$, which are harder to solve in general. Despite its computational superiority and the fact that the objective function $f_{cr_{\max}}$ generally improves the user satisfaction of optimal tours, it has some shortcomings. Indeed, if there is one user with a high regret d_i, for instance,

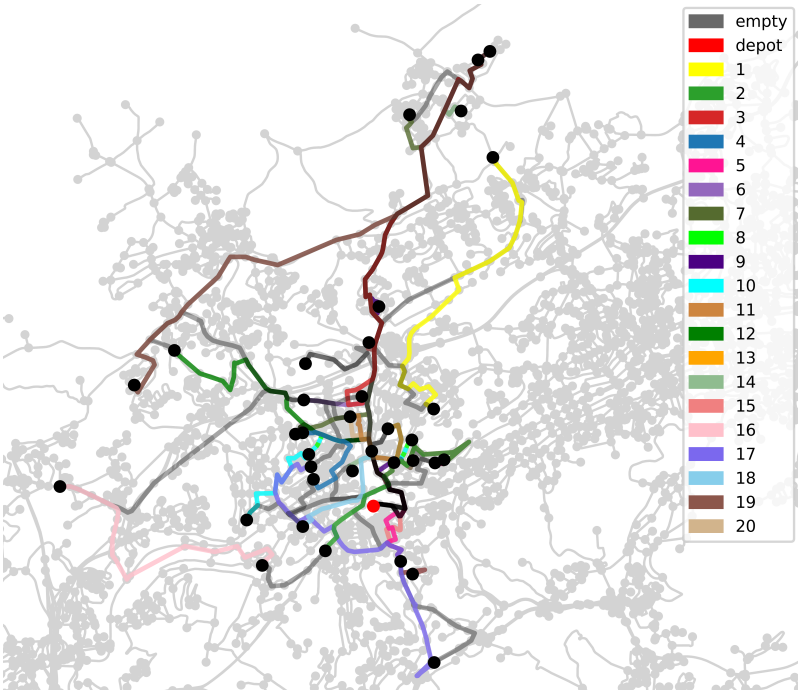


Figure 4.2: Vehicle routes of instance $Q3n20.5$ solved using the objective function f_{cr}.

because he or she is the last user in a tour to be dropped off, the time loss of all other users j with $d_j < d_i$ has no impact on the objective function. Therefore, the other users may not be driven to their delivery points as quickly as possible. This is reflected by the increase in regret using the objective $f_{cr_{\max}}$ as compared to f_{cr}, shown in Table 4.10: 312% (Q3-instances) and 238% (Q6-instances). The routing costs remain roughly the same; there is an average decrease of 5% (Q3-instances) and 2% (Q6-instances). Due to these shortcomings, we do not consider a tri-criterion weighted-sum objective function $f_{rcr_{\max}}$, but restrict our attention to f_{rcr} in the remaining discussion.

The last four columns of Table 4.10, f_{rcr} vs. f_{cr}, illustrate the decrease in the overall routing costs and regret that is obtained when rejecting some of the requests. This is illustrated at the instance $Q3n20.5$ in Figure 4.2, showing an optimal tour w.r.t. f_{cr}, and Figure 4.3 where an optimal tour w.r.t. f_{rcr} is shown. The depot in both figures is marked in red, and a colored part of the route represents the part where a certain request is in the vehicle. Dark gray parts mark parts of the route where the vehicle is empty. Note that parts of the routes (and colors) overlap, as some vehicles might use the same roads on parts of their routes. In Figure 4.2 the driver has to make a detour to transport requests 1 (yellow part of the route) and 19 (dark brown part of the route), while it becomes obvious from Figure 4.3 that the service provider benefits from a large decrease in routing costs. Table 4.11 contains the corresponding vehicle routes including pickup and delivery times in minutes after the start of service. In the vehicle tours computed using f_{rcr}, users 8 and 10 who are transported with an associated regret of 5 and 3 minutes, respectively, are transported without any time loss.

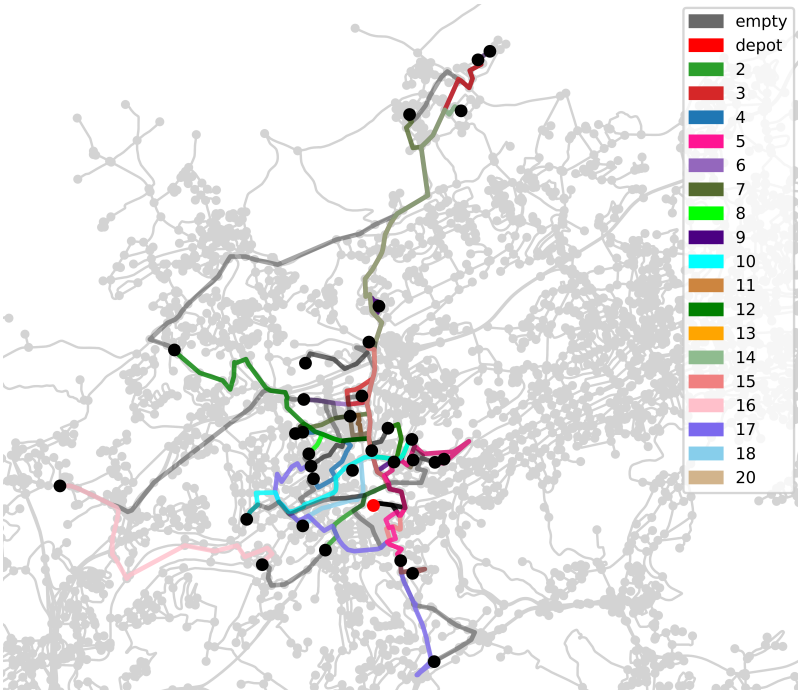


Figure 4.3: Vehicle routes of instance $Q3n20.5$ solved using the objective function f_{rcr}.

		f_{cr}											
Tour 1	Location	1^+	1^-	15^+	15^-	5^+	5^-	8^+	10^+	8^-	10^-		
	Time[m]	24.4	43.2	60.0	76.9	120.0	136.6	156.5	160.0	168.0	175.6		
Tour 2	Location	9^+	9^-	6^+	6^-	7^+	7^-						
	Time[m]	60.0	71.7	95.0	120.6	150.0	173.2						
Tour 3	Location	12^+	12^-	14^+	14^-	17^+	17^-						
	Time[m]	60.0	79.4	105.0	128.0	180.0	200.6						
Tour 4	Location	13^+	13^-	3^+	3^-	19^+	19^-						
	Time[m]	30.0	31.0	85.0	108.9	140.0	170.7						
Tour 5	Location	20^+	20^-	4^+	4^-	18^+	18^-	11^+	11^-	16^+	16^-	2^+	2^-
	Time[m]	50.0	56.2	65.0	72.0	85.0	92.5	95.0	102.4	121.3	138.1	155.0	169.1

Table 4.11: Vehicle routes (without depot) of instance $Q3n20.5$ solved using the objective function f_{cr}.

		f_{rcr}											
Tour 1	Location	15^+	15^-	5^+	5^-	10^+	10^-						
	Time[m]	60.0	76.9	120.0	135.6	160.0	172.5						
Tour 2	Location	13^+	13^-	9^+	9^-	6^+	6^-	7^+	7^-				
	Time[m]	30.0	31.0	60.0	71.7	95.0	120.6	150.0	173.2				
Tour 3	Location	12^+	12^-	14^+	14^-	17^+	17^-						
	Time[m]	60.0	79.4	105.0	128.0	180.0	200.6						
Tour 4	Location	3^+	3^-	8^+	8^-								
	Time[m]	85.0	108.9	155.0	163.0								
Tour 5	Location	20^+	20^-	4^+	4^-	18^+	18^-	11^+	11^-	16^+	16^-	2^+	2^-
	Time[m]	50.0	56.2	65.0	72.0	85.0	92.5	95.0	102.4	121.3	138.1	155.0	169.1

Table 4.12: Vehicle routes (without depot) of instance $Q3n20.5$ solved using the objective function f_{rcr}.

For the Q3- and Q6-instances, on average 5% less of the requests are answered in comparison to the results obtained with the objective function f_{cr}. In turn, we observe an average decrease of the total routing costs and the total regret for all instance sizes. While the decrease in routing costs ranges from 2% to 11%, there are huge savings in regret: There is an average decrease of 55% and 65% (Q3- and Q6-instances, respectively.) Furthermore, computational time decreases by 23% (Q3-instances) and 42% (Q6-instances).

While it comes at cost of an increase in computational time and routing costs, the tests show that it is worthwhile to use more than one optimization criterion. Objective functions f_{cr} and f_{rcr} provide satisfactory results w.r.t. an increase in the users' total regret as compared to the total regret when using it as the only optimization criterion. Moreover, we show that the service quality can be improved for some users and costs can be reduced by rejecting unfavorable requests. Average computation times using objective function f_{rcr} for $Q = 3$, $n \in \{20, 30, 40, 60\}$ and $Q = 6$ are less than 7 seconds. Hence, our model can be used to solve small to medium-sized instances. It is applicable without the need for extensive coding, as it only consists of an MILP solver and the generation of the event-based graph.

4.5 Summary

In this chapter we suggest a new perspective on modeling ridepooling problems. By using an event-based graph representation rather than a location-based model, we show that capacity, pairing and precedence constraints can be handled implicitly. While the resulting MILP formulations are compact and generally have more variables as compared to classical compact models, extensive numerical experiments show that the implicit constraint formulation leads to considerably improved computational times. Indeed, both for benchmark instances from the literature as well as for artificial instances in the city of Wuppertal, problems with up to 80 requests can be solved in less than 7 seconds.

Moreover, we analyse the effects of including additional optimization criteria in the

model. In addition to the classical cost objective function, we consider the (total or maximum) regret as well as the number of rejected requests as measures for user satisfaction. By combining these criteria into a weighted-sum objective function we demonstrate that user satisfaction can be largely improved at only relatively small additional expenses, i.e., overall routing costs.

Model II proposed in this chapter constitutes a basis for the following chapters. Since Model II outperforms Model I, in the following, if we mention the event-based formulation, we refer to Model II and call it the event-based (EB) MILP.

While most real world instances of the DARP are much larger, the proposed approach can be used as a subroutine also for the dynamic DARP. Indeed, often only very few new requests arrive simultaneously and re-routing of already scheduled users is only acceptable if it does not postpone their arrival time too much. Consequently, the number of simultaneous users in such a rolling-horizon version of a dial-a-ride problem is relatively small and can be solved exactly using the EB model as shown in Chapter 6.

Since the number of variables and constraints in the event-based models strongly depends on the size of the graph, in the following chapter our efforts aim at developing techniques to detect infeasible nodes and arcs. Moreover, by combining the EB model with a location-based formulation, a new formulation is proposed, which is proven to be tighter than the location-based standard model proposed in Ropke et al. (2007), and integral under the assumption that every pair of locations induces a unique ordering (and other details).

5 Location-Augmented-Event-Based MILP Models: A Tight Formulation for the Static Dial-a-Ride Problem

In this chapter, we propose novel MILP formulations that combine the advantages of the location-based formulation from Ropke et al. (2007) and the EB formulation presented in the previous chapter (see Model II). We introduce a *location-augmented-event-based (LAEB)* formulation and an *aggregated location-augmented-event-based (ALAEB)* formulation. The ALAEB is a slightly adapted variant of the LAEB that has significantly fewer binary variables and thus potentially reduces the size of the branch-and-bound tree. We show that both formulations have a tighter LP relaxation than the location-based model proposed in Ropke et al. (2007). Moreover, the new formulations are tight in the sense that their LP relaxation has an integral solution if the time windows fulfill additional conditions. This tightness is also reflected by an average root node gap of only 1.6% in our computational study (for general time windows). To preprocess the models, lower and upper bounds on the beginning of service times at all nodes are computed to reduce the size of the event-based graph and hence of the MILP formulations (by 22% on average in our tests). In a second step the new models are further improved by adapting problem specific valid inequalities inspired by Schulz and Pfeiffer (2024). Numerical experiments on benchmark instances show that computational times are on average reduced by 53.9% compared to the state-of-the-art EB formulation.

This chapter deals with the deterministic, homogeneous and static DARP (see Section 3.3). We assume that all requests have to be accepted and that the objective is to minimize the total routing costs.

The chapter is structured as follows: First, we present the new location-augmented-event-based formulations in Section 5.1. Our theoretical investigation of the new formulations is presented in Section 5.2. In Section 5.3, we introduce methods to improve the performance of branch-and-bound search. The new formulations as well as the introduced methods are evaluated in the computational study in Section 5.4. Finally, the chapter closes with a conclusion in Section 5.5. The results in this chapter have been published in Gaul et al. (2023).

5.1 Location-Augmented-Event-Based MILP Models

The DARP can be represented by different graph-based formulations: location-based and event-based representations. In this chapter, the LB formulation from Ropke et al. (2007) (see Section 3.4) is used as a location-based reference model. Comparing the EB formulation with the LB formulation we find advantages for both of them (implicit formulation of precedence, pairing, and capacity constraints vs. fewer binary variables). We combine

both formulations into two location-augmented-event-based formulations in Sections 5.1.1 and 5.1.2, respectively, by using the advantages of both the event- and the location-based formulation.

5.1.1 Location-Augmented-Event-Based MILP Formulation

In this section, we integrate the implementation of time consistency in the LB formulation into the EB formulation to obtain a more efficient formulation.

$$Z_{LAEB} = \min \sum_{a \in A} c_a \cdot x_a \tag{5.1a}$$

$$\text{s.t. } (3.12\text{j}), (4.1\text{b}) - (4.1\text{d}), \text{ and } (4.9\text{h}),$$

$$e_j \leq \bar{B}_j \leq \ell_j \qquad \forall j \in J, \tag{5.1b}$$

$$\bar{B}_j \geq \bar{B}_i + s_i + \bar{t}_{ij} - \bar{M}_{ij} \left(1 - \sum_{v:v_1=i,\, w:w_1=j} x_{(v,w)} \right) \qquad \forall i, j \in J. \tag{5.1c}$$

Due to the construction of the event-based graph and constraints (4.1c) and (4.9h) there is exactly one node w with $w_1 = j$ and $\sum_{a \in \delta^{in}(w)} x_a = 1$ while $\sum_{a \in \delta^{in}(w')} x_a = 0$ for all other nodes w' with $w'_1 = j$. Therefore, only for the first one $\tilde{M}_{v,w}(1 - x_{(v,w)})$ is 0 in constraints (4.9b). This constraint sets $B_w = \bar{B}_j$ while the others only ensure that feasible values are used for $B_{w'}$. This leads to the simplification of time consistency constraints (5.1c). By this, maximum ride time constraints (3.12j) and time windows constraints (5.1b) can be ensured as in the LB formulation. As can be seen, the number of constraints in (3.12j), and (5.1b)–(5.1c) is clearly smaller than in (4.9b)–(4.9g). Moreover, the numerically unfavorable big-M-constraints are stronger in (5.1c) than in (4.9b), as

$$\left(1 - \sum_{v:v_1=v'_1,\, w:w_1=w'_1} x_{(v,w)} \right) \leq (1 - x_{(v',w')}) \qquad \forall (v', w') \in A$$

while $\bar{M}_{ij} = \tilde{M}_{v,w}$ with $v_1 = i$ and $w_1 = j$ with the classical choice

$$\bar{M}_{ij} = \ell_i + s_i + \bar{t}_{ij} - e_j \qquad \forall i, j \in J$$

and

$$\tilde{M}_{v,w} = \ell_{v_1} + s_{v_1} + t_{(v,w)} - e_{w_1} \qquad \forall (v, w) \in A.$$

Thus, the model formulation (3.12j), (4.1b)–(4.1d), (4.9h), and (5.1a)–(5.1c) is an improved version of (4.9a)–(4.9h). In the next section, we present another location-augmented-event-based formulation with fewer binary variables.

5.1.2 Aggregated Location-Augmented-Event-Based MILP Formulation

The LB formulation has the advantage that only $\mathcal{O}(n^2)$ binary variables are required while the EB formulation has $\mathcal{O}(n^{Q+1})$ binary variables (for $n \geq Q + 1$). We use the equality

$$\bar{x}_{ij} = \sum_{(v,w) \in A:v_1=i,\, w_1=j} x_{(v,w)}, \tag{5.2}$$

which is already used in (5.1c). Thereby, we get the following ALAEB formulation:

$$
\begin{aligned}
Z_{ALAEB} = \min \quad & \sum_{i,j \in J} \bar{c}_{ij} \cdot \bar{x}_{ij} && && \text{(5.3a)}\\
\text{s.t.} \quad & (3.12\text{j}), (4.1\text{b}) - (4.1\text{d}), (5.1\text{b}) \text{ and } (5.2), && &&\\
& \bar{B}_j \geq \bar{B}_i + s_i + \bar{t}_{ij} - \bar{M}_{ij}\,(1 - \bar{x}_{ij}) && \forall i, j \in J, && \text{(5.3b)}\\
& \bar{x}_{ij} \in \{0, 1\} && \forall i, j \in J, && \text{(5.3c)}\\
& 0 \leq x_a \leq 1 && \forall a \in A. && \text{(5.3d)}
\end{aligned}
$$

While (5.3a) corresponds to (3.12a), constraints (5.3b) are equivalent to (5.1c) using (5.2). Thus, the difference between both location-augmented-event-based models is the fact that $\bar{x}$ variables are added as binary variables while x variables are relaxed in constraints (5.3d). For this reason, the model contains fewer binary variables than the LAEB formulation. Note, that given an integer solution $\bar{x}$ the tours are completely described. Thus, there are at most K flows of size one from depot to depot through the event-based graph, i.e., x_a variables are also integer. In total, the ALAEB formulation replaces an exponential number of constraints (3.12e) by an additional number of $\mathcal{O}(n^{Q+1})$ continuous variables.

5.2 Theoretical Analysis

In this section, we investigate the properties of the LP relaxations of the LAEB and the ALAEB formulation in comparison to the LP relaxation of the LB formulation. We introduced the ALAEB formulation mainly to branch on a different set of variables. The following theorem shows, however, that the LP relaxations of the LAEB and the ALAEB formulation are equivalent.

Theorem 5.1. *The LP relaxations of the LAEB formulation* (3.12j)*,* (4.1b)–(4.1d)*,* (4.9h)*, and* (5.1a)–(5.1c) *and of the ALAEB formulation* (3.12j)*,* (4.1b)–(4.1d)*,* (5.1b)*,* (5.2)*, and* (5.3a)–(5.3d) *are equivalent.*

Proof. Constraints (3.12j), (4.1b)–(4.1d) and (5.1b) are part of both model formulations. As we consider the LP relaxation, constraints (4.9h) and (5.3d) are identical. Thus, we have to show that (5.1a) and (5.1c) are equivalent to (5.2) and (5.3a)–(5.3c).

Starting with (5.2), and (5.3a)–(5.3c), we can replace $\bar{x}_{ij}$ in objective (5.3a) and constraints (5.3b) by $\sum_{(v,w) \in A : v_1 = i,\, w_1 = j} x_{(v,w)}$ (equations (5.2)) and obtain objective (5.1a) and constraints (5.1c), respectively. Moreover, equations (5.2) and constraints (4.1b) and (4.1c) imply $0 \leq \bar{x}_{ij} \leq 1$ for all $i, j \in J$. Thus, constraints (5.3c) are redundant in the LP relaxation of ALAEB. However, then $\bar{x}_{ij}$ is only set in (5.2) but not used in the model any more such that (5.2) is also redundant and we obtain (5.1a) and (5.1c).

Analogously, we can start with (5.1a) and (5.1c), add the redundant equations (5.2) and constraints (5.3c). Using equations (5.2), we can replace $\sum_{(v,w) \in A : v_1 = i,\, w_1 = j} x_{(v,w)}$ by $\bar{x}_{ij}$ in objective (5.1a) and constraints (5.1c) to obtain objective (5.3a) and constraints (5.3b), respectively. By this, we obtain (5.2), and (5.3a)–(5.3c). □

We formulate and prove the following theorems only for the LAEB formulation, but due to Theorem 5.1 the results hold also for the ALAEB formulation. The next theorem

considers the special case that the time windows induce a unique order in time for every pair of locations. We note that in Bertsimas et al. (2019) a similar result for the large-scale dynamic optimization of taxis is used.

Theorem 5.2. *If the time windows $[e_j, \ell_j]$, $j \in J$, fulfill the following conditions*

1. *$\ell_{i^-} - e_{i^+} - s_i \leq L_i$ for all $i \in P$ and*
2. *either $\ell_i + s_i + \bar{t}_{ij} \leq e_j$ or $e_i + s_i + \bar{t}_{ij} > \ell_j$ holds for all $i, j \in J$,*

the LAEB formulation (3.12j), (4.1b)–(4.1d), (4.9h), *and* (5.1a)–(5.1c) *is integral if all arcs (v, w) with $v_1 = i$ and $w_1 = j$ are deleted if $\ell_j < e_i + s_i + \bar{t}_{ij}$.*

Proof. Condition 1 ensures that maximum ride time constraints (3.12j) are fulfilled. Because of the first case in condition 2

$$\bar{B}_i + s_i + \bar{t}_{ij} \leq \ell_i + s_i + \bar{t}_{ij} \leq e_j \leq \bar{B}_j,$$

i.e., $\bar{B}_i + s_i + t_{ij} \leq \bar{B}_j$ holds for these pairs i, j. In the second case, $e_i + s_i + \bar{t}_{ij} > \ell_j$, location j is not reached within the time window even if the vehicle starts as early as possible in location i. Thus, $x_{(v,w)} = 0$ for all (v, w) with $v_1 = i$ and $w_1 = j$. In this case, we can simply omit arc (v, w) from the graph. Due to condition 2 all remaining arcs fulfill $\ell_i + s_i + \bar{t}_{ij} \leq e_j$, i.e., location j is reached latest at the beginning of the time window if the vehicle visits location i directly before. Thus, $\bar{B}_i + s_i + t_{ij} \leq \bar{B}_j$ with $v_1 = i$ and $w_1 = j$ is always fulfilled. As a consequence constraints (5.1c) are always fulfilled. For the same reason we always find a solution with $\bar{B}_i \in [e_i, \ell_i]$ for all $i \in J$. Hence, we can always find a solution fulfilling the time window constraints (5.1b). Together, we can omit constraints (3.12j), and (5.1b)–(5.1c).

Constraints (4.1d) are equivalent to $\sum_{a \in \delta^{out}(0,\dots,0)} x_a = K$ by $c_{(0,\dots,0),(0,\dots,0)} = 0$, i.e., vehicles can drive from the depot to the depot without any costs. Without constraints (4.1c), the remaining formulation (4.1b), (4.1d), and (4.9h) with objective (5.1a) is a minimum cost flow problem with arc capacities $x_a \leq 1$. The left side of Figure 5.1

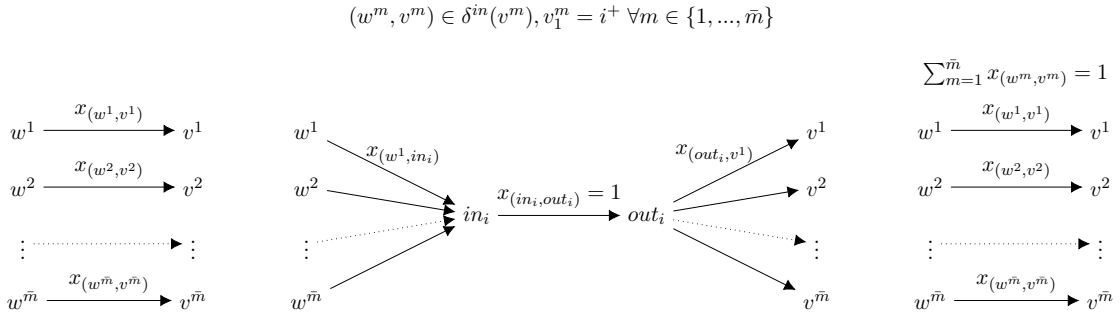


Figure 5.1: Network transformation.

corresponds to (4.1b), (4.1d), and (4.9h) with objective (5.1a) for a fixed location i^+, where v^m, $m \in \{1, \dots, \bar{m}\}$, are all events with $v_1^m = i^+$ and w^m the possible predecessor events. Note that, $v^m = v^{m'}$ for $m' \neq m$ might hold such that all relations between

predecessors w and events v with $v_1 = i^+$ are included. We can add two further nodes in_i and out_i in between and define out_i as a source with an outflow of 1 and in_i as a sink with inflow 1 without destroying the network flow property. In fact, this is equivalent to set the flow variable $x_{(in_i,out_i)} = 1$ (middle part of the figure).

As $x_{(w^m,v^m)} = x_{(w^m,in_i)} = x_{(out_i,v^m)}$, there is a flow of $x_{(w^m,v^m)}$ from w^m via in_i and out_i to v^m. First, this means that $\sum_{m=1}^{\bar{m}} x_{(w^m,v^m)} = x_{(in_i,out_i)} = 1$. Second, we can shrink $x_{(w^m,in_i)}$, $x_{(out_i,v^m)}$, and the corresponding flow between in_i and out_i to variable $x_{(w^m,v^m)}$ and obtain the situation on the right side of Figure 5.1. By definition $\sum_{m=1}^{\bar{m}} x_{(w^m,v^m)}$ sums up all incoming flow to all events with $v_1 = i^+$. Thus, it is equivalent to (4.1c) for the considered $i \in R$. By repeating this step for all $i \in R$, formulation (3.12j), (4.1b)–(4.1d), (4.9h), and (5.1a)–(5.1c) is integral if conditions 1 and 2 are fulfilled and all arcs (v, w) with $v_1 = i$ and $w_1 = j$ are deleted if $\ell_j < e_i + s_i + \bar{t}_{ij}$. □

Note that, condition 2 holds in particular if each pickup and delivery has a fixed time, that is, the time windows are limited to one unique point in time, i.e., $e_j = \bar{B}_j = \ell_j$ for all $j \in P \cup D$ (and time windows at depots are set appropriately). The theorem allows for several conclusions:

- The sequencing of requests makes the problem challenging. In fact, it is well-known that exact solution approaches for the DARP typically perform better for instances with tighter time windows which have a lower number of feasible sequences of requests.
- Condition 2 ensures that $\bar{B}_i + s_i + \bar{t}_{ij} \leq \bar{B}_j$ holds for all arcs (v, w) with $v_1 = i$ and $w_1 = j$ in the reduced graph. Besides the fact that constraints (5.1c) become unnecessary, this leads to a cycle-free graph structure. That is, there are no events any more which can be predecessors as well as successors of each other in different solutions.
- Example 5.3 shows that Theorem 5.2 cannot be transferred to the LB formulation (3.12a)–(3.12m). This fact suggests that the LAEB and the ALAEB formulation are tighter than the LB formulation.
- Moreover, to the best of our knowledge, EB, LAEB, and ALAEB are the first compact formulations (given the vehicle capacity) for the DARP for which Theorem 5.2 holds. This result is important for practice, as there are applications with fewer cycles in the graph, e.g., if customers' trips start or end at the same location (for example at a public transport stop), as well as solution approaches for a dynamic taxi routing problem which use similar results (Bertsimas et al., 2019).

Example 5.3. *Consider the instance with three customers, $q_1 = q_2 = 2$, $q_3 = 3$, $Q = 6$, and the travel times given in the left part of Figure 5.2. We assume $\bar{c}_{ij} = \bar{t}_{ij}$ for all $i, j \in J$ and that maximum ride time constraints are fulfilled. Moreover, let $e_{1^+} = \ell_{1^+} = 10$, $e_{2^+} = \ell_{2^+} = 11$, $e_{3^+} = \ell_{3^+} = 12$, $e_{1^-} = \ell_{1^-} = 13$, $e_{2^-} = \ell_{2^-} = 14$, $e_{3^-} = \ell_{3^-} = 15$. Figure 5.2 presents an optimal solution for* (3.12a)–(3.12j) *and $0 \leq \bar{x}_{ij} \leq 1$ for all $i, j \in J$ in the middle of the figure and an optimal solution for* (3.12a)–(3.12m) *on the right side of the figure. As can be seen there is a positive flow of 0.917 from 1^+ via 2^+ to 3^+, which*

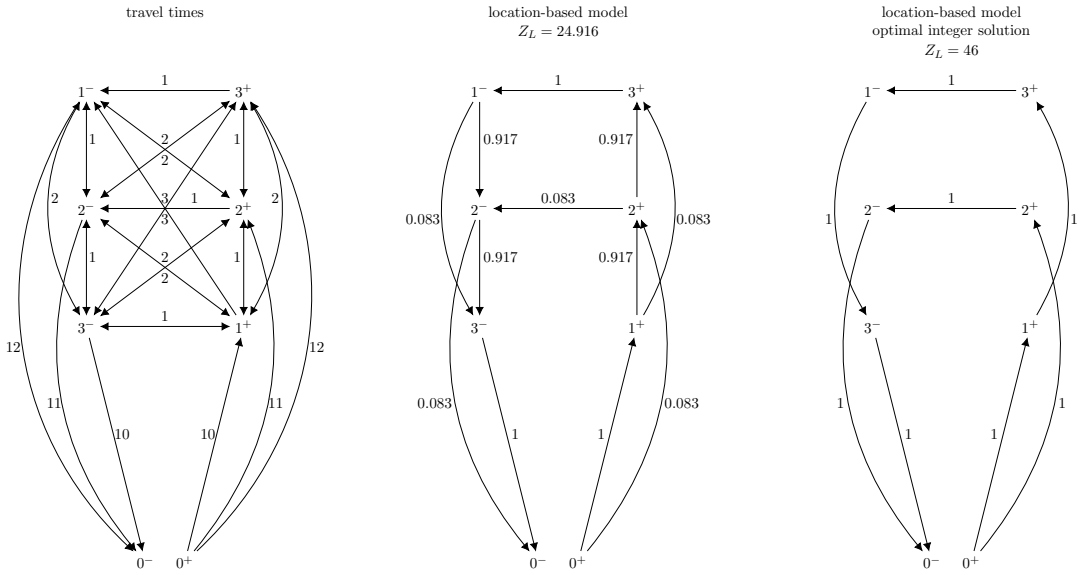


Figure 5.2: Example for LB formulation.

means that all three customers are in the vehicle at the same time at least for a fractional flow. As they require seven seats in total, but the vehicle has only six, all three customers cannot share the vehicle at the same time.

The following theorem gives formal evidence for the last point.

Theorem 5.4. *Let z^{rel}_{LB} be the objective value of an optimal solution of the LP relaxation of the LB model* (3.12a)–(3.12m)*, i.e., with $0 \leq \bar{x}_{ij} \leq 1$ for all $i, j \in \bar{J}$ instead of* (3.12m)*. Let further z^{rel}_{LAEB} be the objective value of an optimal solution of the LP relaxation of the LAEB model* (3.12j), (4.1b)–(4.1d), (4.9h)*, and* (5.1a)–(5.1c)*, i.e., with $0 \leq x_a \leq 1$ for all $a \in A$ instead of* (4.9h)*. Then, $z^{rel}_{LAEB} \geq z^{rel}_{LB}$ holds. Moreover, there are instances in which $z^{rel}_{LAEB} > z^{rel}_{LB}$ holds.*

Proof. The proof consists of three steps. First, we prove that every feasible LP solution of the LAEB formulation is also a feasible LP solution of the LB formulation, i.e., $z^{rel}_{LAEB} \geq z^{rel}_{LB}$. Second, we give an example that there is a feasible LP solution for the LB formulation which is not LP feasible for the LAEB formulation. Third, we give a concrete instance in which $z^{rel}_{LAEB} > z^{rel}_{LB}$ holds.

1. Set $\bar{x}_{ij} = \sum_{v:v_1=i,\, w:w_1=j} x_{(v,w)}$ for all $i, j \in J$, where $x_{(v,w)}$, $(v, w) \in A$ are given by a feasible LP solution of the LAEB formulation. Furthermore, define $\bar{x}_{0^+j} = \bar{x}_{0^-j} = \bar{x}_{0j}$ and $\bar{x}_{i0^+} = \bar{x}_{i0^-} = \bar{x}_{i0}$. Due to (4.1c) the ingoing flow is 1 for each pickup location i^+. Because of (4.1b) this holds also for the outgoing flow. All outgoing arcs of a node v with $v_1 = i^+$ end in a node w with $w_j = i$ for one $j \in \{2, \ldots, Q\}$ until a node w with $w_1 = i^-$ is reached. This holds also for all nodes connected by an arc with node w. With the same argument this holds for their connected nodes and so on. On the other hand there is no arc between a node w' with $w'_1 \neq i^+$ and $w'_l \neq i$ for all $l \in \{2, \ldots, Q\}$ and a node w with

$w_1 = i^-$. Thus, constraints (4.1b) lead to

$$\sum_{(v,w):w_1=i^+} x_{(v,w)} = 1 = \sum_{(v,w):w_1=i^-} x_{(v,w)}. \tag{5.4}$$

Together, (3.12b) and (3.12c) are fulfilled. With our choice of $\bar{x}_{ij}$, (4.1d) leads to

$$\sum_{j\in P} \bar{x}_{0^+j} = \sum_{(v,w):v_1=0} x_{(v,w)} \leq K,$$

which means that (3.12d) is fulfilled.

There is one constraint in (3.12e) for each $S \in \mathcal{S}$. Each S consists of a path starting in the starting depot 0^+ and visiting the delivery location of a request i without having visited the respective pickup location before. In the EB formulation, a flow of 1 enters an event-node associated with the pickup location of i due to (4.1c). Due to the construction of the event-based graph and flow constraints (4.1b), this flow originates in the depot (as we consider circulation flows, we may w.l.o.g. define the depot as source and sink of the circulation). Thus, and due to our definition of $\bar{x}_{ij}$, a flow of 1 has to leave set S. Moreover, due to construction of the event-based graph and our definition of $\bar{x}_{ij}$, all flow leaving a pickup location of i has to go to a delivery location of i, i.e., enter set S again. In total, the flow amongst nodes in S cannot exceed $|S| - 2$. Thus, constraints (3.12e) are fulfilled.

Constraints (3.12h) are fulfilled by the values of the variables $\bar{B}_j$, $j \in J\backslash\{0\}$ of a feasible solution of the LP relaxation of the LAEB model and by setting $\bar{B}_{0^-} \coloneqq \bar{B}_0$ and $\bar{B}_{0^+} \coloneqq e_0$.

Furthermore, constraints (3.12g) and (3.12i) are fulfilled by $Q_{0^+} = Q_{0^-} = 0$ and

$$Q_j = \mathbb{1}_P(j)\, q_j + \sum_{a\in\delta^{\text{in}}(v):v\in V_j} x_a \sum_{l=2}^{Q} q_{v_l} \qquad \forall j \in P \cup D$$

whereat $\mathbb{1}_P$ is the indicator function of P, i.e.,

$$\mathbb{1}_P(j) = \begin{cases} 1 & \text{if } j \in P, \\ 0 & \text{otherwise.} \end{cases}$$

As all terms are non-negative, $Q_j \geq 0$ for all $j \in \bar{J}$. For pickup locations $Q_j \geq q_j$; thus, $Q_j \geq \max\{0, \mathcal{I}_j q_j\}$ for all $j \in \bar{J}$. Due to construction of event nodes,

$$\sum_{l=1}^{Q} q_{v_l} \leq Q \qquad \forall v \in V$$

and

$$\sum_{l=2}^{Q} q_{v_l} \leq Q + \mathcal{I}_j q_j \qquad \forall v \in V$$

for delivery locations. Thus, $Q_j \leq \min\{Q, Q + \mathcal{I}_j q_j\}$ for all $j \in \bar{J}$ and constraints (3.12i) are fulfilled.

Due to construction of the event-based graph, there are no arcs (v, w) with $v_1 = i$ and $w_1 = j$ and

$$\mathbb{1}_P(i)\, q_i + \sum_{l=2}^{Q} q_{v_l} + \mathcal{I}_j q_j > \mathbb{1}_P(j)\, q_j + \sum_{l=2}^{Q} q_{w_l}$$

for any pair $i, j \in J$. Thus

$$\mathbb{1}_P(i)\, q_i + \sum_{l=2}^{Q} q_{v_l} + \mathcal{I}_j q_j \leq \mathbb{1}_P(j)\, q_j + \sum_{l=2}^{Q} q_{w_l}$$

holds for every $i, j \in J$ if $x_{(v,w)} > 0$ with $v_1 = i$ and $w_1 = j$ and thereby

$$\begin{aligned} &\sum_{(v,w)\in A: v_1=i,\, w_1=j} x_{(v,w)} \left(\mathbb{1}_P(i)\, q_i + \sum_{l=2}^{Q} q_{v_l} + \mathcal{I}_j q_j \right) \\ \leq &\sum_{(v,w)\in A: v_1=i,\, w_1=j} x_{(v,w)} \left(\mathbb{1}_P(j)\, q_j + \sum_{l=2}^{Q} q_{w_l} \right) \end{aligned} \tag{5.5}$$

for every $i, j \in J$. Moreover,

$$\mathbb{1}_P(i)\, q_i + \sum_{l=2}^{Q} q_{v_l} + \mathcal{I}_j q_j - Q \leq \mathbb{1}_P(j)\, q_j \qquad \forall i, j \in J, v \in V : v_1 = i$$

holds because of construction of the events. Thus,

$$\begin{aligned} &\left(1 - \sum_{(v,w):v_1=i,\, w_1=j} x_{(v,w)}\right) \left(\mathbb{1}_P(i)\, q_i + \sum_{l=2}^{Q} q_{v_l} + \mathcal{I}_j q_j - Q \right) \\ \leq &\left(1 - \sum_{(v,w):v_1=i,\, w_1=j} x_{(v,w)}\right) \mathbb{1}_P(j)\, q_j \quad \forall i, j \in J. \end{aligned} \tag{5.6}$$

Together, we get

$$
\begin{aligned}
& Q_i + \mathcal{I}_j q_j - Q\left(1 - \sum_{(v,w):v_1=i,\, w_1=j} x_{(v,w)}\right) \\
= \quad & \mathbb{1}_P(i)\, q_i + \underbrace{\sum_{a\in\delta^{\text{in}}(v):v\in V_i} x_a}_{\leq 1\ (5.4)} \sum_{l=2}^{Q} q_{v_l} + \mathcal{I}_j q_j - Q\left(1 - \sum_{(v,w):v_1=i,\, w_1=j} x_{(v,w)}\right) \\
\leq \quad & \mathbb{1}_P(i)\, q_i + \sum_{l=2}^{Q} q_{v_l} + \mathcal{I}_j q_j - Q\left(1 - \sum_{(v,w):v_1=i,\, w_1=j} x_{(v,w)}\right) \\
\overset{(5.5)-(5.6)}{\leq} \quad & \left(1 - \sum_{(v,w):v_1=i,\, w_1=j} x_{(v,w)}\right) \mathbb{1}_P(j)\, q_j + \sum_{(v,w):v_1=i,\, w_1=j} x_{(v,w)} \left(\mathbb{1}_P(j)\, q_j + \sum_{l=2}^{Q} q_{w_l}\right) \\
\leq \quad & \mathbb{1}_P(j)\, q_j + \sum_{a\in\delta^{\text{in}}(w):w\in V_j} x_a \sum_{l=2}^{Q} q_{w_l} \\
= \quad & Q_j \qquad \forall i, j \in J.
\end{aligned}
$$

With our choice of $\bar{x}_{ij}$, $i, j \in \bar{J}$ constraints (3.12g) are fulfilled. Together, the first step follows.

2. Figure 5.3 shows an instance with two requests, $q_1 = q_2 = 3$, and one vehicle. Furthermore, we assume that time windows are not binding. The figure shows feasible LP solutions for the EB formulation on the left side and for the LB formulation on the right side. Note that, only arcs with positive value for $x_{(v,w)}$ and $\bar{x}_{ij}$, respectively, are included in the figure. In the upper part of the figure, the vehicle capacity is 6, i.e., both customers can be transported simultaneously. The graphs of both approaches show the solutions

$$
\begin{aligned}
& 0 \to 1^+ \to 2^+ \to 2^- \to 1^- \to 0 \\
& 0 \to 1^+ \to 1^- \to 2^+ \to 2^- \to 0
\end{aligned}
$$

both with a weight of 0.5. If we reduce the vehicle capacity to 5, the first solution is not integer feasible any more. While the event-based graph pictures this fact (lower left part of the figure), the solution in the LB formulation does not change. The reason is that capacity constraints are still fulfilled if the first solution is weighted with 0.5 and

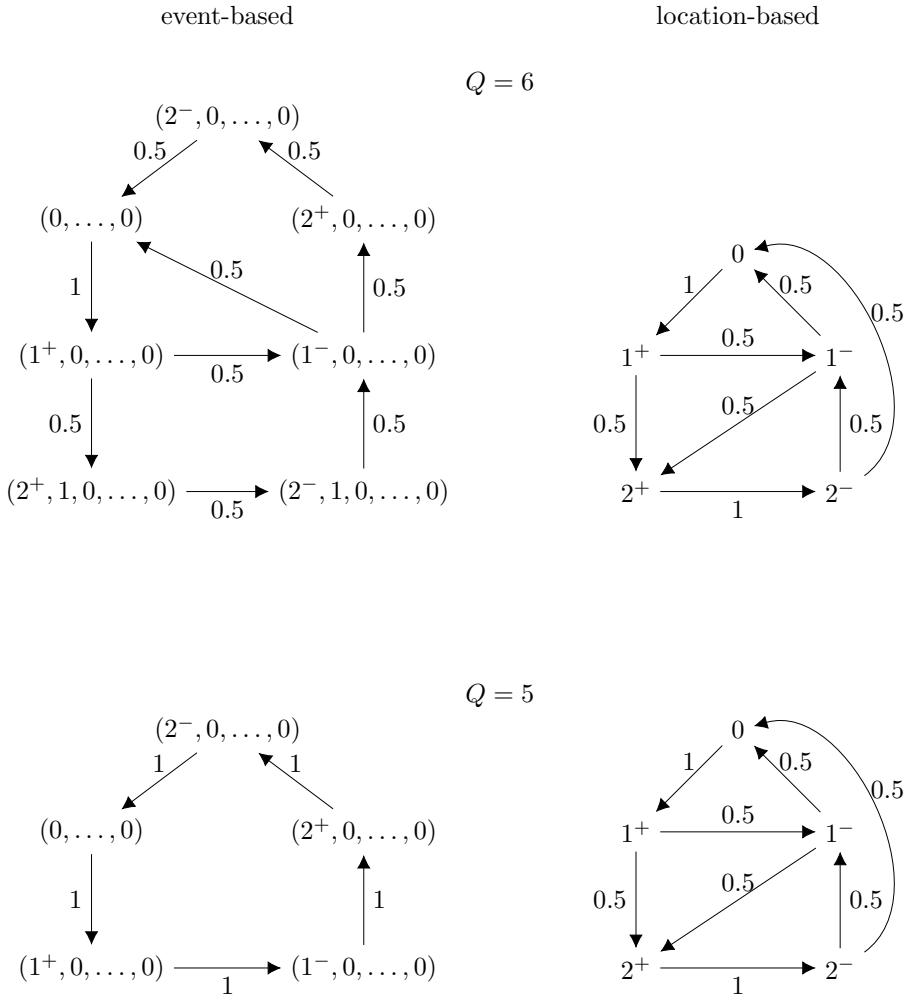


Figure 5.3: Example for step two in the proof of Theorem 5.4.

$Q_0 = 0, Q_{1^+} = 3$, $Q_{1^-} = 0$, $Q_{2^+} = 5$, and $Q_{2^-} = 2$ as the following evaluation shows:

	$Q_i + \mathcal{I}_j q_j - Q(1 - x_{ij})$	$\leq Q_j$
$0 \to 1^+$	$0 + 3 - 5 \cdot (1 - 0.5 - 0.5) = 3$	≤ 3
$1^+ \to 1^-$	$3 - 3 - 5 \cdot 0.5 = -2.5$	≤ 0
$1^+ \to 2^+$	$3 + 3 - 5 \cdot 0.5 = 3.5$	≤ 5
$1^- \to 0$	$0 + 0 - 5 \cdot 0.5 = -2.5$	≤ 0
$1^- \to 2^+$	$0 + 3 - 5 \cdot 0.5 = 0.5$	≤ 5
$2^+ \to 2^-$	$5 - 3 - 5 \cdot (1 - 0.5 - 0.5) = 2$	≤ 2
$2^- \to 0$	$2 + 0 - 5 \cdot 0.5 = -0.5$	≤ 0
$2^- \to 1^-$	$2 - 3 - 5 \cdot 0.5 = -3.5$	≤ 0

In all other cases, $\bar{x}_{ij} = 0$. Since $Q_j \geq \max\{0, \mathcal{I}_j q_j\}$, it holds that $Q_i - Q \leq 0 \leq Q_j - \mathcal{I}_j q_j$ and the inequality is fulfilled in these cases. Thus, there is an LP solution in the LB formulation which is not feasible for the EB formulation.

3. It remains to show that these LP solutions can be optimal. With the travel costs in

	0	1^+	2^+	1^-	2^-
0	0	1	1	10	10
1^+	1	0	2	9	9
2^+	1	2	0	9	9
1^-	10	9	9	0	1
2^-	10	9	9	1	0

Table 5.1: Example for step three in the proof of Theorem 5.4.

Table 5.1 the solution $0 \to 1^+ \to 2^+ \to 2^- \to 1^- \to 0$ has an objective value of 23 and the solution $0 \to 1^+ \to 1^- \to 2^+ \to 2^- \to 0$ an objective value of 38. Moreover, swapping 1^+ and 2^+ or 2^- and 1^- in the first solution or swapping the indices of customers 1 and 2 leads to the same objective values due to symmetry. Therefore, the LB model will use as much as possible of the first solution and its symmetric ones (such that capacity constraints stay fulfilled) while the EB model uses the second solution. Thus, $z^{rel}_{LAEB} > z^{rel}_{LB}$ holds.

□

The proof highlights again that the LAEB and the ALAEB formulation implicitly ensure that pairing, precedence, and capacity constraints are fulfilled. The implicit implementation of capacity constraints leads even to a tighter polyhedron of the LP relaxation. Due to their exponential size, pairing and precedence constraints (3.12e) are typically not added upfront but only if they are violated. Therefore, the LAEB and the ALAEB LP relaxation are also tighter in this aspect. However, we still face a significant number of nodes and arcs in the event-based graph, which we compactify in Section 5.3.

5.3 Preprocessing and Branch-and-Cut Methods

We present methods to reduce the size of the event-based graph in preprocessing and in branch-and-cut nodes as well as new types of additional valid inequalities in this section. Figure 5.4 gives an overview of the developed methods.

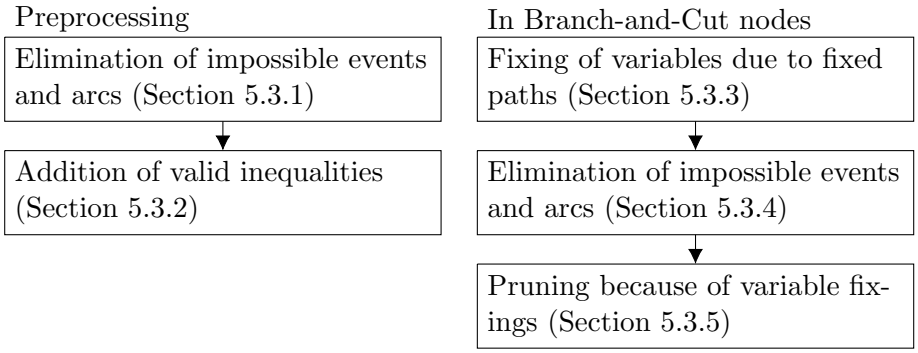


Figure 5.4: Overview of developed methods.

First, we introduce preprocessing methods to eliminate impossible events and arcs in the event-based graph based on bounds for B_v variables (Section 5.3.1). Moreover, we propose new valid inequalities to avoid subtours, incompatible events, and infeasible paths (Section 5.3.2). Second, methods are presented to improve the search in branch-and-cut nodes based on previous branching decisions, see Sections 5.3.3–5.3.5.

5.3.1 Elimination of Impossible Event Nodes and Arcs in Preprocessing

We already explained in Section 4.1 that the number of events can be reduced by considering pairwise incompatibilities with respect to time windows and ride time constraints. We present a method to systematically compute earliest and latest starting times of service at all events. The earliest starting time B_v^{LB} at node v can be interpreted as the shortest path between depot event $(0,\dots,0)$ and the considered event $(v_1,\dots,v_Q)$, i.e., the fastest way to pick up first customers $v_2,\dots,v_Q$ (for all $v_j \neq 0$) and v_1 afterwards if $v_1 \in P$ or deliver v_1 afterwards if $v_1 \in D$ while respecting the corresponding time windows. Analogously, the latest starting time of service B_v^{UB} in $(v_1,\dots,v_Q)$ is the latest time for start of service such that customers $v_1,\dots,v_Q$, $v_j \neq 0$, can all be delivered within the time windows. Naturally, an event cannot be feasible if $B_v^{UB} < B_v^{LB}$. An event v is also infeasible if there is a customer i in the vehicle who cannot be delivered within the maximum ride time, i.e., if $\bar{B}_i^{UB} + s_i + L_i - s_{v_1} - t_{v_1,i^-} < B_v^{LB}$, whereat $\bar{B}_i^{UB} = \max_{v\in V_{i^+}}\{B_v^{UB}\}$. Moreover, an arc (v,w) is infeasible if $B_v^{LB} + s_{v_1} + t_{(v,w)} > B_w^{UB}$. Then, we can eliminate (v,w) from the graph.

Due to construction of the event-based graph,

$$B_v^{LB} = \max\left\{ e_{v_1}, \min_{w:(w,v)\in\delta^{in}(v),\, w\in\bigcup_{i=1}^{n} V_{i^+}} \left\{ B_w^{LB} + s_{w_1} + t_{(w,v)} \right\} \right\} \tag{5.7}$$

and

$$B_v^{UB} = \min \Bigg\{ \ell_{v_1} - \sum_{i \in R} \mathbb{1}_{\bigcup_{i=1}^{n} V_{i^-}}(v) \cdot \max\{0, \ell_{v_1} - (\bar{B}_i^{UB} + s_i + L_i)\}, \\ \max_{w:(v,w)\in\delta^{out}(v),\, w\in\bigcup_{i=1}^{n} V_{i^-}} \left\{ B_w^{UB} - t_{(v,w)} - s_{v_1} \right\}, \\ \min_{i\in R|\exists l\geq 2: v_l = i} \{\bar{B}_i^{UB} + s_i + L_i - s_{v_1} - t_{v_1,i^-}\} \Bigg\}. \tag{5.8}$$

We can compute B_v^{LB} and B_v^{UB} in $\mathcal{O}(|V|)$ by evaluating the events v systematically in the correct sequence to ensure that all predecessors (with $(w,v) \in A$) and successors (with $(v,w) \in A$), respectively, are evaluated before. B_v^{LB} is computed in the following sequence:

1. v_1 is a pickup location and ...
   a) $v_2 = \ldots = v_Q = 0$, i.e., all events where a customer is entering an empty vehicle.
   b) $v_2 \neq 0$, and $v_3 = \ldots = v_Q = 0$.
   c) $v_2, v_3 \neq 0$, and $v_4 = \ldots = v_Q = 0$.
   d) ...
   e) $v_2, \ldots, v_Q \neq 0$.
2. v_1 is a delivery location.

Note that, the vehicle could also drive from a delivery location to a pickup location. However, because of the triangle inequality this path cannot be shorter than not visiting neither the pickup nor the delivery location of the corresponding customer. Therefore, it is reasonable to require $w \in \bigcup_{i=1}^{n} V_{i^+}$ in (5.7). For the same reason we only need B_w^{LB} of events where w_1 is a pickup location to compute B_v^{LB} for events where v_1 is a delivery location.

Because of the triangle inequality the shortest path between an event v and the end depot $(0, \ldots, 0)$ is to deliver customers $v_1, \ldots, v_Q$ without picking up a new customer. When we first iterate over $v \in V$ to compute B_v^{UB}, maximum ride times cannot be considered yet, because we need the upper bounds B_v^{UB} of all pickup nodes to compute $\bar{B}_i^{UB}$. However, for the computation of B_v^{UB}, where v is a pickup node, we need the upper bounds of all delivery nodes. Hence, in the first loop, we omit maximum ride times from the computation of B_v^{UB}. The following sequence is most efficient to determine B_v^{UB}:

1. v_1 is a delivery location and ...
   a) $v_2 = \ldots = v_Q = 0$, i.e., all events where v_1 is the last customer leaving the vehicle.
   b) $v_2 \neq 0$, and $v_3 = \ldots = v_Q = 0$.
   c) $v_2, v_3 \neq 0$, and $v_4 = \ldots = v_Q = 0$.
   d) ...

e) $v_2, \ldots, v_Q \neq 0$.

2. v_1 is a pickup location.

In total, all events in V have to be considered once to determine B_v^{LB} and once to determine B_v^{UB} without consideration of maximum ride times. After all upper bounds B_v^{UB} are computed, we are able to determine $\bar{B}_i^{UB}$, $i \in R$. We repeat the procedure to compute B_v^{UB}, this time including ride times. So, one iteration of the procedure requires an effort of $\mathcal{O}(|V|)$.

After computing B_v^{UB} for an event v, we can directly check whether $B_v^{UB} < B_v^{LB}$ and delete the event if so. Due to our sequence to compute B_v^{UB} values, the event is not considered in (5.8) any more to compute upper bounds for its predecessors.

However, if an event v with $v_1 \in P$ is deleted, lower bounds B_w^{LB} of its successor events with $(v, w) \in A$ might also change. Therefore, we store all of these successors in a list and update their lower bounds afterwards. When going through the list, deleting an event w can also lead to updated upper bounds for predecessors if $w_1 \in D$. Moreover, $\bar{B}_i^{UB}$ can change if B_v^{UB} changes for a $v \in V_{i^+}$, $i \in R$. Thus, we need three lists, one for predecessors, one for successors, and one to store all i for which $\bar{B}_i^{UB}$ has to be updated. Then, we run alternately through the lists in the sequence described above until the lists are empty. In the worst case, only one bound changes in each iteration.

Whenever updating a lower or upper bound of event v, we also check feasibility of all in- and outgoing arcs ($(w, v) \in \delta^{in}(v)$ and $(v, w) \in \delta^{out}(v)$, respectively). In our tests, the computation of lower and upper bounds for all nodes and the detection of infeasible arcs was $< 0.1s$ for instances with $n = 100$ and $Q = 6$.

We can use the bounds to add the following valid inequalities to the EB formulation

$$B_v^{LB} + (\ell_{i^-} - L_i - s_i - B_v^{LB})\left(1 - \sum_{a \in \delta^{\text{in}}(v)} x_a\right) \leq B_v \leq B_v^{UB} \qquad \forall i \in R, v \in V_{i^+} \quad (5.9)$$

$$B_v^{LB} \leq B_v \leq e_{i^+} + s_i + L_i + (B_v^{UB} - (e_{i^+} + s_i + L_i)) \sum_{a \in \delta^{in}(v)} x_a \qquad \forall i \in R, v \in V_{i^-} \quad (5.10)$$

and the following valid inequalities to the location-augmented-event-based formulations:

$$\sum_{v: v_1 = j} B_v^{LB} \cdot \sum_{a \in \delta^{in}(v)} x_a \leq \bar{B}_j \leq \sum_{v: v_1 = j} B_v^{UB} \cdot \sum_{a \in \delta^{in}(v)} x_a \qquad \forall j \in P \cup D \quad (5.11)$$

5.3.2 Addition of Valid Inequalities in Preprocessing

We can add several further valid inequalities to the model formulation identifying events which cannot occur simultaneously.

Infeasible Paths In the literature, several authors introduced infeasible paths constraints, see, e. g., Ascheuer et al. (2000), Cordeau (2006), and Ropke et al. (2007), based on sequences of request locations which cannot occur due to time windows. Certainly, if a path of locations $i \to j \to k$ would lead to a violated time window, this is also true for all

event paths $v \to w \to u$ with $v_1 = i$, $w_1 = j$, and $u_1 = k$. However, it might be that the event path $v \to w \to u$ is infeasible although the location path $i \to j \to k$ is feasible (e.g., because in event v another customer is in the vehicle who leads to a later departure time at i). Thus, it is possible to add a larger amount of infeasible path constraints based on events.

Let v be an event. Compute for all events w with $(w, v) \in A$ the earliest starting time $B^{LB}_{wv} = B^{LB}_{w} + s_{w_1} + t_{(w,v)}$ at v coming from w and for all events u with $(v, u) \in A$ the latest possible starting time of service $B^{UB}_{vu} = B^{UB}_{u} - s_{v_1} - t_{(v,u)}$ in v to reach u on time. If $B^{UB}_{vu} < B^{LB}_{wv}$, $x_{(w,v)} + x_{(v,u)} \leq 1$ is a valid inequality which can be lifted to

$$\sum_{w' \in V : (w',v) \in A,\, B^{LB}_{w'v} \geq B^{LB}_{wv}} x_{(w',v)} + \sum_{u' \in V : (v,u') \in A,\, B^{UB}_{vu'} \leq B^{UB}_{vu}} x_{(v,u')} \leq 1 \quad \forall v \in V \setminus V_0 \tag{5.12}$$

by including other predecessor (successor) events leading to a later earliest (earlier latest) starting time of service in v.

In general, let $\bar{S} = \{(u^1, u^2), \ldots, (u^{\bar{m}-1}, u^{\bar{m}})\}$ be a path $u^1 \to u^2 \to \cdots \to u^{\bar{m}}$, which is infeasible if $B^{LB}_{u^1} + \sum_{m=1}^{\bar{m}-1} s_{u^m} + t_{(u^m, u^{m+1})} > B^{UB}_{u^{\bar{m}}}$, i.e., if location $u^{\bar{m}}$ is not reached before its latest departure time. Then, $\sum_{a \in \bar{S}} x_a \leq \bar{m} - 2$ is a valid inequality.

This valid inequality can be lifted in two ways as Figure 5.5 shows. In line with the

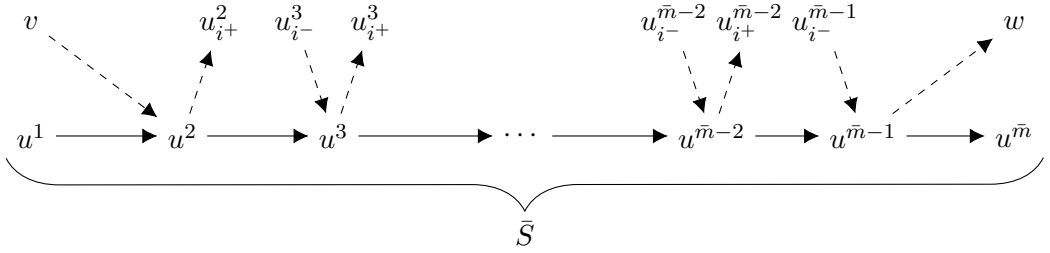


Figure 5.5: Lifting for infeasible paths.

argumentation above, $\sum_{a \in \bar{S}} x_a \leq \bar{m} - 2$ can be lifted by adding $x_{(v,u^2)}$ if $B^{LB}_{vu^2} \geq B^{LB}_{u^1 u^2}$ and $x_{(u^{\bar{m}-1}, w)}$ if $B^{UB}_{u^{\bar{m}-1}, w} \leq B^{UB}_{u^{\bar{m}-1} u^{\bar{m}}}$ on the left-hand side. Moreover, further paths between u^2 and $u^{\bar{m}-1}$ can be added if $(i^-, 0, \ldots, 0) \notin \{u^2, \ldots, u^{\bar{m}-1}\}$ for all $i \in R$, i.e., the vehicle is not empty in between. In Figure 5.5, $u^m_{i^+}$ is the event where customer i enters the vehicle and all customers who are in the vehicle after event u^m are still there. Analogously, $u^m_{i^-}$ is the event where customer i leaves the vehicle and all customers who are present in the vehicle directly before event u^m are in the vehicle. After each of the path's locations the vehicle can leave it to visit the pickup location of a further customer i and come back to the path when customer i is delivered. This might be directly, i.e., the path $u^m \to u^m_{i^+} \to u^{m+1}_{i^-} \to u^{m+1}$ replaces the path $u^m \to u^{m+1}$, or indirectly, i.e., the vehicle also visits further locations between pickup and delivery of customer i or drives from i's delivery location to a later location of the original path. In the direct case, arcs $(u^m, u^m_{i^+})$ and $(u^{m+1}_{i^-}, u^{m+1})$ replace arc (u^m, u^{m+1}) such that we have to weight both added arc variables with a factor $1/2$ to ensure feasibility of the lifted inequality. Note that, we only "replace" the original arc conceptually in the solution, but we do not actually remove it from the lifted inequality. With respect to feasibility of the lifting, there are three cases:

- In generalization of the already discussed situation, $u^m \to u^m_{i^+} \to \cdots \to u^{m+1}_{i^-} \to u^{m+1}$ replaces $u^m \to u^{m+1}$, which is a feasible lifting, as one arc with a factor of 1 is replaced by two arcs with a factor of 1/2. Note that, we do not need to consider the path between customer i's pickup and delivery. Due to the triangle inequality, the arrival time at u^{m+1} cannot be earlier than in the original path independent of the path between customer i's pickup and delivery.
- $u^m \to u^m_{i^+} \to \cdots \to u^{m'}_{i^-} \to u^{m'}$ with $m' > m+1$ replaces the path $u^m \to \cdots \to u^{m'}$, which is a feasible lifting, as at least two arcs with a factor of 1 ((u^m, u^{m+1}) and $(u^{m'-1}, u^{m'})$) are replaced by two arcs with a factor of 1/2. Again, we do not need to consider the path between customer i's pickup and delivery here.
- If any path including arcs $(u^m_{i^-}, u^m)$ and $(u^{m'}, u^{m'}_{i^+})$ with $m' \geq m$ is part of the lifted path, the path is already infeasible due to precedence relations.

Note that, the event $u^m_{i^+}$ ($u^m_{i^-}$) or arc $(u^m, u^m_{i^+})$ ($(u^m_{i^-}, u^m)$) might not exist for some $i \in R$ and $m \in \{2, \ldots, \bar{m}-2\}$. Then, the corresponding arcs are simply not added to the valid inequality. As all other corresponding arcs are added, the lifting includes also paths where the delivery location visited right before the vehicle returns to the original path needs not to belong to the same customer as the pickup location visited directly after leaving the original path. Together,

$$\sum_{(v,u^2)\in A, v\neq u_1 : B^{LB}_{vu^2} \geq B^{LB}_{u^1u^2}} x_{(v,u^2)} + \sum_{a\in\bar{S}} x_a + \sum_{m=2}^{\bar{m}-2} \sum_{(u^m,w)\in A: w_1 \in P} \frac{1}{2}\cdot x_{(u^m,w)}$$
$$+ \sum_{m=3}^{\bar{m}-1} \sum_{(w,u^m)\in A: w_1 \in D} \frac{1}{2}\cdot x_{(w,u^m)} + \sum_{(u^{\bar{m}-1},w)\in A, w\neq u^{\bar{m}} : B^{UB}_{u^{\bar{m}-1}w} \leq B^{UB}_{u^{\bar{m}-1}u^{\bar{m}}}} x_{(u^{\bar{m}-1},w)} \leq \bar{m}-2 \tag{5.13}$$

is a valid inequality. We implemented inequalities (5.13) for the case of $\bar{m} = 4$. In total, for each arc $(v,u) \in A$ we construct one inequality of type (5.12), hence there are $\mathcal{O}(|A|)$ valid inequalities. In the case of inequalities (5.13), we construct each inequality from an arc $(u^2, u^3) \in A$ and u^3's successors, hence there are $\mathcal{O}\big(|A|\cdot(|R|+|Q|)\big)$ valid inequalities.

For the implementation of inequalities (5.12) and (5.13) we first compute B^{LB}_{vw} and B^{UB}_{vw} for all $(v,w) \in A$. To implement inequalities (5.12), for all events v, we loop over all successors u (in decreasing order of B^{UB}_{vu}), determine the first predecessor w of v with $B^{UB}_{vu} < B^{LB}_{wv}$, and add all valid inequalities of type (5.12), i.e., for each v we add the inequality with variables $x_{(v,u')}$ such that $B^{UB}_{vu'} \leq B^{UB}_{vu}$ and variables $x_{(w',v)}$ such that $B^{LB}_{w'v} \geq B^{LB}_{wv}$. Note that there is a dominance if for two successors $\bar{u}^1$ and $\bar{u}^2$ of the decreasing ordered list with $B^{UB}_{v\bar{u}^1} \geq B^{UB}_{v\bar{u}^2}$ the set of arcs (w,v) with $B^{UB}_{v\bar{u}^1} < B^{LB}_{wv}$ and $B^{UB}_{v\bar{u}^2} < B^{LB}_{wv}$, respectively, are identical. Then, the valid inequality (5.12) for $\bar{u}^2$ is equal or weaker to the one for $\bar{u}^1$ and therefore not added. For the implementation of (5.13) we consider the case $\bar{m} = 4$. For all arcs $(u^2,u^3) \in A \setminus \big(\delta^{\text{in}}((0,\ldots,0)) \cup \delta^{\text{out}}((0,\ldots,0))\big)$, we loop over all successors u^4 of u^3 (in decreasing order of $B^{UB}_{u^3u^4}$) and determine the first predecessor u^1 (in increasing order of $B^{LB}_{u^1u^2}$) of u^2 for which the path $u^1 \to u^2 \to u^3 \to u^4$

is infeasible. We can apply the same dominance criterion like in the implementation of (5.12) here. Again, we add all variables $x_{(u^3,w)}$ with $B^{UB}_{u^3w} \leq B^{UB}_{u^3u^4}$, $w \neq u^4$, and all variables $x_{(v,u^2)}$ with $B^{LB}_{vu^2} \geq B^{LB}_{u^1u^2}$, $v \neq u^1$, to the left side of the inequality. We loop over all requests $i \in R$ and check whether events $u^2_{i^+}$ and $u^3_{i^-}$ and arcs $(u^2, u^2_{i^+})$ and $(u^3_{i^-}, u^3)$ exist. If so, we add $\frac{1}{2} \cdot x_{(u^2,u^2_{i^+})}$ and $\frac{1}{2} \cdot x_{(u^3_{i^-},u^3)}$ to the inequality.

Vehicle Sharing Let i and j be two customers. If they use the same vehicle, only one of them can enter the vehicle while the other is already sitting in it. Both of these events are incompatible to an event where i or j leaves the vehicle before the other one entered it. Thus,

$$\sum_{(v,w)\in A: w_1=j^+,\, \exists l: w_l=i} x_{(v,w)} + \sum_{(v,w)\in A: w_1=i^+,\, \exists l: w_l=j} x_{(v,w)} + \sum_{(v,w)\in A: v_1=i^-,\, w_1=j^+} x_{(v,w)} + \sum_{(v,w)\in A: v_1=j^-,\, w_1=i^+} x_{(v,w)} \leq 1 \quad \forall i,j \in R : i < j \tag{5.14}$$

are valid inequalities. We have $\mathcal{O}(|R|^2)$ valid inequalities of type (5.14) and add all of them in the preprocessing.

Analogously, only one of them can leave the vehicle while the other is still sitting in it or one of them can leave the vehicle before the other enters it. Hence,

$$\sum_{(v,w)\in A: w_1=i^-,\, \exists l: w_l=j} x_{(v,w)} + \sum_{(v,w)\in A: w_1=j^-,\, \exists l: w_l=i} x_{(v,w)} + \sum_{(v,w)\in A: v_1=i^-,\, w_1=j^+} x_{(v,w)} + \sum_{(v,w)\in A: v_1=j^-,\, w_1=i^+} x_{(v,w)} \leq 1 \quad \forall i,j \in R : i < j \tag{5.15}$$

are valid inequalities. We have $\mathcal{O}(|R|^2)$ valid inequalities of type (5.15) and add all of them in the preprocessing.

Moreover, an event where i or j leaves the vehicle directly after the other one, is incompatible to an event where one of them left the vehicle directly before the other one entered it. Thus,

$$\sum_{(v,w)\in A: v_1=i^-,\, w_1=j^-} x_{(v,w)} + \sum_{(v,w)\in A: v_1=j^-,\, w_1=i^-} x_{(v,w)} + \sum_{(v,w)\in A: v_1=i^-,\, w_1=j^+} x_{(v,w)} + \sum_{(v,w)\in A: v_1=j^-,\, w_1=i^+} x_{(v,w)} \leq 1 \quad \forall i,j \in R : i < j \tag{5.16}$$

are valid inequalities. We have $\mathcal{O}(|R|^2)$ valid inequalities of type (5.16) and add all of them in the preprocessing.

Furthermore, if we select an arc (v, w) with $v_1 \in D$ and $w_1 \in P$, customers v_1 and w_1 cannot share a vehicle simultaneously. Thus,

$$\sum_{(v,w)\in A: v_1=j^-,\, w_1=i^+} x_{(v,w)} + \sum_{(v,w)\in A: v_1=i^-,\, w_1=j^+} x_{(v,w)}$$

$$+ \sum_{(u',u)\in A: u_1=k,\, \exists l_1: u_{l_1}=i,\, \exists l_2: u_{l_2}=j} x_{(u',u)} \leq 1 \tag{5.17}$$

is a valid inequality for all pairs of customers $i, j \in R$, $i < j$, being pairwise compatible and a further location $k \in J \backslash \{0, i^+, i^-, j^+, j^-\}$. We have $\mathcal{O}(|J| \cdot |R|^2)$ valid inequalities of type (5.17) and add all of them in the preprocessing.

Customer Incompatibility With the pairwise incompatibilities presented in Section 4.1 we can identify customer pairs i and j which cannot be served by the same vehicle due to time windows or ride time in advance. Our procedure in Section 5.3.1 eliminates all events where i and j are simultaneously in the vehicle. If the paths $i^+ \to i^- \to j^+ \to j^-$ and $j^+ \to j^- \to i^+ \to i^-$ are infeasible due to time windows, we can conclude that at most one of them can share the vehicle with another customer k. Otherwise customers i and j would be served by the same vehicle (not necessarily at the same time) due to transitivity. Thus,

$$\begin{aligned} &\sum_{(u,v)\in A: v_1=k^+,\, \exists l: v_l=i} x_{(u,v)} + \sum_{(u,v)\in A: v_1=i^+,\, \exists l: v_l=k} x_{(u,v)} \\ &+ \sum_{(u,v)\in A: v_1=k^+,\, \exists l: v_l=j} x_{(u,v)} + \sum_{(u,v)\in A: v_1=j^+,\, \exists l: v_l=k} x_{(u,v)} \leq 1 \end{aligned} \tag{5.18}$$

is a valid inequality. We have $\mathcal{O}(|R|^3)$ valid inequalities of type (5.18). However, the valid inequality is only relevant if i and j are incompatible, but both of them are compatible with k. We add all of them in the preprocessing.

5.3.3 Fixing of Variables Due to Fixed Paths

Knowing which customers are together in a fixed path, i.e., a sequence of events connected by arcs whose variables are fixed to 1, also leads to further incompatible events if the depot is part of the path (see Schulz and Pfeiffer (2024)). We update the set of fixed paths when an upward branch, i.e., a branch where an x or $\bar{x}$ variable is fixed to 1, is created. Let two fixed paths be given which start in the depot. As both have to be served by different vehicles, customers in one of the paths cannot share a vehicle with customers of the other path. Let customer i be in the first and customer j be in the second path. Then, we can eliminate all events v with $v_{l_1} \in \{i, i^+, i^-\}$ and $v_{l_2} \in \{j, j^+, j^-\}$ with $l_1, l_2 \in \{1, \ldots, Q\}$ and their incident arcs. The same is true if we consider two fixed paths ending in the depot. Thus, we check such incompatibilities whenever a fixed path is merged with another fixed path containing the depot.

5.3.4 Elimination of Impossible Events and Arcs in Branch-and-Cut Nodes

Variable fixings in branch-and-cut nodes influence the lower and upper bounds computed in the preprocessing, see Section 5.3.1. While for the EB formulation and the LAEB formulation of the DARP we branch on variables x_a, $a \in A$, using the ALAEB formulation branching takes place on variables $\bar{x}_{ij}$, $i, j \in J$. If a variable $x_{(v,w)}$ is fixed to 0 or to 1, either directly because a new branch for the EB or the LAEB model is created, or as

an indirect result of another fixed variable $x_{(v',w')}$ or $\bar{x}_{ij}$, the effects on upper and lower bounds are as follows: If $x_{(v,w)}$ is fixed to 0, w cannot be the successor of v any more. Thus, w can be excluded in the maximum in (5.8). Analogously, w can be excluded in the minimum in (5.7) if $x_{(w,v)}$ is fixed to 0. If we fix $x_{(v,w)}$ to 1, w is determined as the successor of v such that we can replace (5.7) by

$$B_w^{LB} = \max\{e_{w_1},\ B_v^{LB} + s_{v_1} + t_{(v,w)}\} \tag{5.19}$$

and (5.8) by

$$B_v^{UB} = \min\left\{\ell_{v_1} - \sum_{i\in R} \mathbb{1}_{\bigcup_{i=1}^{n} V_{i^-}}(v) \cdot \max\{0,\ \ell_{v_1} - (\bar{B}_i^{UB} + s_i + L_i)\},\right.$$
$$\left. B_w^{UB} - t_{(v,w)} - s_{v_1},\ \min_{i\in R|\exists l\geq 2: v_l = i}\{\bar{B}_i^{UB} + s_i + L_i - s_{v_1} - t_{v_1,i^-}\}\right\}. \tag{5.20}$$

In other words, the events v and w are merged to a single event if $x_{(v,w)} = 1$ Schulz and Pfeiffer (cf. 2024). Moreover, we eliminate events and arcs from the graph which have become infeasible due to variable fixings in a branch-and-cut node. For example, if arc $x_{(v,w)}$ is fixed to 1 and w is a pickup node, we know that all other events $u \in V$, $u_1 = w_1$, $u \neq w$, are infeasible for the subtree rooted in the current branch-and-cut node.

After we ran the procedures described in this and the previous section, we store all events v for which either B_v^{LB} or B_v^{UB} changed and update lower and upper bounds for the beginning of service of all possible predecessor and successor events as described at the end of Section 5.3.1. Moreover, we check feasibility of events v and arcs (v, w), i.e., check if $B_v^{LB} \leq B_v^{UB}$ and $B_v^{LB} + s_{v_1} + t_{(v,w)} \leq B_w^{UB}$, respectively.

If fixing of variables $x_{(v,w)}$ or the update of lower and upper bound results in an infeasible event, the event is removed from the graph together with its incident arcs. Finally, we add new inequalities of types (5.9)–(5.11) if bounds changed, and inequalities $x_a \leq 0$ if the arc a is declared infeasible.

5.3.5 Pruning Because of Variable Fixing in Branch-and-Cut Nodes

If one of the eliminated events v is already merged with other events, i.e., is incident to an arc for which the corresponding arc variable is fixed to 1, the branch-and-cut node can be pruned.

5.4 Numerical Results

In this section, we evaluate the efficiency of the presented MILP models, lower and upper bounding strategies, and valid inequalities. Comparing the EB and the LAEB model, the same branching decisions can be applied, as in both models branching takes place on variables x_a, $a \in A$. The LAEB model has fewer variables, as we replace variables B_v, $v \in V$, by variables $\bar{B}_j$, $j \in J$. Hence, we expect the LAEB model to be more efficient. Comparing our two location-augmented-event-based formulations, the LAEB model has

fewer variables but more binary variables. Moreover, variable fixing in Section 5.3.5 can more likely be applied, as we directly branch on the arcs of the event-based graph. Therefore, a branch where an x_a, $a \in A$, variable is fixed to 1 directly implies that both adjacent events occur. This branching step contains significantly more information than just fixing a single relation between two locations. On the other hand, a sequence of variables $\bar{x}_{ij}$ has to be fixed to 1 to ensure that an event occurs if we branch on $\bar{x}_{ij}$ variables, but we have significantly fewer $\bar{x}_{ij}$ variables. Taking everything together, it is unclear which advantage predominates. We compare all three formulations in a computational study.

Our computations are performed on an Intel Core i7-8700 CPU, 3.20 GHz, 32 GB memory and implemented in C++ using CPLEX 12.10. The code can be found in the git repository Gaul (2023). The time limit for the solution in all tests was set to 7200 seconds. Note, that the execution of CPLEX was limited to one thread only and the search method was limited to traditional branch-and-cut. Throughout this section, we use some abbreviations which are summarized in Table 5.2. We first analyze the performance of

Abbreviation	Explanation/ Reference
General abbreviations	
Inst.	Name of instance
Obj.	Objective Value
CPU	Computational time in seconds
N/A	Not applicable, no integer solution found within the time limit
Gap	Gap obtained by CPLEX solution (in percent)
rGap	Root node gap (in percent)
Avg. rGap	Average root node gap (in percent)
# Impr.	Number of instances with improved computational time
Avg. Dev.	Average deviation of CPU time compared to the benchmark (first row of the corresponding table)
Avg.* Dev.	Avg. Dev. only on instances with improved computational time
Preprocessing	
GP	Graph preprocessing, see Section 5.3.1
VS1–VS4	Vehicle Sharing, see (5.14)–(5.17)
CI1	Customer Incompatibility, see (5.18)
IP1–IP2	Infeasible Paths, see (5.12)–(5.13)
Branch-and-Cut Algorithm	
LBsUBs	Add new inequalities of type (5.9)–(5.11) if bounds changed, see Section 5.3.4
FA	Add inequalities $x_a \leq 0$ for infeasible arcs a, see Section 5.3.4
FP	Update the set of fixed paths and eliminate events (and incident arcs) which represent users of two different paths connected to the start or end depot together in the vehicle, see Section 5.3.3

Table 5.2: Abbreviations used in Section 5.4.

the presented MILP formulations (Section 5.4.1). Then, our preprocessing components are validated in Section 5.4.2. Afterwards, we test the components for the branch-and-cut algorithm (Section 5.4.3). We present our final results on benchmark instances in Section 5.4.4.

5.4.1 Comparing the MILP formulations

The EB, LAEB, and ALAEB formulations are compared against each other using the a- and b-benchmark instances (see Section 4.4.1 for a description of the benchmark instances). Furthermore, we use an extended version of these instances denoted by a-X and b-X from Gschwind and Irnich (2015), where the time window length is doubled by postponing the upper bound of the pickup or delivery time window by 15 minutes, i.e., ℓ_i += 15. Due to the larger time windows these instances are harder to solve. Note, that we do not increase the vehicle capacity as done in Gschwind and Irnich (2015). Although this would increase the complexity of the models, a vehicle capacity of $Q = 12$ is a rather unrealistic assumption for ridepooling services (compare, e.g., the ridepooling cabs of MOIA[1] or *Hol mich! App*[2], both based in Germany, which have a capacity of $Q = 6$). Previous studies have shown that even six seats are rarely fully occupied in practical relevant settings (Pfeiffer and Schulz, 2021). Thus, extended time windows are more relevant for the practical application than larger vehicles. The extended instances are denoted as aK-n-X and bK-n-X, where K indicates the number of vehicles and n denotes the number of requests. In the a- and a-X-instances, $Q = 3$ and $L_i = 30$ and $q_i = 1$ for all $i \in R$, whereas in the b- and b-X-instances $Q = 6$ and $L_i = 45$ and $q_i \in \{1, \dots, 6\}$ for all $i \in R$.

The results are presented in Tables 5.3 and 5.4. On the smaller as well as on the larger benchmark instances, the LAEB model outperforms the other two formulations, but its superiority becomes most evident when comparing its performance to the other formulations on the harder a-X and b-X instances. The average root node gap for all three formulations is very small and ranges from 1.1% to 5.4%, which demonstrates that, although some instances still take a long time to solve, the MILP formulations are already very tight.

To sum it up, it seems that the smaller number of integer variables in the ALAEB formulation does not make up for the loss of information compared to the EB or LAEB formulation when branching on $\bar{x}_{ij}$ variables instead of $x_{(v,w)}$. Comparing the EB and the LAEB model, the replacement of variables B_v, $v \in V$, (of which there may be $\mathcal{O}(n^Q)$ (see Section 4.1) by variables $\bar{B}_j$, $j \in J$, (of which there are $2n$ variables) as well as the tighter big-M formulation explain the presented speedup. Since the LAEB formulation improves on the EB formulation and turns out to be the best formulation, we restrict ourselves to this formulation in the remaining parts of the numerical tests.

5.4.2 Validating Preprocessing Components

We have proposed a preprocessing procedure to eliminate infeasible nodes and arcs (Section 5.3.1) and three types of new valid inequalities (Section 5.3.2). To evaluate the effect of the aforementioned methods, we use the following test set:

- instances a6-72, a8-80, b6-72, and b8-96 introduced above

[1] `https://www.moia.io`
[2] `https://www.holmich-app.de/`

		EB			LAEB			ALAEB		
Inst.	Obj.	Gap	CPU	rGap	Gap	CPU	rGap	Gap	CPU	rGap
a2-16	294.3		0.03	0.8		0.01	0.8		0.03	0.8
a2-20	344.9		0.03	0.0		0.01	0.0		0.02	0.0
a2-24	431.1		0.09	1.9		0.03	2.1		0.06	2.1
a3-18	300.5		0.04	1.8		0.06	1.8		0.04	1.8
a3-24	344.8		0.14	0.5		0.08	2.8		0.3	3.0
a3-30	494.8		0.05	0.0		0.02	0.0		0.08	0.6
a3-36	583.2		0.21	2.5		0.13	2.5		0.16	2.5
a4-16	282.7		0.35	0.7		0.04	0.7		0.05	0.7
a4-24	375.0		0.05	0.0		0.02	0.0		0.03	0.1
a4-32	485.5		0.43	1.8		0.12	2.0		0.24	2.1
a4-40	557.7		0.83	1.7		1.11	1.8		2.04	2.8
a4-48	668.8		0.39	2.0		0.47	2.2		2.66	4.1
a5-40	498.4		0.24	1.1		0.11	1.2		0.32	1.2
a5-50	686.6		1.85	2.6		1.8	2.6		77.42	4.0
a5-60	808.3		0.72	2.0		0.93	2.1		16.71	2.2
a6-48	604.1		0.68	0.7		0.42	0.6		3.85	2.2
a6-60	819.3		9.61	2.4		9.59	2.4		85.33	3.2
a6-72	916.1		16.75	2.5		28.49	2.6		656	3.5
a7-56	724.0		4.85	1.3		1.76	1.6		54.59	3.1
a7-70	875.7		14.99	1.2		4.47	1.3		199	2.2
a7-84	1033.3		20.72	1.9		16.59	2.1		153	4.6
a8-64	747.5		8.35	2.0		5.59	1.9		420	2.9
a8-80	945.8		23.06	2.7		15.84	2.8	1.0	2h	4.5
a8-96	1229.7	1.0	2h	4.1	1.1	2h	4.1	2.9	2h	5.2
Total/Avg.			7304	1.6		7287	1.8		17456	2.5
b2-16	309.4		0.03	0.6		0.07	0.6		0.04	0.6
b2-20	332.7		0.01	0.0		0.04	0.0		0.01	0.0
b2-24	444.7		0.05	1.7		0.05	1.8		0.05	1.8
b3-18	301.6		0.04	2.0		0.05	2.2		0.04	2.2
b3-24	394.5		0.26	2.2		0.19	2.2		0.16	2.2
b3-30	531.4		0.03	0.5		0.03	0.5		0.03	0.5
b3-36	603.8		0.05	0.3		0.04	0.3		0.04	0.3
b4-16	296.9		0.01	1.6		0.03	1.7		0.01	1.7
b4-24	371.4		0.04	0.7		0.04	0.8		0.05	0.8
b4-32	494.9		0.03	0.0		0.03	0.0		0.03	0.0
b4-40	656.6		0.09	0.3		0.07	0.3		0.13	0.3
b4-48	673.8		0.76	0.9		0.42	1.0		2.42	1.1
b5-40	613.7		0.17	0.4		0.09	0.5		0.1	0.4
b5-50	761.4		0.42	1.4		0.2	1.5		0.55	1.6
b5-60	902.0		0.99	1.8		2.16	1.9		7.74	1.9
b6-48	714.8		0.22	0.5		0.12	0.5		0.13	0.5
b6-60	860.0		0.28	0.6		0.17	0.6		0.25	0.6
b6-72	978.5		8.85	1.0		5.64	1.0		26.95	1.0
b7-56	824.0		6.79	0.9		4.01	0.9		22.72	0.9
b7-70	912.6		2.51	2.0		2.77	2.3		9.12	2.5
b7-84	1203.3		2.87	1.2		3.07	1.2		6.3	1.3
b8-64	839.9		2.27	2.3		1.1	2.3		5.56	2.3
b8-80	1036.4		3.49	1.3		1.46	1.9		16.89	1.9
b8-96	1185.6		78.05	1.4		34.43	1.4		287	1.7
Total/Avg.			108	1.1		56	1.1		386	1.2

Table 5.3: Comparing the plain MILP formulations using the a- and b- instances.

	EB				LAEB				ALAEB			
Inst.	Obj.	Gap	CPU	rGap	Obj.	Gap	CPU	rGap	Obj.	Gap	CPU	rGap
a2-16-X	278.2		0.14	1.3	278.2		0.08	1.7	278.2		0.09	1.7
a2-20-X	330.7		0.03	2.0	330.7		0.02	2.0	330.7		0.05	2.4
a2-24-X	389.1		0.26	2.6	389.1		0.20	2.6	389.1		0.62	2.8
a3-18-X	272.7		0.07	2.5	272.7		0.11	3.0	272.7		0.39	7.0
a3-24-X	289.6		0.73	2.2	289.6		0.75	3.5	289.6		1.99	3.5
a3-30-X	452.8		0.34	5.0	452.8		0.28	5.4	452.8		1.32	5.6
a3-36-X	501.0		0.37	1.7	501.0		0.29	1.9	501.0		1.08	1.9
a4-16-X	235.2		0.43	5.9	235.2		0.34	5.9	235.2		1.54	5.9
a4-24-X	359.4		0.37	2.3	359.4		0.12	2.5	359.4		0.71	4.2
a4-32-X	447.3		6.39	3.8	447.3		2.54	3.9	447.3		19.78	4.5
a4-40-X	509.0		38.56	2.9	509.0		21.93	3.0	509.0		85.36	3.2
a4-48-X	620.3		1194	6.4	620.3		660	7.8	622.5	2.5	2h	10.6
a5-40-X	464.0		16.77	3.1	464.0		8.79	3.3	464.0		33.64	4.5
a5-50-X	621.9		285	5.8	621.9		213	5.5	621.9	0.5	2h	6.6
a5-60-X	745.4		281	3.6	745.4		254	4.0	745.4	0.5	2h	6.2
a6-48-X	572.5		7082	5.5	572.5		2675	5.6	572.5	1.1	2h	6.4
a6-60-X	757.9		1993	3.7	757.9		855	4.6	757.9	1.8	2h	5.5
a6-72-X	N/A	N/A	2h	N/A	868.4	3.4	2h	6.8	869.6	4.5	2h	8.5
a7-56-X	663.5		4411	5.0	663.5		5131	5.7	668.9	1.9	2h	6.7
a7-70-X	815.3	0.6	2h	6.2	815.3		1592	6.4	815.3	3.0	2h	8.0
a7-84-X	N/A	N/A	2h	N/A	N/A	N/A	2h	N/A	N/A	N/A	2h	N/A
a8-64-X	702.6	1.9	2h	5.6	701.2	0.9	2h	5.5	701.4	2.8	2h	7.1
a8-80-X	N/A	N/A	2h	N/A	N/A	N/A	2h	N/A	N/A	N/A	2h	N/A
a8-96-X	N/A	N/A	2h	N/A	N/A	N/A	2h	N/A	N/A	N/A	2h	N/A
Total/Avg.			58514	3.9			47418	4.3			86546	5.4
b2-16-X	282.5		0.31	2.2	282.5		0.40	2.4	282.5		0.08	2.4
b2-20-X	323.6		0.02	3.0	323.6		0.02	3.0	323.6		0.04	3.0
b2-24-X	412.3		0.03	0.0	412.3		0.02	0.0	412.3		0.02	0.0
b3-18-X	290.4		0.06	3.0	290.4		0.04	3.0	290.4		0.07	3.0
b3-24-X	363.7		0.12	0.4	363.7		0.06	0.4	363.7		0.28	0.5
b3-30-X	504.3		0.17	2.3	504.3		0.20	2.3	504.3		0.26	2.3
b3-36-X	565.9		0.10	0.9	565.9		0.05	0.9	565.9		0.06	0.9
b4-16-X	289.9		0.03	2.2	289.9		0.02	2.3	289.9		0.02	2.3
b4-24-X	347.0		3.55	4.4	347.0		0.95	5.0	347.0		2.17	5.0
b4-32-X	491.0		0.07	1.0	491.0		0.05	1.0	491.0		0.06	1.0
b4-40-X	628.3		0.21	1.0	628.3		0.25	1.1	628.3		0.37	1.1
b4-48-X	627.4		4.89	1.4	627.4		1.61	1.4	627.4		6.68	2.0
b5-40-X	585.1		5.53	2.7	585.1		1.72	2.8	585.1		24.13	2.7
b5-50-X	708.8		2.31	0.8	708.8		1.12	0.8	708.8		5.33	0.8
b5-60-X	851.9		10.72	1.8	851.9		5.03	1.9	851.9		25.92	2.6
b6-48-X	691.6		2.58	1.7	691.6		0.79	1.7	691.6		1.94	2.2
b6-60-X	841.6		2.32	1.7	841.6		0.63	1.8	841.6		5.14	1.8
b6-72-X	930.3		52.73	1.7	930.3		49.56	1.5	930.3		31.17	1.9
b7-56-X	787.9		142	1.4	787.9		4.32	1.5	787.9		285	1.5
b7-70-X	865.3		19.34	2.6	865.3		14.68	2.5	865.3		132	2.8
b7-84-X	1141.2		3676	3.2	1141.2		2428	3.2	1141.9	1.0	2h	3.6
b8-64-X	818.3		12.07	1.9	818.3		9.16	2.0	818.3		24.68	2.4
b8-80-X	998.3		16.10	2.2	998.3		14.04	2.5	998.3		27.32	2.5
b8-96-X	1137.7	0.3	2h	2.0	1137.7		3425	2.1	1139.5	1.1	2h	2.7
Total/Avg.			11152	1.9			5958	2.0			14973	2.1

Table 5.4: Comparing the plain MILP formulations using the a-X and b-X instances.

- instances Q3n80_2, Q3n80_3, Q3n80_4, Q6n80_5, Q6n100_1, and Q6n100_5 from the artificial instance test set used in Section 4.4
- five real data instances from *Hol mich! App* containing one full day of completed trips each

The characteristics of the instances are summarized in Table 5.5. The instances are selected to provide adequate diversity in terms of share of requests per vehicle, share of number of requests per service duration, and length of time windows. First, we test the efficiency of

Inst.	n	K	T	TW	L_i	Q
a6-72	72	6	720	15	30	3
a8-80	80	8	600	15	30	3
b6-72	72	6	720	15	45	6
b8-96	96	8	720	15	45	6
Q3n80_2	80	11	240	15	1.5 $\bar{t}_i$	3
Q3n80_3	80	11	240	15	1.5 $\bar{t}_i$	3
Q3n80_4	80	10	240	15	1.5 $\bar{t}_i$	3
Q6n80_5	80	12	240	15	$1.5\bar{t}_i$	6
Q6n100_1	100	17	240	15	$1.5\bar{t}_i$	6
Q6n100_5	100	14	240	15	$1.5\bar{t}_i$	6
2021-10-05	85	8	960	25	$\bar{t}_i + \max(10, 0.75\bar{t}_i)$	6
2021-10-07	89	8	960	25	$\bar{t}_i + \max(10, 0.75\bar{t}_i)$	6
2021-10-09	97	12	1080	25	$\bar{t}_i + \max(10, 0.75\bar{t}_i)$	6
2021-10-16	120	12	1080	25	$\bar{t}_i + \max(10, 0.75\bar{t}_i)$	6
2021-10-18	68	2	960	25	$\bar{t}_i + \max(10, 0.75\bar{t}_i)$	6

Table 5.5: Characteristics of the test instances.

graph preprocessing (GP). On average, the number of nodes is reduced by 32%, and the number of arcs by 12%. As the number of nodes and arcs translates directly to the number of variables in the MILP, GP leads to a reduction of 22% of the variables in total. Table 5.6 shows the results for a combination of GP with different subsets of valid inequalities. The first row shows the total solution time for all instances when no preprocessing components are switched on. In the following rows, we test all components and all but one component to be switched on and compare computation times to the first row, where no preprocessing takes place. We achieve an average deviation in computation time ranging from -17 to -52% over all test instances and an average deviation ranging from -64% to -82% over the test instances where computation times are improved. The significant difference between the average deviation over all instances and the average deviation only over the instances with an improvement underlines the heterogeneity of the test set. In the next part of the table, we compare the influence of only one of the valid inequalities switched on (and GP switched on, since otherwise the comparison would be unfair, as IP1 and IP2 cannot be switched on without GP). The improvements in the second part of the table are not as strong as the improvement in the first part, indicating that a combination of multiple valid inequalities and GP is most promising. Taking a look at the average root node gap, we are able to reduce it from 2.9% to 1.9% with multiple combinations of preprocessing steps. This again highlights the tightness of the LAEB formulation, but also underlines

the efficiency of the preprocessing methods. It turns out that for this set of test instances, the best combination is the one given in the bottom row of the table, leading to an average reduction of 58% in CPU time. Although the test instances are very diverse, we cannot exclude that for other sets of test instances another combination prevails, so that we suggest to use a combination of all of the presented preprocessing methods.

After verifying the new presented MILP formulation and preprocessing methods, we next examine the influence of the presented strategies in branch-and-cut nodes.

GP	VS1	VS2	VS3	VS4	CI1	IP1	IP2	# Impr.	Total CPU	Avg. rGap	Avg. Dev.	Avg.* Dev.
0	0	0	0	0	0	0	0	-	407	2.9	-	-
1	1	1	1	1	1	1	1	12	340	1.9	-17 %	-64 %
1	0	1	1	1	1	1	1	12	195	1.9	-52 %	-82 %
1	1	0	1	1	1	1	1	12	196	1.9	-52 %	-79 %
1	1	1	0	1	1	1	1	13	278	1.8	-32 %	-68 %
1	1	1	1	0	1	1	1	13	289	1.9	-29 %	-75 %
1	1	1	1	1	0	1	1	12	306	1.9	-25 %	-65 %
1	1	1	1	1	1	0	1	13	322	2.0	-21 %	-70 %
1	1	0	0	0	0	0	0	11	299	2.6	-27 %	-66 %
1	0	1	0	0	0	0	0	10	345	2.5	-15 %	-46 %
1	0	0	1	0	0	0	0	9	425	2.6	4 %	-59 %
1	0	0	0	1	0	0	0	10	390	2.7	-4 %	-71 %
1	0	0	0	0	1	0	0	11	302	2.6	-26 %	-60 %
1	0	0	0	0	0	1	0	12	168	2.4	-58 %	-74 %
1	0	0	0	0	0	0	1	13	307	2.1	-25 %	-69 %
1	0	0	0	1	1	1	1	14	171	1.9	-58 %	-75 %

Table 5.6: Comparing individual and joint effect of preprocessing components on the test instances. A zero represents a component to be switched off and a one represents it to be switched on.

5.4.3 Testing Branch-and-Cut Algorithm

The lower part of Table 5.2 gives an overview of the methods introduced in Sections 5.3.3–5.3.5 to reduce the size of the graph in branch-and-cut nodes. Following our strategy to evaluate individual components, we solved the LAEB model together with all of the presented preprocessing steps once for each subset of branch-and-cut components on the set of test instances introduced in the previous section. The results are listed in Table 5.7. The first row of Table 5.7 shows the total computational time for the LAEB model and preprocessing steps. In the next rows, all three, two or only one of the presented branching techniques are switched on. Note, that we did not include the case where neither LBsUBs or FA are switched on, since in this case no valid inequalities would be added. We were able to achieve further improvements of about 25% compared to the total computational time after applying preprocessing techniques in about a third of the instances. However, taking the whole set of test instances into account, there is an overall increase ranging from 54% to 66% in CPU. This may be explained by the fact that the overhead caused by the implementation of our cuts in CPLEX is not balanced by the profit from using additional

LBsUBs	FA	FP	Avg. rGap	# Impr.	Total CPU	Avg. Dev.	Avg.* Dev.
0	0	0	1,9	-	340	-	-
1	1	1	1.9	5	583	66 %	-25 %
0	1	1	1.9	6	557	54 %	-20 %
1	0	1	1.9	6	512	56 %	-25 %
1	1	0	1.9	5	580	66 %	-25 %
1	0	0	1.9	6	508	56 %	-25 %
0	1	0	1.9	6	555	54 %	-20 %

Table 5.7: Evaluating the performance of the branch-and-but algorithm. A zero represents a component to be switched off and a one represents it to be switched on.

cuts. Also the average root node gap was very small already such that we might not get low enough in the branch-and-cut tree to fully use the benefit of the presented methods. Nevertheless, the results underline again the heterogeneity of the instances and show that the presented methods are very effective on certain instances.

5.4.4 Results On Benchmark Instances

In this section, we show results on the benchmark instances using the LAEB formulation and the preprocessing methods. The results are shown in Table 5.8. On the a-instances, the total computational time was reduced by 94% compared to using the plain LAEB or the plain EB formulation. On the b-instances, the computational time was reduced by 59% and 79%, respectively (compare Table 5.3). On the a-X instances, the computational time was reduced by 27% compared to using the plain LAEB formulation and by 41% compared to the EB formulation. On the b-X instances, the computational time was reduced by 92% and 96%, respectively. These results prove the efficiency of the new LAEB formulation and the high impact of the proposed preprocessing techniques.

Inst.	Obj.	CPU	rGap	Inst.	Obj.	CPU	rGap	Inst.	Obj.	Gap	CPU	rGap	Inst.	Obj.	CPU	rGap
a2-16	294.3	0.10	0.0	b2-16	309.4	0.17	0.6	a2-16-X	278.2		0.07	0.0	b2-16-X	282.5	0.57	1.3
a2-20	344.9	0.02	0.0	b2-20	332.7	0.02	0.0	a2-20-X	330.7		0.03	0.0	b2-20-X	323.6	0.02	0.8
a2-24	431.1	0.05	0.8	b2-24	444.7	0.07	1.0	a2-24-X	389.1		0.13	1.4	b2-24-X	412.3	0.03	0.0
a3-18	300.5	0.02	0.3	b3-18	301.6	0.03	0.9	a3-18-X	272.7		0.11	2.8	b3-18-X	290.4	0.05	1.3
a3-24	344.8	0.07	1.6	b3-24	394.5	0.12	1.9	a3-24-X	289.6		0.74	3.3	b3-24-X	363.7	0.08	0.4
a3-30	494.8	0.06	0.0	b3-30	531.4	0.03	0.0	a3-30-X	452.8		0.27	2.2	b3-30-X	504.3	0.16	1.6
a3-36	583.2	0.17	2.5	b3-36	603.8	0.05	0.1	a3-36-X	501.0		0.48	0.6	b3-36-X	565.9	0.07	0.0
a4-16	282.7	0.04	0.7	b4-16	296.9	0.02	0.6	a4-16-X	235.2		0.50	5.4	b4-16-X	289.9	0.02	1.7
a4-24	375.0	0.04	0.0	b4-24	371.4	0.04	0.1	a4-24-X	359.4		0.15	1.5	b4-24-X	347.0	0.47	2.1
a4-32	485.5	0.12	1.1	b4-32	494.9	0.04	0.0	a4-32-X	447.3		1.78	3.4	b4-32-X	491.0	0.08	0.7
a4-40	557.7	0.28	1.2	b4-40	656.6	0.09	0.3	a4-40-X	509.0		0.95	2.1	b4-40-X	628.3	0.28	0.1
a4-48	668.8	0.27	1.7	b4-48	673.8	0.65	0.7	a4-48-X	620.3		56.36	5.5	b4-48-X	627.4	1.46	0.9
a5-40	498.4	0.16	0.6	b5-40	613.7	0.12	0.4	a5-40-X	464.0		6.12	2.9	b5-40-X	585.1	2.65	1.9
a5-50	686.6	1.51	2.0	b5-50	761.4	0.21	1.0	a5-50-X	621.9		73.02	4.0	b5-50-X	708.8	0.53	0.6
a5-60	808.3	0.45	1.4	b5-60	902.0	0.86	1.6	a5-60-X	745.4		39.78	3.3	b5-60-X	851.9	1.66	1.3
a6-48	604.1	0.38	0.4	b6-48	714.8	0.18	0.3	a6-48-X	572.5		935	4.7	b6-48-X	691.6	0.53	0.4
a6-60	819.3	2.14	1.6	b6-60	860.0	0.24	0.1	a6-60-X	757.9		332	3.2	b6-60-X	841.6	1.13	1.3
a6-72	916.1	11.34	2.2	b6-72	978.5	3.95	0.9	a6-72-X	869.6	2.1	2h	5.6	b6-72-X	930.3	7.16	1.4
a7-56	724.0	2.32	1.2	b7-56	824.0	1.46	0.8	a7-56-X	663.5		551	4.1	b7-56-X	787.9	5.06	0.8
a7-70	875.7	2.11	0.9	b7-70	912.6	0.65	1.5	a7-70-X	815.3		711	4.7	b7-70-X	865.3	6.11	1.6
a7-84	1033.3	11.13	1.6	b7-84	1203.3	1.50	0.8	a7-84-X	950.6	1.4	2h	6.2	b7-84-X	1141.2	89.10	2.1
a8-64	747.5	2.29	1.1	b8-64	839.9	0.70	1.6	a8-64-X	701.2		3042	4.1	b8-64-X	818.3	2.38	1.8
a8-80	945.8	7.58	1.9	b8-80	1036.4	0.64	0.8	a8-80-X	880.3	2.7	2h	6.4	b8-80-X	998.3	6.00	1.7
a8-96	1229.7	425	2.8	b8-96	1185.6	11.13	1.1	a8-96-X	N/A	N/A	2h	N/A	b8-96-X	1137.7	340	2.3
Total		467	1.2			22	0.7				34554	3.4			466	1.2

Table 5.8: Results for the benchmark instances solved using the LAEB formulation and preprocessing.

Although computation times on different computers are only comparable to a limited extent, our results are highly competitive with the branch-and-cut-and-price algorithm proposed in Gschwind and Irnich (2015): While some of the a-instances were solved faster by our approach and some of them were solved faster by their branch-and-cut algorithm, the solution times on the b-instances are smaller by about 50 orders of magnitude.

Inst.	Obj.	Gap	CPU	Inst.	Obj.	CPU
a2-16-X	278.2		0.13	b2-16-X	282.5	0.12
a2-20-X	330.7		0.03	b2-20-X	323.6	0.06
a2-24-X	389.1		0.09	b2-24-X	412.3	0.03
a3-18-X	272.7		0.09	b3-18-X	290.4	0.08
a3-24-X	289.6		0.34	b3-24-X	363.7	0.12
a3-30-X	452.8		0.14	b3-30-X	504.3	0.14
a3-36-X	501.0		0.20	b3-36-X	565.9	0.07
a4-16-X	235.2		0.20	b4-16-X	289.9	0.03
a4-24-X	359.4		0.15	b4-24-X	347.0	0.27
a4-32-X	447.3		0.76	b4-32-X	491.0	0.10
a4-40-X	509.0		1.75	b4-40-X	628.3	0.21
a4-48-X	620.3		15.88	b4-48-X	627.4	1.18
a5-40-X	464.0		2.17	b5-40-X	585.1	1.39
a5-50-X	621.9		15.94	b5-50-X	708.8	0.48
a5-60-X	745.4		16.68	b5-60-X	851.9	1.72
a6-48-X	572.5		101	b6-48-X	691.6	0.54
a6-60-X	757.9		100	b6-60-X	841.6	1.04
a6-72-X	868.4		4368	b6-72-X	930.3	6.08
a7-56-X	663.5		102	b7-56-X	787.9	6.50
a7-70-X	815.3		100	b7-70-X	865.3	2.77
a7-84-X	950.6		6396	b7-84-X	1141.2	50.73
a8-64-X	701.2		191	b8-64-X	818.3	3.04
a8-80-X	880.3		4579	b8-80-X	998.3	2.73
a8-96-X	1118.5	2.3	2h	b8-96-X	1137.7	147
Total			23196			226

Table 5.9: Results for the a-X and b-X benchmark instances on 12 threads.

However, our solution times cannot compete with the speed of the branch-and-cut algorithm developed in Rist and Forbes (2021) but could still prove useful in practice, as the implementation of an MILP formulation and preprocessing methods is very fast and easy compared to the implementation of a branch-and-cut algorithm. Moreover, the results show that our methods are able to solve instances of substantial size within seconds. Thus, they can be used in a practical dynamic setting of a ridepooling provider within a rolling horizon approach like it is shown for the EB formulation in the next chapter. This is especially true, as the LAEB formulation improves the EB formulation, and our results show that the model performs even better for instances where groups are transported (b and b-X instances) which is an important case for ridepooling providers. The results in this section are still for one thread only. Since we do not use user-defined cuts if we omit the methods presented in Sections 5.3.3–5.3.5, we do not need to limit the number of threads used by CPLEX or to restrict the search method to traditional branch-and-cut. Using 12 threads the whole set of a- and b-instances was solved in 62 seconds. The corre-

sponding results for the a-X and b-X instances show that medium-sized and some of the large instances of the harder benchmark set can be solved within a few seconds (compare Table 5.9).

5.5 Summary

The new presented MILP formulations combine the existing state-of-the-art formulations, the LB and the EB formulation of the DARP. As shown in Theorem 5.4, the LAEB formulation is tighter than the LB formulation. Computational tests on large benchmark instances show that, using the LAEB formulation, computational times can be reduced by 19% (a-X instances) and 47% (b-X instances), respectively, compared to the EB formulation. Additionally, graph preprocessing and the introduction of new valid inequalities which eliminate subtours, infeasible events, and infeasible paths strongly further improve computational times by 27% (a-X) instances and 92% (b-X), respectively. An average root node gap of 1.6% proves that our formulation and preprocessing methods generate a very tight MILP model. Our methods are useful for application in practice, as the implementation is fast and easy and even medium-sized instances of the harder benchmark set can be solved within a few seconds (compare Table 5.9).

While the literature mostly focuses on the static DARP, on-demand ridepooling services need to include customer requests on time when they arise. In Chapter 6 we show that the dynamic DARP can be solved using a rolling-horizon algorithm in which the EB formulation is updated and solved whenever new requests arrive. Since the LAEB formulation improves on the EB formulation, this strategy could be adapted using the LAEB formulation as well.

6 Rolling-Horizon Event-Based MILP for the Dynamic Dial-a-Ride Problem

In many ridepooling applications transportation requests arrive throughout the day and have to be answered and integrated into the existing (and operated) vehicle routing. To solve this dynamic dial-a-ride problem we present a rolling-horizon algorithm that dynamically updates the current solution by solving the EB formulation. As highlighted in the literature review on solution approaches to the dynamic DARP in Section 3.1.2, the standard approach to solve the dynamic DARP is to apply a two-phase algorithm consisting of an insertion heuristic and a re-optimization phase. In this chapter, we suggest a more global perspective and aim at the iterative computation of exact optimal solutions that satisfy all feasibility constraints and that respect previous routing decisions. Only when this global optimization exceeds a prespecified time limit of 30 seconds without proving global optimality, the computed schedule is re-optimized in the following iteration. We present computational experiments for real-world data from a ridepooling service in the city of Wuppertal in Germany with up to 500 requests. In all tested instances the average response time was never more than 2.9 seconds. Moreover, a re-optimization was necessary in no more than 0.5% of the iterations. In all other iterations the algorithm returned a globally optimal solution w.r.t. the current situation, which can generally not be guaranteed by common two-phase heuristics.

The topic of this chapter is the deterministic, dynamic and homogeneous DARP. We allow transport requests to be rejected. In addition to the minimization of routing costs and regret, the maximization of the number of accepted requests is one optimization goal. The remainder of the chapter is structured as follows. A description of the dynamic characteristics and an outline of the solution strategy applied in this chapter is given in Section 6.1. In Section 6.2 the concept of the event-based graph is transferred to the dynamic DARP by associating a dynamic event-based graph with each subproblem of the DARP. The corresponding MILP model is introduced in Section 6.3. Finally, the procedure of updating the event-based graph and solving the MILP model, resulting in a decision on the acceptance of new requests, is outlined in the framework of a rolling-horizon algorithm in Section 6.4. To validate our approach, computational results on two real-world instances are presented in Section 6.5. A short summary of our results is given in Section 6.6. The contents of this chapter are based on Gaul et al. (2021).

Note that, the algorithm proposed in this chapter can be adapted to iteratively solve the LAEB formulation instead of the EB formulation. Also, the preprocessing techniques proposed in the previous chapter can be used before the MILPs are passed to the solver. However, the MILPs solved in every iteration of the rolling-horizon algorithm are small- to medium-sized as usually only a few new requests arrive in every iteration and part of the routing decisions have been fixed already (i.e., have been realized) at the time new requests arrive. For small- to medium-sized instances the performance of the EB and the

LAEB formulation is similar. Also the effect of preprocessing techniques is more significant for larger instances. Moreover, the overhead caused by the implementation of new valid inequalities in every iteration of the algorithm is likely to outweigh the savings in computational time caused by the inequalities. For these reasons we use the EB formulation as underlying MILP for the DARP in the rolling-horizon algorithm in the following.

6.1 Solution Strategy

In this chapter, we consider a dynamic DARP in which a finite set of transport requests submitted by users have to be either accepted and scheduled or rejected. We assume that the transport requests arrive throughout the day. We consider discrete points in time $\tau_1 \leq \cdots \leq \tau_n$ such that request i becomes known at time $\tau_i - \Delta$, where $\Delta \geq 0$ is the predefined time-limit for the update of the current solution (we set $\Delta = 0.5$ minutes in our numerical experiments). Every user that is accepted is communicated a pickup time Γ_i. This time may not be postponed by more than γ minutes.

Due to the dynamic nature of the problem, at any time τ only the requests that have arrived up to time τ are known. In addition, some requests might have been rejected and some of the accepted requests might already have been delivered to their delivery location at time τ. Therefore, at any time τ, only a subproblem DARP(τ) related to the *active requests* $\mathcal{A}(\tau)$ at time τ needs to be considered which comprises all requests that are known but neither rejected nor dropped off w.r.t. the current solution $\boldsymbol{x}(\tau)$. To distinguish between these different types of requests at a given time τ, let

- $\mathcal{N}(\tau)$ denote the subset of *new requests* that were revealed at time $\tau - \Delta$,
- $\mathcal{S}(\tau)$ denote the subset of *scheduled requests*, i.e., requests that have been accepted but have not been picked up up to time τ,
- $\mathcal{P}(\tau)$ denote the subset of *picked up requests* that have not been dropped off up to time τ,
- $\mathcal{D}(\tau)$ denote the subset of *dropped off requests* up to time τ and
- $\mathcal{R}(\tau)$ denote the subset of *rejected requests* up to time τ.

Then $\mathcal{A}(\tau) = \mathcal{N}(\tau) \cup \mathcal{S}(\tau) \cup \mathcal{P}(\tau)$ while $\mathcal{D}(\tau), \mathcal{R}(\tau) \not\subseteq \mathcal{A}(\tau)$. Note that, the sets $\mathcal{S}(\tau)$, $\mathcal{P}(\tau)$, $\mathcal{D}(\tau)$, $\mathcal{R}(\tau)$ do in fact not only depend on the time τ but also on the solutions determined in previous time steps. Each feasible solution $\boldsymbol{x}(\tau)$ to a subproblem DARP(τ) consists of at most K vehicle routes which start and end at the depot. If a user is served by a vehicle, the user has to be picked up and dropped off by the same vehicle. On the other hand, a rejected user may not be picked up or dropped off by any of the vehicles.

A solution to the dynamic DARP is a strategy that, every time one or more new requests are revealed, modifies the solution of the last subproblem so that each of the new requests is either assigned to a vehicle route or rejected. In the course of assigning new requests to already existing vehicle routes, old requests, if not yet picked up or dropped off, might have to be reassigned. However, every request, once accepted, has to be served and every request, once rejected, cannot be served by any vehicle in the following subproblems.

The solution approach we propose in this chapter is based on the EB MILP formulation for the static DARP presented in Section 4.2.3, for which we showed that it efficiently generates exact solutions to small to medium sized static benchmark problems in a few seconds. The idea of a solution strategy for the dynamic DARP is as follows: 1. An initial solution is obtained by solving the EB MILP for the requests that are revealed at time $\tau_1 - \Delta$, which is interpreted as the time when the routes are initialized. 2. When new requests arrive at time $\tau_i - \Delta$, $i \geq 2$, the respective users are notified within 30 seconds whether they have been accepted or rejected. Therefore, the vehicle routes up to time τ_i are frozen and the set of active requests $\mathcal{A}(\tau_i)$ is updated. The underlying event-based graph is modified by removing all nodes and arcs corresponding to rejected requests and partially removing nodes and arcs corresponding to dropped off or picked up users. Nodes and arcs for the new requests are added to the event-based graph. Then the MILP is updated and solved again.

6.2 Event-Based Graph Model for a Rolling-Horizon

In order to extend the concept of the event-based graph to the dynamic DARP, we assume that solutions are extended iteratively whenever new requests arrive and introduce a *dynamic event-based graph* $G(\tau) = (V(\tau), A(\tau))$ for the subproblem DARP(τ) at time τ. When new requests are revealed at time $\tau_i - \Delta$, $i \in \{1, \ldots, n\}$, then the event-based graph $G(\tau_i)$ is updated based on the event-based graph $G(\tau_{i-1})$ and the associated solution $\boldsymbol{x}(\tau_{i-1})$ of the last subproblem: Nodes and arcs corresponding to rejected, dropped off and picked up users are (partially) removed from the graph while nodes and arcs corresponding to new requests are added.

The node set $V(\tau)$ represents events which are feasible w.r.t. the vehicle capacity Q and also reflect pairwise feasibilities w.r.t. time window and ride time constraints of the requests that have been revealed up to time τ. The set of nodes representing an event in which a user $i \in \mathcal{A}(\tau) \setminus \mathcal{P}(\tau)$ is picked up is called the set of *pickup nodes up to time* τ and is given by

$$V_{i^+}(\tau) \coloneqq \Bigg\{ (v_1, v_2, \ldots, v_Q) \colon v_1 = i^+,\ v_j \in \mathcal{A}(\tau) \cup \{0\} \setminus \{i\},\ f^1_{i,v_j} + f^2_{i,v_j} \geq 1 \\ \forall j \in \{2, \ldots, Q\},\ \big(v_j > v_{j+1} \vee v_{j+1} = 0\big)\, \forall j \in \{2, \ldots, Q-1\},\ \sum_{j=1}^{Q} q_{v_j} \leq Q \Bigg\}. \tag{6.1}$$

Similarly, the set of *delivery nodes up to time* τ corresponds to events where a user $i \in \mathcal{A}(\tau)$

is dropped off and is given by

$$V_{i^-}(\tau) \coloneqq \Bigg\{ (v_1, v_2, \ldots, v_Q) \colon v_1 = i^-,\ v_j \in \mathcal{A}(\tau) \cup \{0\} \setminus \{i\},\ f^1_{v_j,i} + f^2_{i,v_j} \geq 1 \\ \forall j \in \{2, \ldots, Q\},\ \big(v_j > v_{j+1} \vee v_{j+1} = 0\big)\, \forall j \in \{2, \ldots, Q-1\},\ \sum_{j=1}^{Q} q_{v_j} \leq Q \Bigg\}. \tag{6.2}$$

Note that, just like in the case of the event-based graph, from the set of all pickup and delivery nodes associated with an accepted user, exactly one pickup and one delivery node are contained in the dicycle flow representing the vehicle tour to which the user is assigned in the current solution. The set of nodes $V_{\mathcal{A}(\tau)}$ corresponding to the set of active requests $\mathcal{A}(\tau)$ is then given by

$$V_{\mathcal{A}(\tau)} = V_0 \cup \bigcup_{i \in \mathcal{A}(\tau) \setminus \mathcal{P}(\tau)} V_{i^+}(\tau) \cup \bigcup_{i \in \mathcal{A}(\tau)} V_{i^-}(\tau).$$

Simply put, $V_{\mathcal{A}(\tau)}$ represents the set of nodes that are available at time τ but have not been reached by any vehicle (yet). This set does not include nodes (and hence events) corresponding to users that have been rejected or dropped off up to time τ since $\mathcal{D}(\tau), \mathcal{R}(\tau) \not\subseteq \mathcal{A}(\tau)$. Moreover, pickup nodes corresponding to users $\mathcal{P}(\tau)$ are not considered since they have already been reached by a vehicle, where the user has been picked up. Nodes where a pickup or delivery has already been realized up to time τ are referred to as *realized nodes*. As a consequence, each request that is known at time $\tau - \Delta$ falls in one of the following three categories:

- If $i \in \mathcal{N}(\tau) \cup \mathcal{S}(\tau) \cup \mathcal{R}(\tau)$, then no associated node (event) is realized since request i was either rejected or the scheduled pickup and delivery times are larger than τ.
- If $i \in \mathcal{P}(\tau)$, then exactly one associated node (event) is a realized node, which is a pickup node.
- If $i \in \mathcal{D}(\tau)$, then exactly one associated pickup node (event) and one associated delivery node (event) is realized.

Let $V^{\text{realized}}_{\mathcal{D}(\tau)}$ denote the set of all realized pickup and delivery nodes for each user $i \in \mathcal{D}(\tau)$ and let $V^{\text{realized}}_{\mathcal{P}(\tau)}$ denote the set of all realized pickup nodes associated with each user $i \in \mathcal{P}(\tau)$. Then the node set $V(\tau)$ is defined as

$$V(\tau) \coloneqq V_{\mathcal{A}(\tau)} \cup V^{\text{realized}}_{\mathcal{D}(\tau)} \cup V^{\text{realized}}_{\mathcal{P}(\tau)}.$$

Hence, for a user $i \in \mathcal{D}(\tau)$ that has been dropped off up to time τ only the unique realized pickup and delivery nodes are contained in $V(\tau)$, i.e., $V_{i^+}(\tau) \coloneqq \{v \in V^{\text{realized}}_{\mathcal{D}(\tau)} \colon v_1 = i^+\}$ and $V_{i^-}(\tau) \coloneqq \{v \in V^{\text{realized}}_{\mathcal{D}(\tau)} \colon v_1 = i^-\}$. Analogously, for a picked up user $i \in \mathcal{P}(\tau)$ only the unique realized pickup node is contained in $V(\tau)$, i.e., $V_{i^+}(\tau) \coloneqq \{v \in V^{\text{realized}}_{\mathcal{P}(\tau)} \colon v_1 = i^+\}$.

Similar to the node set $V(\tau)$, the arc set $A(\tau)$ of $G(\tau)$ has to reflect the fact that some routing decisions have already been fixed up to time τ in the rolling-horizon framework. This motivates the introduction of the concept of *realized arcs*: Each realized pickup and delivery node $v \in V^{\text{realized}}_{\mathcal{D}(\tau)} \cup V^{\text{realized}}_{\mathcal{P}(\tau)}$ is contained in a dicycle flow representing a vehicle's tour. The ingoing arc of a realized node, which is part of this dicycle flow, is referred to as *realized arc*. We denote the set of realized arcs by $A^{\text{realized}}(\tau)$. Let $v \in V^{\text{realized}}_{\mathcal{D}(\tau)} \cup V^{\text{realized}}_{\mathcal{P}(\tau)}$ be chosen such that there is no arc $a = (v, w) \in A^{\text{realized}}(\tau)$. Thus, v is the last realized node in the corresponding dicycle flow at time τ. Such nodes indicate the last realized stop on the current tour, from which on the solution may be modified if this is advantageous given the newly revealed requests. We denote the set of "last realized nodes" as $V^{\text{l-realized}}(\tau)$. Then, the arc set $A(\tau)$ is composed of seven subsets that will be further specified below:

$$A(\tau) = \bigcup_{k=1}^{6} A_k(\tau) \cup A^{\text{realized}}(\tau).$$

As in the static case, c.f. Section 4.1, $A(\tau)$ represents the set of transits from one event node to another. Let i and j be requests that have been revealed up to time $\tau - \Delta$. Then the six subsets $A_k(\tau)$, $k = 1, \ldots, 6$ are defined as follows:

- The first set $A_1(\tau)$ describes the transit from a pickup node from a set $V_{i^+}(\tau)$ to a delivery node from a set $V_{j^-}(\tau)$:

$$A_1(\tau) \coloneqq \Big\{ \big((i^+, v_2, \ldots, v_Q), (j^-, w_2, \ldots, w_Q) \big) \in \big(V_{\mathcal{A}(\tau)} \cup V^{\text{l-realized}}(\tau) \big) \times V_{\mathcal{A}(\tau)} : \\ \{j, w_2, \ldots, w_Q\} = \{i, v_2, \ldots, v_Q\} \Big\}.$$

- The transit from a pickup node from a set $V_{i^+}(\tau)$ to another pickup node from a set $V_{j^+}(\tau)$ with $j \neq i$ is represented by the following set:

$$A_2(\tau) \coloneqq \Big\{ \big((i^+, v_2, \ldots, v_{Q-1}, 0), (j^+, w_2, \ldots, w_Q) \big) \in \big(V_{\mathcal{A}(\tau)} \cup V^{\text{l-realized}}(\tau) \big) \times \\ V_{\mathcal{A}(\tau)} : \{i, v_2, \ldots, v_{Q-1}\} = \{w_2, \ldots, w_Q\} \Big\}.$$

- $A_3(\tau)$ is comprised of arcs which describe the transit from a delivery node in a set $V_{i^-}(\tau)$ to a pickup node in a set $V_{j^+}(\tau)$, $j \neq i$:

$$A_3(\tau) \coloneqq \Big\{ \big((i^-, v_2, \ldots, v_Q), (j^+, v_2, \ldots, v_Q) \big) \in \big(V_{\mathcal{A}(\tau)} \cup V^{\text{l-realized}}(\tau) \big) \times V_{\mathcal{A}(\tau)} : i \neq j \Big\}.$$

- The transit from a delivery node from a set $V_{i^-}(\tau)$ to another delivery node from a set $V_{j^-}(\tau)$, $j \neq i$, is represented by:

$$A_4(\tau) \coloneqq \Big\{ \big((i^-, v_2, \ldots, v_Q), (j^-, w_2, \ldots, w_{Q-1}, 0) \big) \in \big(V_{\mathcal{A}(\tau)} \cup V^{\text{l-realized}}(\tau) \big) \times \\ V_{\mathcal{A}(\tau)} : \{v_2, \ldots, v_Q\} = \{j, w_2, \ldots, w_{Q-1}\} \Big\}.$$

- A dicycle in $G(\tau)$ representing a vehicle tour always contains an arc describing the transit from the depot to a pickup node in a set $V_{i^+}(\tau)$, as well as an arc describing the transit from a delivery node from a set $V_{j^-}(\tau)$ to the depot. The following two sets describe these transitions:

$$A_5(\tau) \coloneqq \Big\{\big((0,\dots,0),\big(i^+,0,\dots,0\big)\big) \in V_0 \times V_{\mathcal{A}(\tau)}\Big\},$$

$$A_6(\tau) \coloneqq \Big\{\big(\big(j^-,0,\dots,0\big),(0,\dots,0)\big) \in \big(V_{\mathcal{A}(\tau)} \cup V^{\text{l-realized}}(\tau)\big) \times V_0\Big\}.$$

Example 6.1. *We give an example of the changes in the event-based graph for three requests and one vehicle with capacity $Q = 3$. Let $R = \{1,2,3\}$. The request data is as follows:*

i	q_i	τ_i	$[e_{i^+}, \ell_{i^+}]$	$[e_{i^-}, \ell_{i^-}]$
1	*1*	*5*	$[10,25]$	$[15,40]$
2	*2*	*15*	$[20,35]$	$[30,50]$
3	*2*	*45*	$[50,65]$	$[55,80]$

For the sake of clarity, we assume that all requests are accepted. Furthermore, we assume that the remaining parameters (e.g., travel times) allow all variants of routing described in the following, but are omitted in this example.

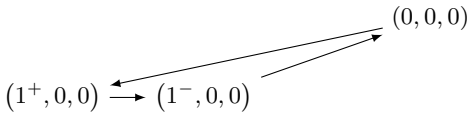
When the first request is revealed, we have $\mathcal{A}(\tau_1) = \mathcal{N}(\tau_1) = \{1\}$ and $\mathcal{S}(\tau_1) = \mathcal{P}(\tau_1) = \mathcal{D}(\tau_1) = \emptyset$. The initial graph $G(\tau_1)$ is depicted in Figure 6.1(a). We assume that by the time request 2 is revealed, user 1 has not been picked up yet, i.e., $\mathcal{N}(\tau_2) = \{2\}$, $\mathcal{S}(\tau_2) = \{1\}$, $\mathcal{A}(\tau_2) = \{1,2\}$ and $\mathcal{P}(\tau_2) = \mathcal{D}(\tau_2) = \emptyset$. Therefore, we only have to add additional nodes and arcs induced by request 2 as illustrated in $G(\tau_2)$ in Figure 6.1(b). According to the time windows, by the time request 3 is revealed user 1 must have been dropped off and user 2 must have been picked up. We assume that user 2 has not been dropped off yet and that the vehicle tour induced by the current solution is given by the dicycle

$$\big(0,0,0\big) \to \big(1^+,0,0\big) \to \big(2^+,1,0\big) \to \big(1^-,2,0\big) \to \big(2^-,0,0\big) \to \big(0,0,0\big).$$

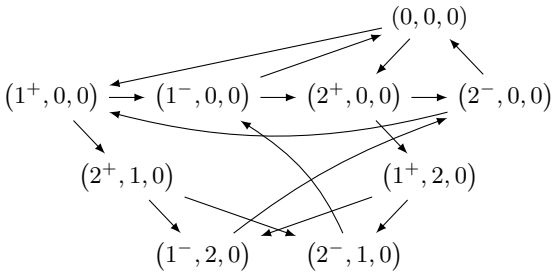
Hence, $\mathcal{N}(\tau_3) = 3$, $\mathcal{A}(\tau_3) = \{2,3\}$, $\mathcal{P}(\tau_3) = \{2\}$, $\mathcal{D}(\tau_3) = \{1\}$ and $\mathcal{S}(\tau_3) = \emptyset$. The corresponding realized nodes are $V^{realized}_{\mathcal{P}(\tau_3)} = \{(2^+,1,0)\}$ and $V^{realized}_{\mathcal{D}(\tau_3)} = \{(1^+,0,0),(1^-,2,0)\}$. The set of realized arcs is

$$A^{realized}(\tau_3) = \{((0,0,0),(1^+,0,0)),((1^+,0,0),(2^+,1,0)),((2^+,1,0),(1^-,2,0))\}$$

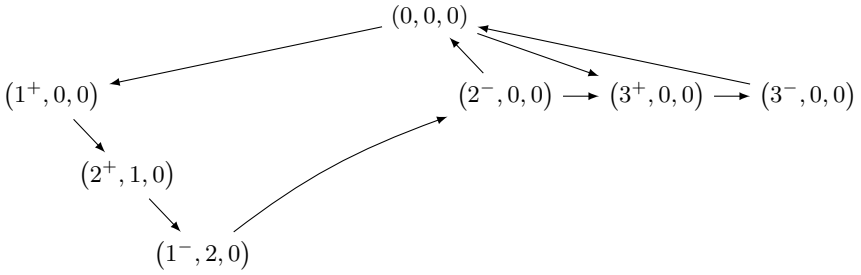
and $V^{l\text{-}realized}(\tau_3) = \{(1^-,2,0)\}$. The update of the event-based graph to obtain $G(\tau_3)$ is illustrated in Figure 6.1(c). Note that, there are no nodes $v \in V(\tau_3)$ that simultaneously contain users 1 (i.e., 1^+ or 1^-) and 3 (i.e., 3^+ or 3^-) as $1 \notin \mathcal{A}(\tau_3)$, which means that according to equations (6.1) and (6.2) there are no shared nodes. Similarly, the seats requested by users 2 and 3 combined exceed the vehicle capacity of three.



(a) Graph $G(\tau_1)$.



(b) Graph $G(\tau_2)$.



(c) Graph $G(\tau_3)$.

Figure 6.1: Evolution of the dynamic event-based graph for an instance with three requests.

6.3 Event-Based MILP Model for a Rolling-Horizon

Based on the event-based graph model we update and solve an MILP in a rolling-horizon strategy whenever new requests arrive, that is, at times $\tau = \tau_j$ for $j = 1, \ldots, n$. Every subproblem DARP(τ) can be modeled as a variant of a minimum cost flow problem with additional constraints in the dynamic event-based graph $G(\tau) = (V(\tau), A(\tau))$.

For the MILP formulation of DARP(τ) we use the following additional parameters and variables:

Since every node in the dynamic event-based graph $G(\tau) = (V(\tau), A(\tau))$ corresponds to a uniquely determined geographical location, we can associate routing costs $c_a \geq 0$ and a travel times $t_a \geq 0$ with the respective arcs $a \in A(\tau)$ in $G(\tau)$. Let $\delta^{\text{in}}(v, \tau)$ and $\delta^{\text{out}}(v, \tau)$ denote the set of ingoing and outgoing arcs of v at time τ, respectively. A solution of DARP(τ) is denoted by $\boldsymbol{x}(\tau)$ and is composed of the following variables: The binary variables x_a with $a \in A(\tau)$ are equal to one if and only if arc $a \in A(\tau)$ is used by a vehicle. A feasible tour of a vehicle is then represented by a dicycle C in the dynamic event-based graph $G(\tau)$ such that $x_a = 1$ for all $a \in A(C)$, where $A(C)$ is the arc set of C. If a vehicle has reached the last delivery location in the dicycle representing its route, it will wait at its current location for new requests until it has to start its journey back to the depot to arrive there before the end of service ℓ_0. Since requests might be rejected, we introduce a binary variable p_i for each $i \in \mathcal{A}(\tau) \setminus \mathcal{P}(\tau)$ with $p_i = 1$ indicating that request i is accepted. To model the beginning of service at a node $v \in V(\tau)$, i.e., the time at which a vehicle arrives at the location represented by v to pickup or delivery passengers, we use continuous variables B_v. The continuous variables d_i, $i \in \mathcal{A}(\tau)$ measure a user's regret compared to his or her earliest delivery time.

The parameters x_a^{old} and B_v^{old} are used to store the values of the variables x_a and B_v from the previous iteration in the rolling-horizon framework. Once a vehicle has departed from a location, we cannot divert it from its next destination (as this brings technical difficulties related to the calculation of distances, see Berbeglia et al., 2010). Also, if an arc has been realized up to time τ, it has to be included in a dicycle flow in all later subproblems. Therefore, if $\tau > \tau_1$ then all partial routes up to time τ and hence all variables x_a corresponding to the set

$$A^{\text{fixed}}(\tau) := \{(v, w) \in A(\tau) \colon x_{(v,w)}^{\text{old}} = 1, \tau \geq B_w^{\text{old}} - t_{(v,w)}\}$$

are fixed in the MILP corresponding to the current subproblem DARP(τ). The set of realized arcs $A^{\text{realized}}(\tau)$ is a subset of the set of fixed arcs $A^{\text{fixed}}(\tau)$. We set $A^{\text{fixed}}(\tau_1) = \emptyset$. Furthermore, let $A^{\text{new}}(\tau)$ be the set of all arcs that have not been contained in the graph corresponding to the previous subproblem. We have $A^{\text{new}}(\tau_1) = A(\tau_1)$.

For the remainder of this section, let $j \in \{1, \ldots, n\}$ be arbitrary but fixed. To prepare the MILP formulation of DARP(τ_j), we define a set of travel time constraints ($C_{v,w}(\tau_j)$) for all $(v, w) \in A^{\text{new}}(\tau_j) \setminus \delta^{\text{out}}((0, \ldots, 0), \tau_j)$:

$$B_w \geq \max\{B_v, \tau_j\} + s_{v_1} + t_{(v,w)} - M_{v,w}(\tau_j) \cdot (1 - x_{(v,w)}), \qquad (C_{v,w}(\tau_j))$$

$$\text{where } M_{v,w}(\tau_j) = \begin{cases} \ell_{v_1} - e_{w_1} + s_{v_1} + t_{(v,w)} & \text{if } B_v \geq \tau_j, \\ \tau_j - e_{w_1} + s_{v_1} + t_{(v,w)} & \text{otherwise,} \end{cases}$$

is a sufficiently large constant. The constraints $(C_{v,w}(\tau_j))$ guarantee that for all arcs $(v,w) \in A^{\text{new}}(\tau_j) \setminus \delta^{\text{out}}((0,\dots,0),\tau_j)$ the beginning of service at a node w is greater than or equal to the earliest departure time at a preceding node v plus the time needed to travel from node v to node w. If $(v,w) \in A^{\text{new}}(\tau_j) \setminus \delta^{\text{out}}((0,\dots,0),\tau_j)$, then the arc (v,w) is related to a new request that has been revealed at time $\tau_j - \Delta$. This implies that travel from v to w can start no earlier than $\max\{B_v, \tau_j\} + s_{v_1}$. Note that, in this case constraint $(C_{v,w}(\tau_j))$ can be linearized by rewriting it using two constraints where $\max\{B_v, \tau_j\}$ is once replaced by B_v and once by τ_j.

We are now ready to formulate the dynamic event-based MILP(τ_j) for each subproblem DARP(τ_j).

Event-Based MILP(τ_j) for a Rolling-Horizon

$$\min \ \omega_1 \sum_{a \in A(\tau_j)} c_a\, x_a + \omega_2 \sum_{i \in \mathcal{A}(\tau_j) \setminus \mathcal{P}(\tau_j)} (1 - p_i) + \omega_3 \sum_{i \in \mathcal{A}(\tau_j)} d_i, \tag{6.3a}$$

$$\text{s. t.} \quad \sum_{a \in \delta^{\text{in}}(v,\tau_j)} x_a - \sum_{a \in \delta^{\text{out}}(v,\tau_j)} x_a = 0 \qquad \forall v \in V(\tau_j), \tag{6.3b}$$

$$\sum_{\substack{a \in \delta^{\text{in}}(v,\tau_j) \\ v \in V_{i^+}}} x_a = p_i \qquad \forall i \in \mathcal{A}(\tau_j) \setminus \mathcal{P}(\tau_j), \tag{6.3c}$$

$$\sum_{a \in \delta^{\text{out}}((0,\dots,0),\tau_j)} x_a \le K, \tag{6.3d}$$

$$e_0 \le B_{(0,\dots,0)} \le \ell_0, \tag{6.3e}$$

$$e_{i^+} + (\ell_{i^+} - e_{i^+}) \left(1 - \sum_{a \in \delta^{\text{in}}(v,\tau_j)} x_a\right) \le B_v \le \ell_{i^+} \ \ \forall i \in \mathcal{A}(\tau_j) \setminus \mathcal{P}(\tau_j),\, v \in V_{i^+}(\tau_j), \tag{6.3f}$$

$$e_{i^-} \le B_v \le e_{i^+} + L_i + s_{i^+} + (\ell_{i^+} - e_{i^+}) \sum_{a \in \delta^{\text{in}}(v,\tau_j)} x_a \quad \forall i \in \mathcal{A}(\tau_j),\, v \in V_{i^-}(\tau_j), \tag{6.3g}$$

$$B_v \le \ell_{i^+} \left(1 - \sum_{a \in \delta^{\text{in}}(v,\tau_j)} x_a\right) + (\Gamma_i + \gamma) \sum_{a \in \delta^{\text{in}}(v,\tau_j)} x_a \qquad \forall i \in \mathcal{S}(\tau_j), \forall v \in V_{i^+}(\tau_j), \tag{6.3h}$$

$$B_w - B_v - s_{i^+} \le L_i \qquad \forall i \in \mathcal{A}(\tau_j),\, v \in V_{i^+}(\tau_j),\, w \in V_{i^-}(\tau_j), \tag{6.3i}$$

$$B_w \ge \tau_j + t_{(v,w)} x_{(v,w)} \qquad \forall (v,w) \in \delta^{\text{out}}((0,\dots,0),\tau_j) \setminus A^{\text{fixed}}(\tau_j), \tag{6.3j}$$

$$(C_{v,w}(\tau_k)) \qquad \forall (v,w) \in A^{\text{new}}(\tau_k) \setminus \delta^{\text{out}}((0,\dots,0),\tau_k),\, \forall k = 1,\dots,j, \tag{6.3k}$$

$$d_i \ge B_v - e_{i^-} \qquad \forall i \in \mathcal{A}(\tau_j),\, \forall v \in V_{i^-}(\tau_j), \tag{6.3l}$$

$$p_i = 1 \quad \forall i \in \mathcal{S}(\tau_j), \tag{6.3m}$$

$$x_{(v,w)} = 1,\ B_w = B_w^{\text{old}} \qquad \forall (v,w) \in A^{\text{fixed}}(\tau_j), \tag{6.3n}$$

$$p_i \in \{0,1\} \qquad \forall i \in \mathcal{A}(\tau_j) \setminus \mathcal{P}(\tau_j), \qquad d_i \ge 0 \quad \forall i \in \mathcal{A}(\tau_j), \tag{6.3o}$$

$$x_a \in \{0,1\} \quad \forall a \in A(\tau_j), \qquad B_v \ge 0 \quad \forall v \in V(\tau_j). \tag{6.3p}$$

The objective function (6.3a) minimizes the total routing cost, the total regret and the number of unaccepted requests, where $\omega_1, \omega_2, \omega_3 > 0$ are weighting parameters that can be adapted to represent the respective importance of these optimization criteria. The flow conservation constraints (6.3b) ensure that only dicycle flows in $G(\tau_j)$ are feasible. Every accepted user has to be picked up at one of its pickup nodes by exactly one vehicle (6.3c). Constraint (6.3d) is a capacity constraint on the number of vehicles. The constraints (6.3e)–(6.3g) are time-window constraints for the vehicles to arrive at events (nodes). Constraints (6.3h) guarantee that the start of service at a pickup node of a user $i \in \mathcal{S}(\tau_j)$ which has not been picked up yet, is not later than the pickup time Γ_i communicated to the user plus an additional constant γ. Furthermore, the maximum ride time of a user is bounded by constraint (6.3i), while constraints (6.3j)–(6.3k) model the travel-time from node to node. Constraints (6.3l) measure a user's regret. The constraints (6.3m) ensure that a request is contained in a vehicle's route if and only if it is accepted (indicated by $p_i = 1$). Finally, constraints (6.3n) ensure that the next solution respects the partial routes up to time τ_j, including the scheduled service times that are inherited from the previous iteration. Vehicle capacity, pairing and precedence constraints are ensured by the structure of the event-based graph. Furthermore, it guarantees that picked up users will not be relocated to any other vehicle and that they will eventually be dropped off. Note that, requests that have been accepted but have not been picked up or dropped off yet may be assigned to other vehicles in the next iteration.

6.4 A Rolling-Horizon Algorithm

We now present the essential aspects of the rolling-horizon algorithm. The approach is based on iteratively updating the dynamic event-based graph whenever new requests arrive, given the information obtained from the previous solution. Then the corresponding MILP is resolved. For each new request we have to determine whether it can be feasibly integrated into the existing schedule. If this is possible, then a schedule including the new request that minimizes routing costs and regret is computed. We impose a time limit of 30 seconds to decide how to process new requests. If the solution returned by the MILP solver is not yet known to be optimal due to this time limit, then the solution is re-optimized in the next iteration. Note that, this re-optimization can only consider variables that have not yet been fixed due to the advanced time. In the following, let δ be a timer that ensures this time limit by measuring the time in minutes needed to execute lines 4–8 in Algorithm 3. An initial feasible solution containing the initial requests is obtained by solving MILP(τ_1). Every time one or more new requests are revealed at times τ_i, $i \in \{2, \ldots, n\}$, the set of active requests is updated as $\mathcal{A}(\tau_i) = \mathcal{A}(\tau_{i-1}) \cup \mathcal{N}(\tau_i) \setminus (\mathcal{D}(\tau_i) \cup \mathcal{R}(\tau_i))$ and the dynamic event-based graph corresponding to the current time τ_i is computed. Note that, we do not have to recompute the whole graph in each iteration: All not realized pickup and delivery nodes (up to time τ_i) corresponding to dropped off and denied users and all not realized pickup nodes (up to time τ_i) corresponding to picked up users are removed from the graph together with all incident arcs. On the other hand, new nodes and arcs corresponding to new requests are added to the graph and the MILP is updated accordingly. To assure that vehicle routes computed for the current subproblem DARP(τ_i)

Algorithm 3: Rolling-horizon algorithm for the dynamic DARP.

```
(x, B, p, d) = solve(MILP(τ_1))
for i = 2 ... n do                          // new requests N(τ_i) are revealed
    Start timer δ = 0
    Determine D(τ_i), R(τ_i) and P(τ_i)
    A(τ_i) = A(τ_{i-1}) ∪ N(τ_i) \ (D(τ_i) ∪ R(τ_i))
    Compute dynamic event-based graph G(τ_i)
    Determine set of fixed arcs A^fixed(τ_i)      // fix partial routes up to τ_i
    (x, B, p, d) = solve(MILP(τ_i)) and stop prematurely when δ = Δ
    foreach request i ∈ N(τ_i) do
        if p_i = 1 then
            accept request i
        else
            reject request i
```

are consistent with the routes that have been executed up to time $\tau_i - \Delta$, the corresponding variables have to be fixed up to time τ_i before solving the next subproblem MILP(τ_i).

6.5 Numerical Results

In this section we assess the performance of Algorithm 3 based on real data from *Hol mich! App*. We use two instances that differ w.r.t. the length of the planning horizon and the number of requests. *Su_8_22* is an instance with $n = 254$ transportation requests based on accumulated data from nine consecutive Sundays in January and February 2021 with service hours from 8 a.m. until 10 p.m., i.e., $T = 840$ minutes. *Sa_6_3* consists of $n = 519$ requests and is based on accumulated data from nine consecutive Saturdays in January and February 2021 with service hours from 6 a.m. until 3 a.m. the next morning, i.e., $T = 1260$ minutes. Note that, due to the Covid-19 pandemic the demand for ridepooling services was rather low and hence we accumulated requests to obtain realistic instances. Moreover, the ridepooling cabs which are equipped with six seats were not allowed to transport more than three passengers at a time, i.e., $Q = 3$. We used linear regression to approximate unknown travel times from distances and from the known travel times between the pickup and delivery locations of the requests. More precisely, as for the artificial test instances used in Section 4.4, the costs c_a were computed in an OpenStreetMap network of Wuppertal using the Python APIs OSMnx and NetworkX. Then travel times t_a were computed from the regression line $t_a = 1.8246\, c_a + 2.369$. The length of the pickup time window for each user is 25 minutes, and the lower bound of the pickup time window is equal to the time when the transportation request was submitted plus the response time of the algorithm, i.e., $e_i = \tau_i$. Moreover, the maximum ride time of request i is equal to $t_i + \max(10, 0.75\, t_i)$ minutes. The service time for every request is set to 0.75 minutes and the number of requested seats varies from one to three, i.e., $q_i \in \{1, 2, 3\}$. The delivery time window is computed based on the pickup time window, the direct travel time, the maximum ride

time and the service time. The maximum delay of communicated pickup time is set to $\gamma = 5$ minutes. After some preliminary testing, the parameters in the objective function (6.3a) are set to $\omega_1 = 1$, $\omega_2 = 60$ and $\omega_3 = 0.1$. Due to the accumulation of request data, we were not given a fixed number of vehicles by the service provider. An evolution of the number of requests during service hours is depicted in Figure 6.2.

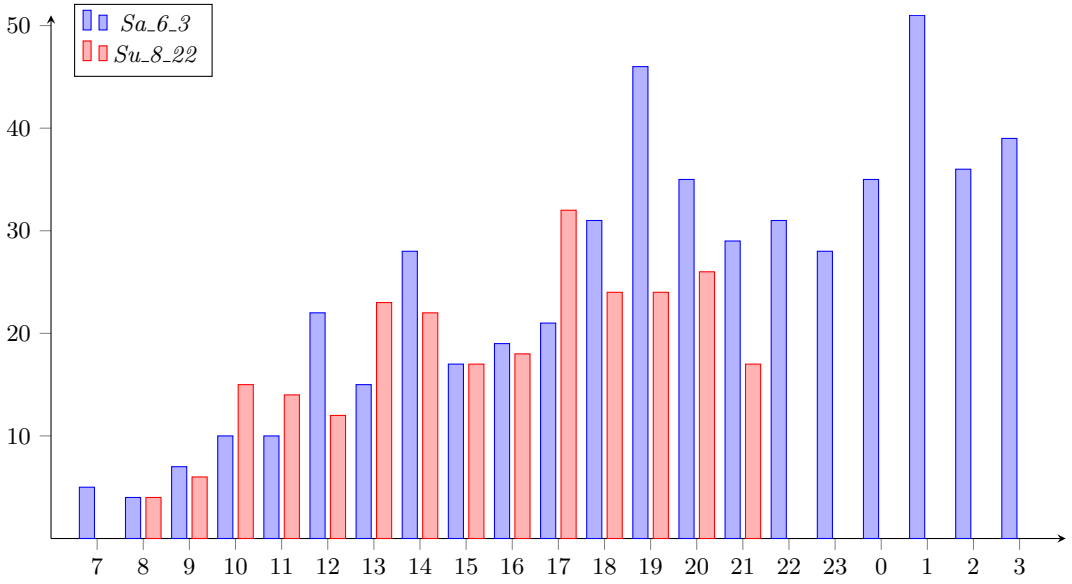


Figure 6.2: Evolution of number of requests during service hours.

In the peak hour, there are 51 requests in instance *Sa_6_3* and 32 requests in instance *Su_8_22*. The average length of a direct trip, i.e., driving from pickup to delivery location without any additional stops, in both instances is 8.4 minutes. In our tests we evaluate different fleet sizes and solve instance *Sa_6_3* with $K \in \{12, 14, 16\}$ and instance *Su_8_22* with $K \in \{6, 8, 10\}$ vehicles.

Algorithm 3 was implemented in C++ and all computations were carried out on an Intel Core i7-8700 CPU, 3.20 GHz, 32GB memory using CPLEX 12.10. The code is available in the git repository Gaul (2022b). The computational results can be found in Table 6.1. For all instances we report the following average values per accepted request: the routing costs C, the regret in minutes E, the waiting time from the time of submitting the request until the time of pickup in minutes W, the ride time in minutes RT, the average time to answer a new request in seconds S, the percentage of requests that are rejected RR, and the number of times CPLEX was terminated prematurely due to a timeout CT. Furthermore, we listed the average detour factor DF, the mean occupancy MO, the percentage of empty mileage EM and the system efficiency SE, which are measures to evaluate the operational efficiency of ridepooling systems. The computation of these measures is based on Liebchen et al. (2021):

$$\text{average detour factor} = \frac{\text{passenger kilometers driven}}{\text{passenger kilometers booked}}$$

Inst.	K	C	E	W	RT	DF	MO	EM	SE	S	RR	CT
Sa_6_3	12	4.4	12.2	9.7	11.6	1.1	1.6	0.3	1.0	2.9	3.5	1
Sa_6_3	14	4.4	12.2	9.6	11.7	1.1	1.6	0.3	1.0	2.8	3.3	2
Sa_6_3	16	4.4	11.8	9.4	11.5	1.1	1.5	0.3	1.0	2.7	3.1	1
Su_8_22	6	4.7	15.9	13.0	12.0	1.2	1.5	0.3	0.9	0.5	3.5	0
Su_8_22	8	4.6	12.3	10.0	11.5	1.1	1.5	0.3	0.9	0.4	1.6	0
Su_8_22	10	4.6	11.8	9.7	11.4	1.1	1.5	0.3	0.9	0.3	1.6	0

Table 6.1: Computational results for instances from *Hol mich! App*.

$$\text{mean occupancy} = \frac{\text{passenger kilometers driven}}{\text{vehicle kilometers occupied}}$$

$$\text{percentage of empty mileage} = \frac{\text{empty mileage}}{\text{total vehicle kilometers}}$$

$$\text{system efficiency} = \frac{\text{mean occupancy} \cdot (1 - \text{ percentage of empty mileage})}{\text{average detour factor}}$$

The results confirm that Algorithm 3 can quickly answer and schedule new requests. No CPLEX timeouts occurred in any run of a *Su_8_22* instance. Thus, all 254 requests are either inserted optimally in the given schedule, given the solution of the preceding iteration, or they are rejected due to infeasibility or unacceptable costs. For the larger *Sa_6_3* instances very few timeouts occurred, and CPLEX terminated prematurely only one or two times out of the 404 iterations[1]. This affected the insertion of five out of 519 requests. The relative gap in these iterations ranged from 0.4% to 0.5%. Moreover, a re-optimization was necessary only in 0.5% of the iterations, which implies that only a very low percentage of requests was rejected while there would have been a feasible and profitable insertion position. From comparing the results for *Su_8_22* and *Sa_6_3* for the different fleet sizes, it becomes evident that by the use of additional vehicles the average routing costs, the average regret, the average waiting time and the average trip length (except *Sa_6_3* with $K = 14$) per accepted user decrease or remain constant. The average detour factor, the mean occupancy, the percentage of empty mileage and the system efficiency remain (nearly) constant for the different values of K, while the percentage of rejected requests decreases with an increasing number of vehicles. The average time to answer new requests ranges from 2.7 to 2.9 seconds (*Sa_6_3*) and 0.3 to 0.5 seconds (*Su_8_22*) on average, demonstrating that Algorithm 3 is stable under different vehicle configurations.

[1]There are less than 519 iterations since several requests are revealed at the same time.

6.6 Summary

We present a rolling-horizon approach for the solution of the dynamic dial-a-ride-problem that adaptively updates a dynamic event-based MILP formulation. The latter is based on a dynamic event-based graph, which is updated every time new requests are revealed: Nodes corresponding to picked up, dropped off or denied requests are (partially) removed from the graph together with incident arcs, while nodes corresponding to the new requests and the remaining active requests are added. This approach can be distinguished from classic solution methods for the dynamic DARP in that it guarantees optimal insertion positions for new requests w.r.t. the previous routing decisions and given that a prespecified time limit of 30 seconds is not violated. Numerical experiments on medium-sized instances from a recently established ridepooling service in the city of Wuppertal confirm the efficiency and reliability of this approach. By adapting the weighting parameters in the objective function, different preferences w.r.t. service cost and customer satisfaction can be implemented. The approach can also be used to assess the quality gain when increasing the fleet size or when changing other parameters in the model. In the next chapter, we simulate the substitution of line-based public transport services by ridepooling during periods of low demand, e.g., during night time. To model the routing decisions of the ridepooling service, the proposed rolling-horizon algorithm is used.

7 Ridepooling and Public Bus Services: A Comparative Case-Study

In this chapter, we compare the service quality of time-tabled buses in the late evening hours in the city of Wuppertal, Germany to that of on-demand ridepooling cabs. To evaluate the service quality of ridepooling as compared to bus services, and to simulate bus rides during the evening hours, transport requests are generated using a predictive simulation. To this end, a framework in the programming language R is created, which automatically combines generalized linear models for count regression to model the demand at each bus stop. Furthermore, we use classification models for the prediction of trip destinations. To solve the resulting dynamic dial-a-ride problem, the rolling-horizon algorithm is extended by a *feasible-path* heuristic to further reduce its computation time in presence of request peaks. The quality of the ridepooling service is evaluated using performance measures such as transportation time, waiting time, ride time and regret. This allows an estimation of the number of cabs needed depending on the weekday to realize the same or a better general service quality as the bus system.

The optimization problem underlying the ridepooling service is a deterministic, dynamic and homogeneous DARP. We allow transport requests to be rejected and the objective function is a weighted-sum minimizing the routing costs, the regret and the number of unaccepted requests.

The remainder of this chapter is structured as follows: After a short description of the problem setting, we outline the concept of the case study in Section 7.1. Section 7.2 deals with the statistical modeling and simulation of transportation data. The feasible-path heuristic introduced to enhance the performance of the rolling-horizon algorithm in the face of request peaks is illustrated in Section 7.3. After all prerequisites are made, a computational study evaluating the effects of the replacement of buses by ridepooling cabs with respect to the service quality is conducted in Section 7.4. Finally, based on the computational results, some conclusions are drawn in Section 7.5. This chapter is based on the case-study conducted in Asatryan et al. (2023).

7.1 Case Study Outline

In the evening hours, it is necessary to offer a transport option, but buses are hardly working to capacity. Figure 7.1 shows the number of bus rides per hour during the first half of 2019. It is obvious that in between 10:00 PM and 3:59 AM the actual number of passengers (marked in red) is lowest, because of equally low supply and demand. For example, between 2:59 AM and 3:59 AM, there are approximately 600 requests, i.e., on average 23 requests a day during the first half of 2019.

This study investigates the advantages and disadvantages of (partially) substituting bus trips during periods of low demand by ridepooling services with respect to criteria related

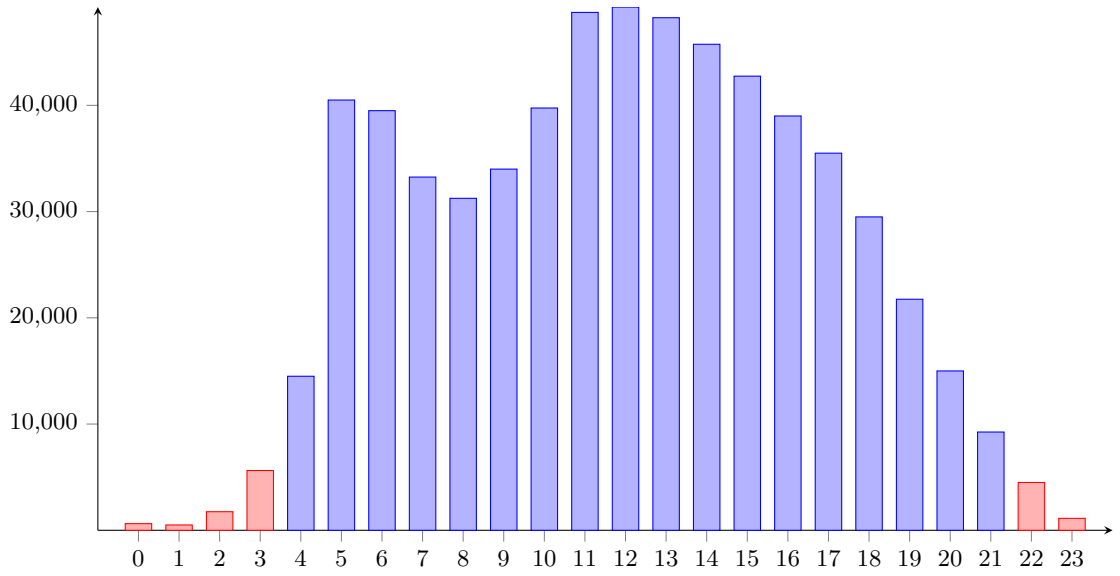


Figure 7.1: Hourly Distribution of W-LAN registered WSW bus rides in the city of Wuppertal during the first half of 2019.

to service quality. We exemplarily conduct the study in the city of Wuppertal (a mid-sized city with about 350,000 inhabitants) and simulate the substitution of the bus trips in the evening hours between 10 p.m. and 4 a.m. by the *Hol mich! App* ridepooling service. This includes, for example, the transportation time, the waiting time, and the regret, i.e., the excess ride time as compared to using a private car. Since this study does neither address social criteria and constraints nor the acceptance of ridepooling services in the general public, the results have to be assessed with care.

The *Hol mich! App* ridepooling operates electrical cabs with six seats and, in contrast to bus services, is not tied to fixed line plans and schedules. The collaboration with the local transport provider WSW within the project bergisch.smart_mobility[1] enabled us to get access to ridepooling as well as bus service data. Using ride statistics recorded by logging data of WSW Wi-Fi users, we model and simulate evening trip scenarios, taking into account the influence of the weekday and the full hour of a trip, as well as information on school holidays.

Transport requests and the destination of rides are modeled statistically in a three step procedure. We first evaluate the number of requests on the basis of mobile logging data for each bus stop in the ridepooling area with destination in the same area. This leads to a Poisson count regression model for each bus stop with the hour, the weekday and an indicator for school holidays as covariates that influence the number of requests. In a second step, the number of persons traveling jointly on the trip is simulated using the statistics of *Hol mich! App* transport requests. Ultimately, we model the probability of a specific destination for the trip on the basis of logging out data from the mobile internet data using a multicategorical logistic regression model with the aforementioned covariates

[1] `https://www.bergischsmartmobility.de/en/the-project/`

per bus stop. On the basis of our statistical models, we simulate transport requests in the area of the *Hol mich! App* ridepooling service by the Monte Carlo method. We thereby generate stochastic scenarios of transport requests that approximately correspond to those of the real world transportation by bus.

We evaluate the service quality of shared evening rides in comparison to line based bus services by modeling the routing decisions of the ridepooling service as a dynamic DARP and apply the rolling-horizon algorithm (Chapter 7), enhanced by a feasible-path heuristic to reduce computation time, to a sufficiently large number of such simulations. In the following, we describe the simulation of transport requests.

7.2 Statistical Modeling and Simulation of Transportation Data

In this section we describe the simulation environment we use to generate realistic transport requests. In order to achieve this, we analyze and connect various public domain and proprietary data sets. On these data sets we base a statistical modeling of rides which so far have been conducted using the public bus system at Wuppertal. This is, among other data sources, based on anonymized Wi-Fi logging data provided by the public transportation agency WSW for the time span from January 1st to June 30th 2019.

For an introduction into statistical analysis and the tools used in this chapter we refer to Faraway (2016), Hastie et al. (2009), and Knight (1999).

In the following, we first describe the data sources and thereafter describe the statistical modeling and simulation approach.

7.2.1 Description of the Data Sets

Our statistical analysis and modeling is based on the following data sets:

- *WSW-LAN data* consisting of information on the Wi-Fi usage by public transport passengers within the first half of the year 2019. We utilize this data to model the transport requests depending on the bus station of departure, the hour of the day, the weekday and public school holidays. This data is collected from logging information to the Wi-Fi in the WSW buses. The data is anonymized and not proprietary and is provided by the WSW under the collaboration of the bergisch.smart_mobility consortium.

- To provide the information, whether a day is public school holiday or not, we link the data of *School holidays*[2] in the state of North Rhine-Westphalia with the dates in the WSW-LAN data set.

- *VRR public transport timetables*[3] in General Transit Feed Specification (GTFS) format. This data set is utilized to analyze regret times (waiting times) by finding the predecessor bus connecting the same bus stops in the time table data.

[2] `https://www.feiertagskalender.ch/export.php?geo=3069&jahr=2019&klasse=3&hl=en`

[3] `https://www.opendata-oepnv.de/ht/de/organisation/verkehrsverbuende/vrr/openvrr/datensaetze`

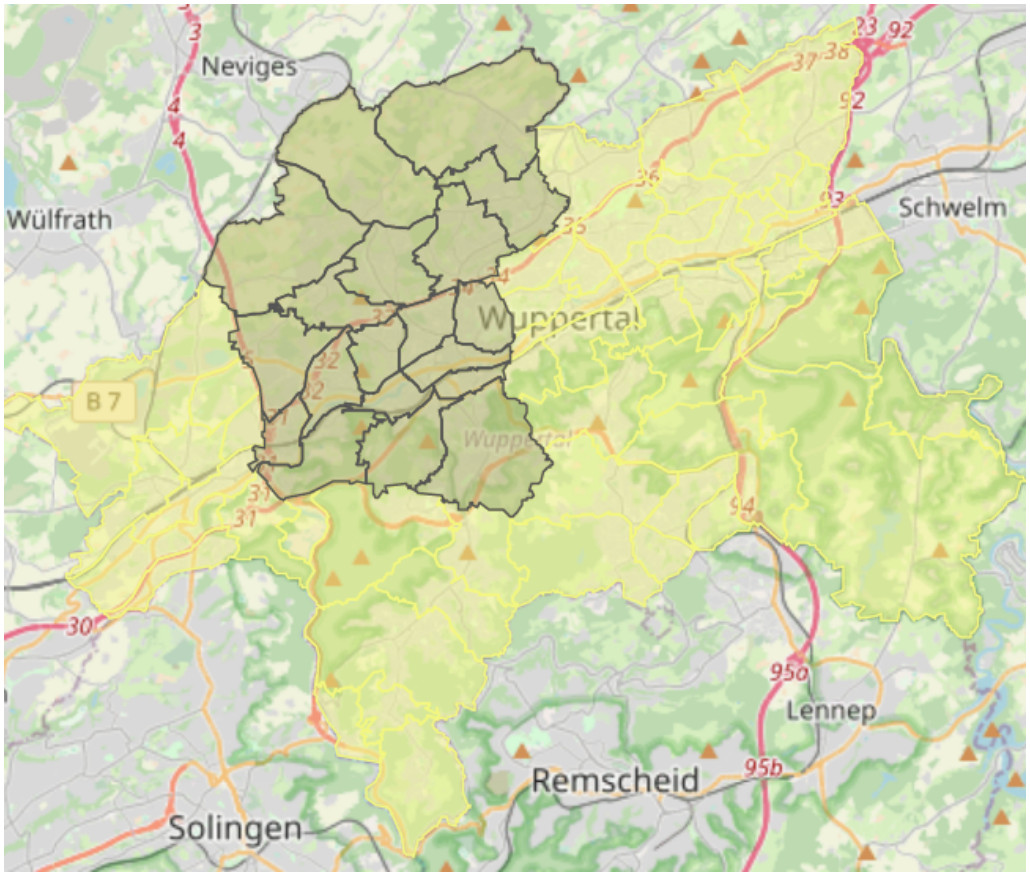


Figure 7.2: Service area of *Hol mich! App*

- *Wuppertal districts*[4] as spatial data in the coordinate system EPSG:25832, see Figure 7.2. This geostatistical data is used to determine whether geocoordinates associated with bus stops are in the ridepooling operation district of the *Hol mich! App*. In this way, we filter out only those bus rides that are inside the region where *Hol mich! App* is available.

- The proprietary data set consisting of all ridepooling requests in 2021 is provided by *Hol mich! App* for the purpose of this study. We use this data to statistically model the group size of a single transport request and assimilate it to the distribution of sizes observed in the *Hol mich! App* data set.

The following variables of the WSW-LAN data set are the most relevant ones for our purposes:

- *Time of the trip start/stop* (named StartTime/StopTime),

- *Station of the trip start/stop* (named StartStation/StopStation).

[4] `https://www.offenedaten-wuppertal.de/dataset/quartiere-wuppertal`

From these variables we extract the date, the weekday and the full hour of the trip start, and we add the school holiday information.

7.2.2 Statistical Modeling of Transport Requests

Generalized linear models (GLM) are the standard models of statistical analysis beyond standard regression, see, e. g., Faraway (2016), Hastie et al. (2009), and Knight (1999). As transportation events at bus stops, i.e., a person taking a bus at a specific time at a given hour, are counted by integer values, the standard regression based on the continuous normal distribution is inadequate. Here, we therefore choose a GLM-based *Poisson count regression* approach, which we now shortly explain:

Let x_i be the covariate (daytime, hour and an indicator of school holidays in our case) and y_i the actual number of rides observed at a particular bus stop using the Wi-Fi logging information and geolocation of the bus that is matched to geolocations of the bus stops. A common assumption in count regression is to model the corresponding random number Y_i from which y_i is a sample, as Poisson distributed

$$P(Y_i = y) = \mathrm{Po}(\lambda_i)(\{y\}) = e^{-\lambda_i} \frac{\lambda_i^y}{y_i!}\,, \qquad y \in \{0, 1, 2 \ldots\}$$

where λ_i is the intensity parameter or the expected value of requests for the instance i. In GLM this intensity parameter is modeled using a link function that maps the expected value to a link variable which is modeled as a linear combination of the covariates plus an intercept. For Poisson count regression, it is common practice to use the canonical link function $\log(\cdot)$, hence

$$\log(\lambda_i) = \beta^\top x_i \iff \lambda_i = \exp(\beta^\top x_i).$$

Inserting this to the Poisson distribution enables one to determine the parameters $\beta \in \mathbb{R}^q$ via Fisher scoring maximizing the likelihood, see Knight (1999). Note that, x_i is composed of one intercept term, 6 dummy 0-1 variables indicating the weekdays Tuesday–Sunday (with all those dummies equal zero indicating Monday) and 23 dummy variables for the hours of daytime different from 12–1 PM. The set of covariates is complemented with a further dummy variable for school holidays, so that we obtain $q = 1{+}6{+}23{+}1 = 30$ parameters that are estimated from the data for each of the 516 bus stops in the *Hol mich! App* ridepooling area separately. This is automated by adequately filtering and accumulating the requests and application of the `glm` function in the statistics programming language R[5].

In this way we estimate the distribution of raw transport requests on the basis of single passengers. While this adequately models single bus passengers, it does not model the group size of a ridepooling request. To obtain a request model with adequate group sizes, we use the statistics of group sizes of the ridepooling requests of the *Hol mich! App* as obtained from the corresponding data set and displayed in Table 7.1. To model the boardings at each station during any full hour, we naturally assume that boardings in disjoint time intervals are independent. Our approach considers group rides, too, and we

[5] `https://r-project.org`

Group size	1	2	3	4	5	6
Probability	0.804	0.153	0.026	0.011	0.004	0.002

Table 7.1: Distribution of passenger group sizes

assume that the different groups have different boarding times (practically, the hour, the minutes, the seconds and the milliseconds of the boarding times of two different groups cannot be all identical). Moreover, we assume that the boarding rate within each full hour does not change essentially. The latter assumption could seem even more reasonable for shorter time intervals, but the disadvantage here would be the smaller set of observations, not sufficient for stochastic simulations.

In order not to increase the average overall number of passengers, downscale the intensities λ_i by the average passenger group size (i.e., by 1.264 in the case of Table 7.1). After using the downscaled intensities for generation of a statistically realistic random number of requests, for each request the size of groups are generated according to their distribution.

7.2.3 Statistical Modeling of Destinations

To simulate the destinations, we use the multinomial logit model, see Faraway (2016). Based on the same set of 30 dummy covariates x_i as described in the previous subsection, we train a $(s-1) \times q$ Θ_i matrix (with $q-1$ the number of covariates), which generates an $s-1$ dimensional activation vector $z_i = z(x_i \mid x_i, \Theta_i)$. Thereby, s denotes the number of bus stations in the ridepooling operation area that have ever been reached from the starting bus station under consideration. One frequent destination bus station from the given bus station of departure is selected, put at the first index position in z_i and associated with intensity $z_{i,1} = 0$ to avoid overparametrization. We thereafter send the entire activation scores z_i through the softmax activation function

$$p(j \mid x_i, i, \Theta_i) = \text{softmax}(z_i)_k = \frac{\exp(z(x_i, \Theta_i)_j)}{\sum_{l=1}^{s} \exp(z(x_i, \Theta_i)_l)}) \quad \text{for} \quad j \in \{1, \ldots, s\},$$

which gives us the vector of discrete probabilities of choosing a specific destination among the s existing options.

For each of the 516 stations of departure, the above multicategorical logistic regression (or equivalently a shallow neural net without hidden layer) is fitted using the R package `nnet` after an adequate filtering for the allowed destinations. Note that, due to the shortcomings of anonymous Wi-Fi tracking data, no bus rides requiring a transfer can be modeled with our approach.

7.2.4 Simulation of Virtual Data Sets

Based on the statistical models for transport requests, group sizes and destinations, we are now able to sample from the corresponding distributions in order to create a virtual request scenario. In detail, this is done for every station simulating an entire week, hour

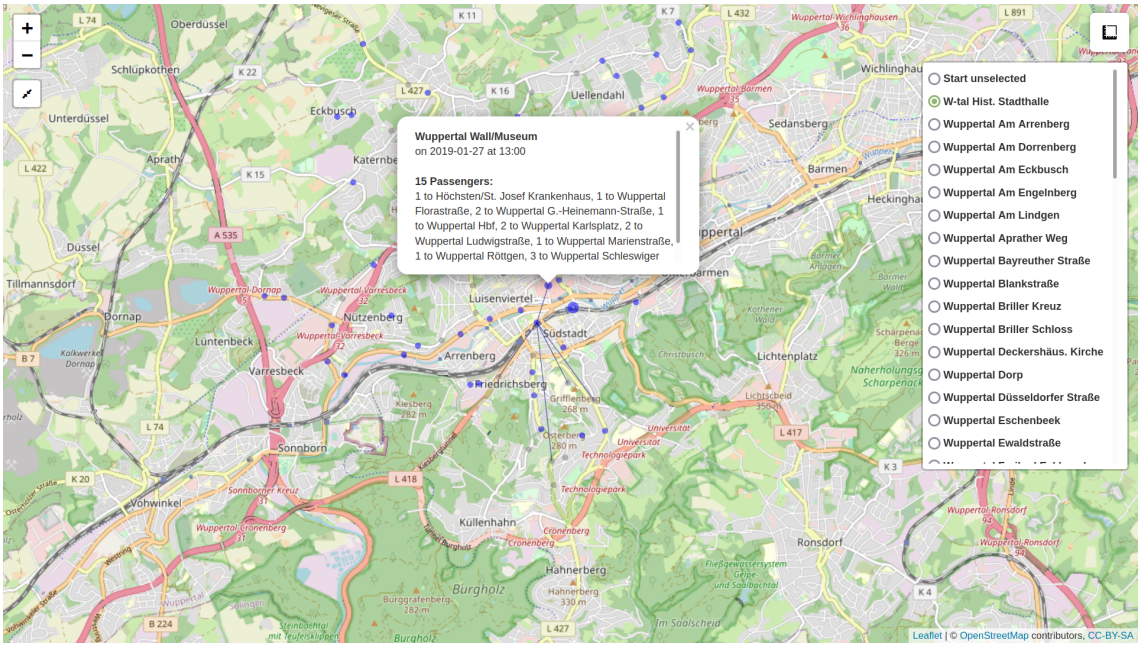


Figure 7.3: Visualization of a ridepooling scenario on the interactive map.

by hour, either for school holidays or not. Within the given hours, the exact time of the transport requests is uniformly distributed.

We also provide an interactive tool for visualization of the request scenarios. The user can select the outbound time (in full hours), the day of the week, and the school holiday information. This results in a Leaflet-based HTML map where stations are displayed as disc-shaped markers with radii proportional to the square root of the corresponding entry number. Hovering over a marker displays the name of the station, and clicking on it displays boarding information, i.e., number of boardings, start time (in full hours), day of the week, school holiday information, and destination stops. For a better view, one can select the outgoing station in the right panel, then the corresponding destinations will be displayed by connecting rays. We use blue color for all the stations where there are boarding passengers, whereas the stations where no passengers get on a bus are displayed using the gray color, see Figure 7.3 for an example scenario.

7.2.5 Modeling of Waiting Times

To model waiting times for bus rides, we analyze the *VRR public transport timetables* by searching a preceding trip in the time tables that directly connects the bus stops of origin with the destination which gives us the maximum waiting time. If the maximum waiting time exceeds a threshold of two hours, the value is set to the threshold. The waiting time is now simulated uniformly in between the maximum waiting time and zero, as, potentially, the passenger might have wanted to travel at any of these times and we have no reason to assume a preference for any point of time in the given time interval.

7.3 Feasible-Path Heuristic

To solve an instance of the dynamic DARP we use the rolling-horizon algorithm (Chapter 6). To further reduce computation times in order to compensate for request peaks, the algorithm is extended by a feasible-path heuristic. By means of the heuristic, the complexity of the MILPs that are solved iteratively can be controlled. Thus, the solution strategy we use is easily adjustable to different volumes of requests, giving way to obtain near-optimal solutions when demand is low, and speeding up computation times when it is high. A description of the newly added heuristic is given below.

The feasible-path heuristic ensures that every time a new request i arrives, only nodes and arcs corresponding to the ρ most temporally and spatially proximate paths $j^+ \to i^+ \to j^- \to i^-$ and $j^+ \to i^+ \to i^- \to j^-$ are added to the dynamic event-based graph. Only paths of pairwise feasible requests i, j are considered. The spatial proximity of the path $j^+ \to i^+ \to j^- \to i^-$ is measured in terms of the function

$$\omega_1 \left(\bar{c}_{j^+ i^+} + \bar{c}_{i^+ j^-} + \bar{c}_{j^- i^-} \right),$$

whereas the temporal proximity is calculated as

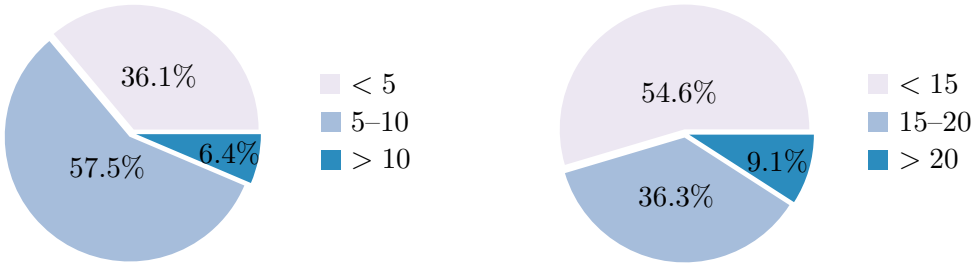
$$\omega_2 \left(2 \left(\max(e_{i^+}, e_{j^+} + s_j + \bar{t}_{j^+ i^+}) + s_i + \bar{t}_{i^+ j^-} \right) - e_{j^-} + s_j + \bar{t}_{j^- i^-} - e_{i^-} \right).$$

Here, ω_1 and ω_2 denote the weighting parameters used in the objective function (6.3a). The temporal and spatial proximity of the path $j^+ \to i^+ \to i^- \to j^-$ is calculated accordingly. The temporal and spatial proximity of a path which is infeasible w.r.t. time window or ride time constraints is set to infinity.

Let ρ_i denote the number of feasible paths associated with a new request i. The number of feasible paths that are considered as a basis for the computation of new nodes and arcs in the dynamic event-based graph is denoted by $\rho = \max(\rho_{\text{abs}}, \rho_{\text{rel}} \rho_i)$, which is composed of a fixed bound ρ_{abs} on the number of considered feasible paths and a relative bound ρ_{rel} on the percentage of considered feasible paths. By the use of the heuristic, the size of the graph and the number of modifications of constraints and new variables in the dynamic event-based MILP at every iteration is kept small. Nevertheless, we lose an important characteristic of the rolling-horizon algorithm: if MILP(τ_j) is solved to optimality within the time limit, it is not guaranteed that the solution returned by the MILP solver is the global optimal solution w.r.t. the current vehicle routes computed in prior iterations. In preliminary tests, we observed that using the heuristic with $\rho_{\text{abs}} = 10$ and $\rho_{\text{rel}} = 0.25$, average routing costs increased by 2.8% while average regret decreased by 0.6% and computational time decreased by 65%. Thus, the rolling-horizon-algorithm paired with the feasible-path-heuristic is able to produce high-quality solutions in a significantly reduced amount of time.

7.4 Numerical Results

In the first part of this section, 30 weeks of simulated ridepooling scenarios with service hours from 22:00 to 03:59 are considered and optimized tours are computed using the rolling-horizon algorithm enhanced by the feasible-path heuristic presented in the previous



(a) Distribution of travel times.

(b) Distribution of ride times.

Figure 7.4: Distribution of travel and ride times (in minutes) among the artificial requests.

section. To evaluate the quality of the transportation via ridepooling as compared to the bus service, we rely on the quality measures which are regularly used as parts of the objective function in ridepooling applications, see Section 3.5. Secondly, to measure the quality of the bus service, the WSW-LAN data set is extended by simulated waiting times at the bus stop as described in Subsection 7.2.5, and the two modes of transportation are compared.

In total, 210 simulated instances are solved, 30 instances for each day of the week. We use the following parameter settings for the feasible-path-heuristic and the rolling-horizon algorithm for the tour computations: $\rho_{\text{abs}} = 10$, $\rho_{\text{rel}} = 0.25$, and $\Delta = 45$. All other parameters are given by the conditions of transport of the service provider, i.e., the service time is considered to be constant and equals $s_i = 45$ seconds and the length of the pickup time window is given by $\ell_{i^+} - e_{i^+} = 25$ minutes. The number of transport requests, the number of requested seats, and pickup and delivery locations are taken from the simulated transport data. The distances between the latter are computed with help of the Python packages `OSMnx` and `NetworkX`, and the travel time between two locations j_1, j_2 is computed based on a linear regression using distances and travel times from the *Hol mich! App* data set: $\bar{t}_{j_1 j_2} = 2.3634 \cdot \bar{c}_{j_1 j_2} + 0.2086$. A visualization of the distribution of travel and ride times among the the transport requests is given in Figure 7.4. It remains to determine the size of the vehicle fleet.

For the purpose of this simulation we use a fixed number of vehicles depending on the weekday; the size of the vehicle fleet should not vary from week to week or throughout the evening. Hence, for each weekday, using our estimated transport requests, we calculate the total number of requests per hour over 30 weeks and take the average among the 30 weeks. Then, the hour with the maximum average number of requests is used as a basis to estimate the number of vehicles needed. Preliminary tests show that a ratio of 8 requests per vehicle and hour on average is a good starting point. In the following we will refer to this parameter setting as scenario A. Additionally, we simulated also with a fleet size reduced by one (scenario B) and increased by one (scenario C). The corresponding sizes of the vehicle fleet are given in Table 7.2. The code of the feasible path heuristic is written in C++ and can be found in the same git repository as the rolling-horizon algorithm. The MILPs are solved using IBM ILOG CPLEX 12.10. The computations are performed on

	max. avg. # requests per hour	A	B	C
Monday	37.2	5	4	6
Tuesday	34.6	5	4	6
Wednesday	26.7	4	3	5
Thursday	30.8	4	3	5
Friday	24.1	4	3	5
Saturday	17.9	3	2	4
Sunday	26.5	4	3	5

Table 7.2: Number of vehicles depending on the weekday and scenario.

an Intel Core i7-8700 CPU, 3.20 GHz, 32 GB memory.

To analyze the service quality of the bus service in comparison to the ridepooling service, we report the following average values (in minutes) per realized trip using the bus or the ridepooling service: the regret E, the waiting time W and the ride time RT. These values are also used to compare the quality of the computed routes using different sizes of vehicle fleets in Section 6.5. Furthermore, we report the average transportation time TT per trip (i.e., the time in minutes from the time of submitting the request/ showing up at the bus stop until the arrival at the request's destination). In addition, we present the following quantities for the simulated bus requests transported via ridepooling, as they are criteria of our weighted-sum objective function: Total routing costs (f_c) and number of rejected requests (we report the percentage of rejected requests RR).

Table 7.3 illustrates the results of the simulation compared against the bus service, where for each scenario and each weekday average values over 30 weeks are reported. Additionally, in Table 7.4 we exemplarily list average hourly results for Monday evening and each scenario. In both tables, we report average regret, waiting time, ride time and transportation time for the bus service based on trip data from the WSW-LAN data set. In order to compare the quality of both transportation modes, the trip data of each passenger in the WSW-LAN data set is extended by a simulated time the passenger has to wait at the stop for the bus to arrive, see Section 7.2.5. As for the ridepooling service, we assume a service time of 0.75 minutes for passengers to enter or leave a bus. Due to the reduced amount of passengers during the late evening hours this is a realistic assumption. The number of buses simultaneously in service is estimated by WSW to be a minimum of 10 buses during the week. Hence we use this number for comparison in the following.

It becomes obvious from Table 7.3, that we can achieve a much higher service quality by ridepooling than by the bus service. While the daily average regret ranges from 9–11 minutes in scenario A, 11–14 minutes in scenario B and 7–9 minutes in scenario C, using the bus service the average regret is between 19 and 26 minutes. Similar tendencies hold for average waiting time, average ride time and average transportation time. In scenario A, where the number of vehicles is selected so that, on average, eight transport requests

	K	f_c	RR	E	W	RT	TT
Scenario A							
Mon	5	251.7	0.8	9.6	8.1	6.9	15.8
Tue	5	268.9	0.0	9.5	8.0	7.0	15.7
Wed	4	238.5	0.9	10.3	8.8	6.8	16.4
Thu	4	231.1	1.2	10.3	8.8	6.9	16.5
Fri	4	217.2	0.3	9.6	8.2	6.7	15.7
Sa	3	148.7	0.4	9.2	8.0	6.3	15.1
Su	4	176.8	0.6	9.7	8.3	6.8	15.9
Scenario B							
Mon	4	243.5	3.3	11.2	9.5	7.2	17.4
Tue	4	262.2	2.1	11.8	10.0	7.2	18.0
Wed	3	222.9	6.6	13.4	11.5	7.2	19.4
Thu	3	218.1	6.8	12.6	10.7	7.3	18.8
Fri	3	207.8	3.9	12.2	10.4	7.0	18.3
Sa	2	140.9	6.9	12.5	10.8	6.8	18.4
Su	3	167.7	5.1	11.6	9.9	7.1	17.8
Scenario C							
Mon	6	251.8	0.1	8.5	7.1	6.8	14.7
Tue	6	267.6	0.0	8.4	7.0	6.8	14.6
Wed	5	237.1	0.0	8.4	7.2	6.5	14.5
Thu	5	234.2	0.0	8.9	7.5	6.8	15.1
Fri	5	216.5	0.0	8.1	6.9	6.5	14.1
Sa	4	148.6	0.0	7.7	6.7	6.1	13.6
Su	5	178.3	0.0	8.1	7.0	6.6	14.4
Bus service							
Mon	10			24.1	17.7	11.2	29.7
Tue	10			24.7	19.0	10.7	30.4
Wed	10			25.1	19.0	11.0	30.7
Thu	10			22.6	16.1	11.3	28.1
Fri	10			22.4	16.8	10.5	28.0
Sa	10			19.5	13.7	10.5	25.0
Su	10			20.1	14.6	10.2	25.6

Table 7.3: Average computational results over 30 weeks of simulated request data and average transportation data computed over bus trips during first half of 2019.

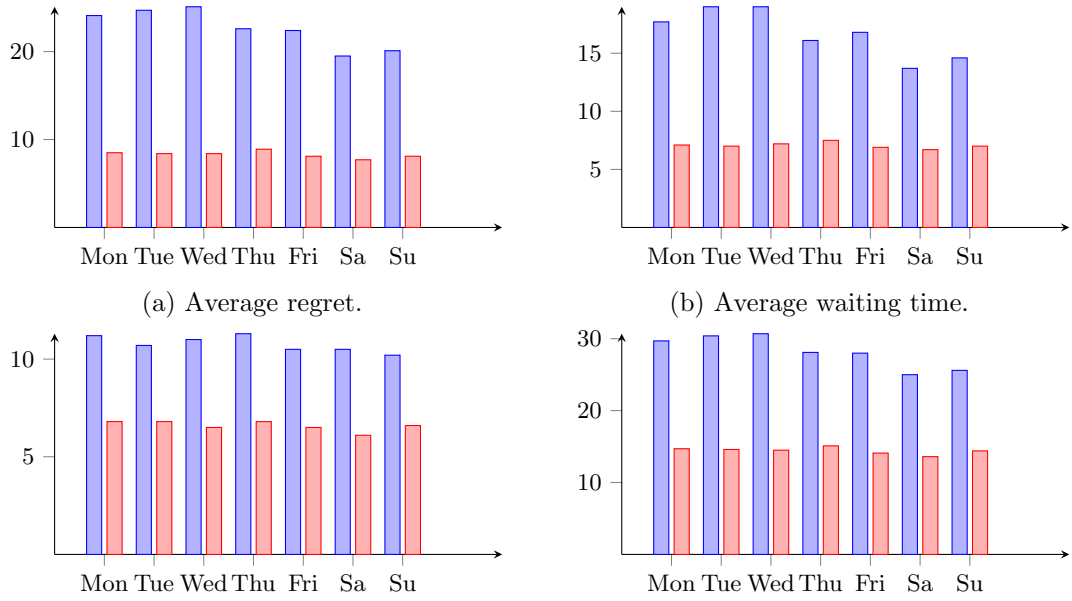
during the peak hour of requests can be accepted, on average at most 1.2% of the transport requests are denied. While this is an acceptable ratio for a ridepooling service, we note that the replacement of bus service generally requires a full coverage of all transport requests. Thus, also scenario B, where one vehicle less per weekday is used than in scenario A, is not eligible for a fair comparison. In scenario C, where one vehicle more than in scenario A is used, the average percentage of denied requests is 0.0% for all weekdays but Monday, where it is 0.1%. Hence, the size of the vehicle fleet for Mondays should be increased to seven vehicles, while for the remaining week the initial estimate is sufficient. Taking a closer look at scenario C, it becomes evident, that using the bus service to transport late evening requests, average regret and average waiting time are about 2.5 to 3 times

	K	f_c	RR	E	W	RT	TT
Scenario A							
Mon	5	251.7	0.8	9.6	8.1	6.9	15.8
Mon 22-23	5		0.0	8.2	6.7	6.3	13.8
Mon 23-00	5		0.0	6.7	6.3	5.6	12.6
Mon 00-01	5		0.0	6.0	5.6	5.5	11.9
Mon 01-02	5		0.0	5.7	5.3	7.0	13.0
Mon 02-03	5		0.0	8.1	7.1	7.5	15.4
Mon 03-04	5		1.9	12.1	10.0	7.4	18.2
Scenario B							
Mon	4	243.5	3.3	11.2	9.5	7.2	17.4
Mon 22-23	4		0.0	9.3	7.6	6.4	14.8
Mon 23-00	4		0.0	7.2	6.7	5.6	13.1
Mon 00-01	4		0.0	6.0	5.5	5.6	11.9
Mon 01-02	4		0.0	6.2	5.8	6.8	13.4
Mon 02-03	4		0.0	8.8	7.5	7.8	16.1
Mon 03-04	4		7.8	14.0	11.6	7.3	19.7
Scenario C							
Mon	6	251.8	0.1	8.5	7.1	6.8	14.7
Mon 22-23	6		0.0	7.9	6.4	6.2	13.5
Mon 23-00	6		0.0	6.4	5.9	5.7	12.3
Mon 00-01	6		0.0	5.6	5.3	5.5	11.5
Mon 01-02	6		0.0	5.4	5.0	6.7	12.5
Mon 02-03	6		0.0	7.5	6.6	7.4	14.7
Mon 03-04	6		0.4	10.1	8.4	7.2	16.4
Bus service							
Mon	10			24.1	17.7	11.2	29.7
Mon 22-23	10			24.1	17.7	11.2	29.6
Mon 23-00	10			24.5	18.1	11.2	30.1
Mon 00-01	10			20.1	15.0	10.1	25.9
Mon 01-02	10			24.7	18.3	11.2	30.2
Mon 02-03	10			24.1	17.7	11.2	29.7
Mon 03-04	10			24.2	17.8	11.2	29.7

Table 7.4: Hourly average computational results over 30 Mondays of simulated request data and average transportation data from Monday bus trips during first half of 2019.

as high, while average ride time and transportation time are about 1.5 to 2 times as high as compared to using the ridepooling service. These ratios can be improved even further when taking into account the missing vehicle on Monday in scenario C: we now take a look at the hourly results for Monday evening. Note, that the average results reported in Table 7.4 are computed individually for every hour, which explains for example the fact that the percentage of denied requests from 3 a.m. to 4 a.m. is higher than the percentage of denied requests for Mondays as a whole day. From Table 7.4 we deduce that the number of vehicles we assumed for Monday is sufficient, only for the hours of 3 a.m. to 4 a.m., an additional vehicle is needed. In this hour, also the average regret, waiting time, ride time

and transportation time is larger than in the rest of the evening, so that the total averages for Monday are increased by this hour with too few vehicles. A summary of the difference in service quality for the case of scenario C can be found in Figure 7.5. We can conclude that there is a high potential in the replacement of bus services by ridepooling during late evening hours if quantitative criteria are used to evaluate the service quality. As discussed previously, a vehicle fleet of 4–7 vehicles would be needed, depending on the weekday, to be able to serve all requests. We note that we do not compare the service costs in this study. The strong improvement in the overall service quality opens the door to an even better and potentially more profitable service, as people might become more willing to use public transport instead of private cars. As ridepooling services generally benefit from a high demand, so that rides can be shared, the results of this study are indeed promising.



(a) Average regret.

(b) Average waiting time.

(c) Average ride time.

(d) Average transportation time.

Figure 7.5: Comparison of dial-a-ride (red) and bus service (blue) for the case of scenario C.

7.5 Summary

In this chapter we investigate the effects on the service quality when replacing buses during late evening hours by a ridepooling service. This is motivated by the observation that particularly late at night, the number of transport requests is usually rather small so that ridepooling services could offer a good alternative to regular bus services. For the purpose of this investigation 30 weeks of simulated transport requests are created from bus trip and ridepooling data sets in the city of Wuppertal in Germany by means of a predictive simulation. The transportation of the simulated requests using the ridepooling

service is mimicked by a rolling-horizon algorithm based on the iterative solution of the dynamic event-based MILP formulation. This algorithm is enhanced by a feasible-path heuristic to cope with request peaks. The computational results show that there is a large potential for improvement of the service quality when offering ridepooling services as compared to bus services: Using the ridepooling service, the average transportation time can be reduced by about 50%. Moreover, an estimation on the number of vehicles needed to provide a comparable service quality is given.

8 Conclusion

In this thesis, we propose a new perspective on modeling DARPs by introducing the concept of the event-based graph. Based on this graph, new (location-augmented-)event-based MILP formulations are presented: the EB, LAEB and ALAEB formulation. These formulations have the advantage that capacity, pairing and precedence constraints which need to be explicitly formulated in location-based models, are implicitly encoded in the event-based graph. While the (location-augmented-)event-based formulations have significantly more variables and constraints than compact location-based models, their superiority is proven computationally and theoretically. In a numerical study we show that the EB formulation strongly outperforms the three-index location-based formulation. Indeed, while the EB model solves benchmark instances with up to 96 users within a couple of seconds, no feasible solution is found within 2 hours using the three-index location-based formulation. Secondly, we prove that both the LAEB and ALAEB formulation have a tighter linear programming relaxation than the LB formulation. In addition, both formulations are tight in the sense that, if time windows fulfill additional conditions (e.g., if they induce a unique ordering of each pair of locations), then the linear programming relaxation yields integer (and hence optimal) solutions. Since the number of variables and constraints in (location-augmented-)event-based formulations strongly depends on the size of the graph, lower and upper bounds on the beginning of service time are computed systematically to remove infeasible nodes and arcs from the graph. In addition, valid inequalities are derived for the EB, LAEB and ALAEB formulation to avoid infeasible paths, customers with incompatibilities and subtours. By using these preprocessing techniques together with the LAEB formulation, extensive numerical experiments on benchmark instances show that computational times are on average reduced by 53.9% compared to the plain EB MILP.

The proposed formulations are compact. Moreover, they can be implemented and solved using standard IP solvers. Although there are approaches for the static DARP which generate solutions even faster (e.g., Gschwind and Irnich, 2015; Rist and Forbes, 2021), these rely on more complex methods such as branch-and-cut and column generation, which cannot be implemented as easily and are less versatile.

To solve the dynamic DARP, a rolling-horizon algorithm is suggested, that iteratively solves a dynamic event-based MILP whenever new requests arrive. Incoming requests can be assigned to optimal insertion positions in 99.5% of all iterations in our computational tests based on *Hol mich! App* data, which is in general not guaranteed by classical two-phase heuristics proposed for the dynamic DARP. The rolling-horizon algorithm is used in a case study investigating the replacement of buses by ridepooling services during the late evening hours. For this purpose artificial ridepooling requests are generated using a predictive simulation calibrated to real-life bus travel data and *Hol mich! App* data. A feasible-path heuristic is suggested to reduce computation times during request peaks.

We conclude that the concept of event-based modeling has advanced the research on the DARP in many ways. The solution of the static DARP has been improved by showing

that location-based formulations can be replaced by (location-augmented-)event-based formulations, which are based on a graph that implicitly contains a lot of information about the problem. Moreover, by solving a plain MILP formulation, results which are competitive with complex algorithms such as branch-and-cut can be produced, thus facilitating the implementation of a high-quality solution strategy for the DARP. The dynamic DARP, which has previously been solved using two-phase heuristics, is solved by an iterative algorithm for the first time, which, in the large majority of iterations, makes the re-optimization of vehicle routes after a request's insertion superfluous. Furthermore, we have used our methods to investigate the potential of ridepooling compared to line-based public transport during periods of low demand.

A limitation of the event-based approach is that it does not immediately transfer to applications with heterogeneous vehicles. Although the use of, e.g., a mixed fleet with vehicle capacities of three and six could be implemented by using a duplicate of the graph, for $Q = 3$ and for $Q = 6$, which is connected through the vehicle depot, the integration of further vehicle characteristics seems to be difficult, if we think of applications in, e.g., healthcare, where configurable vehicles are used.

In the future, the idea of the event-based graph could also be used for other vehicle routing problems, where several users or goods are transported simultaneously. For example, the on-demand transportation of requests on a bus line, i.e., a fixed sequence of bus stations, where the request set is partitioned by the direction of travel, and vehicles may only change the direction of travel when they are empty, could be modeled by using one of the (location-augmented-)event-based formulations and additional constraints. Moreover, applications which guarantee (partially) cycle-free graph structures could be investigated. For example, there are practical settings like the case where all customers have the same pickup or delivery location as in the transition to public transport. Together with our results on the integrality of the LAEB formulation in case of a unique ordering of locations, this could be a promising area of application.

Nomenclature

0	Vehicle depot, sometimes also $0^+, 0^-$
A	Arc set of the event-based graph
$\mathcal{A}(\tau)$	Set of active requests for subproblem DARP(τ)
$A^{\text{fixed}}(\tau)$	Set of fixed arcs corresponding to DARP(τ)
$A^{\text{new}}(\tau)$	Set of arcs that have not been contained in the arc set of the last subproblem
$A^{\text{realized}}(\tau)$	Set of realized arcs corresponding to DARP(τ)
B_v	Continuous variable indicating the beginning of service at node v
B_v^{old}	Value of variables B_v obtained from last subproblem solved
B_v^{LB}	Lower bound for B_v
B_v^{UB}	Upper bound for B_v
$\bar{B}_i^{UB}$	$:= \max_{v \in V_{i^+}} \{B_v^{UB}\}$, i.e., upper bound on the beginning of service at location i^+
B_{vw}^{LB}	Lower bound on the beginning of service at node w when $x_{(v,w)} = 1$
B_{vw}^{UB}	Upper bound on the beginning of service at node v when $x_{(v,w)} = 1$
$\bar{B}_j$	Continuous variable indicating the beginning of service at location j
c_a	Routing costs associated with arc a
$\bar{c}_{ij}$	Routing costs from location i to location j
D	$:= \{1^-, \ldots, n^-\}$, set of delivery locations
d_i	Continuous variable indicating regret of user i w.r.t. e_{i^-}
$d_{\max}$	Continuous variable indicating maximum regret
$\mathcal{D}(\tau)$, $\mathcal{P}(\tau)$, $\mathcal{R}(\tau)$, $\mathcal{S}(\tau)$	Subsets of R of dropped off, picked up, rejected and scheduled requests up to time τ

$[e_j, \ell_j]$	Time window associated with location j
$f^1_{i,j}$, $f^2_{i,j}$	Feasibility of the paths $j^+ \to i^+ \to j^- \to i^-$ and $j^+ \to i^+ \to i^- \to j^-$, respectively, w.r.t. ride time and time window constraints
$G(\tau) = (V(\tau), A(\tau))$	Event-based graph corresponding to subproblem DARP(τ)
$\mathcal{I}_j$	Indicator variable yielding 1 if $j \in P$ and -1 else
i^+, i^-	Pickup and delivery location of request i
J	$\coloneqq P \cup D \cup \{0\}$, set of locations
$\bar{J}$	$\coloneqq P \cup D \cup \{0^+, 0^-\}$, set of locations with start and end depot
K	Number of vehicles
L_i	Maximum ride time associated with request i
$\mathcal{N}(\tau)$	New requests revealed at $\tau - \Delta$
n	Number of transport requests
P	$\coloneqq \{1^+, \ldots, n^+\}$, set of pickup locations
p_i	Binary variable indicating the acceptance of request i
Q	Vehicle capacity
Q_j	Continuous variable indicating the vehicle load after visiting location j
q_i (or q_{i^+}, q_{i^-})	Number of request seats associated with request i (or location i^+, i^-)
R	$\coloneqq \{1, \ldots, n\}$, set of transport requests
s_i (or s_{i^+}, s_{i^-})	Service duration associated with request i (or location i^+, i^-)
$\bar{S}$	Sequence of events $\bar{S} = \{(u^1, u^2), \ldots, (u^{m-1}, u^m)\}$, $m \in \mathbb{N}$ arbitrary but fixed
T	Duration of service
TW	Fixed length of time window constructed from desired pickup or desired delivery time
t_a	Travel time on arc a

$\bar{t}_i$	Travel time for a direct ride from pickup to delivery location of request i
$\bar{t}_{ij}$	Travel time from location i to location j
$u^m_{i^+}$	Unique event s.t. $(u^m_{i^+})_1 = i^+$ and all users sitting in the vehicle directly after event u^m are in $u^m_{i^+}$ in the vehicle too
$u^m_{i^-}$	Unique event s.t. $(u^m_{i^-})_1 = i^-$ and all users sitting in the vehicle directly before event u^m are in $u^m_{i^-}$ in the vehicle too
V	Node set of the event-based graph
V_0	$:= \{(0, \ldots, 0)\}$, set consisting of depot-node
V_{i^+}	Set of pickup nodes associated with request i
V_{i^-}	Set of delivery nodes associated with request i
$V_{i^+}(\tau)$, $V_{i^-}(\tau)$	Set of pickup nodes and set of delivery nodes corresponding to request i and DARP(τ)
$V_{\mathcal{A}(\tau)}$	Set of nodes corresponding to active requests $\mathcal{A}(\tau)$ and DARP(τ)
$V^{\text{realized}}_{\mathcal{D}(\tau)}$, $V^{\text{realized}}_{\mathcal{P}(\tau)}$	Set of realized delivery nodes and set of realized pickup nodes corresponding to DARP(τ)
$V^{\text{l-realized}}(\tau)$	Set of last realized nodes corresponding to DARP(τ)
x_a	Binary variable indicating the use of arc a
$\bar{x}_{ij}$	Binary variable indicating the use of arc (i, j)
x^{old}_a	Value of variables x_a obtained from last subproblem solved
Γ_i	Pickup time communicated to user i
γ	Maximum delay of communicated pickup time
Δ	Time allowed to communicate an answer to new requestst
δ	Timer in minutes to measure time while executing Algorithm 3
$\delta^{\text{in}}(v)$	Set of incoming arcs of node v
$\delta^{\text{out}}(v)$	Set of outgoing arcs of node v
$\delta^{\text{in}}(v, \tau)$	Ingoing arcs of node v corresponding to DARP(τ)

$\delta^{\text{out}}(v, \tau)$	Outgoing arcs of node v corresponding to DARP(τ)
ρ_{abs}	Minimum absolute number of feasible paths allowed
ρ_{rel}	Mininum percentage of feasible paths allowed
ρ_i	Number of feasible paths associated with request i
τ	Current time
$\tau_i - \Delta$	Time at which request i is revealed
ω_1, ω_2, ω_3	Weighting parameters

Bibliography

Ahuja, R. K., T. L. Magnanti, and J. B. Orlin (1993). *Network Flows. Theory, Algorithms, and Applications*. Prentice Hall.

Anair, D., J. Martin, M. C. P. de Moura, and J. Goldman (2020). "Ride-hailing's climate risks: Steering a growing industry toward a clean transportation future". *Union of Concerned Scientists*. Accessed on December 4, 2023. URL: https://www.ucsusa.org/resources/ride-hailing-climate-risks.

Asatryan, H., D. Gaul, H. Gottschalk, K. Klamroth, and M. Stiglmayr (2023). "Ridepooling and public bus services: A comparative case-study". *Submitted to EURO Journal on Transportation and Logistics*. DOI: 10.48550/ARXIV.2302.01709.

Ascheuer, N., M. Fischetti, and M. Grötschel (2000). "A polyhedral study of the asymmetric traveling salesman problem with time windows". In: *Networks* 36.2, pp. 69–79. DOI: https://doi.org/10.1002/1097-0037(200009)36:2<69::AID-NET1>3.0.CO;2-Q.

Atahran, A., C. Lenté, and V. T'kindt (2014). "A multicriteria dial-a-ride problem with an ecological measure and heterogeneous vehicles". In: *Journal of Multi-Criteria Decision Analysis* 21.5-6, pp. 279–298. DOI: 10.1002/mcda.1518.

Attanasio, A., J.-F. Cordeau, G. Ghiani, and G. Laporte (2004). "Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem". In: *Parallel Computing* 30.3, pp. 377–387. DOI: 10.1016/j.parco.2003.12.001.

Balas, E. and N. Simonetti (2001). "Linear time dynamic-programming algorithms for new classes of restricted TSPs: A computational study". In: *INFORMS Journal on Computing* 13.1, pp. 56–75. DOI: 10.1287/ijoc.13.1.56.9748.

Beaudry, A., G. Laporte, T. Melo, and S. Nickel (2008). "Dynamic transportation of patients in hospitals". In: *OR Spectrum* 32.1, pp. 77–107. DOI: 10.1007/s00291-008-0135-6.

Belhaiza, S. (2017). "A data driven hybrid heuristic for the dial-a-ride problem with time windows". In: *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, pp. 1–8. DOI: 10.1109/ssci.2017.8285366.

Belhaiza, S. (2019). "A hybrid adaptive large neighborhood heuristic for a real-life dial-a-ride problem". In: *Algorithms* 12.2, p. 39. DOI: 10.3390/a12020039.

Berbeglia, G., J.-F. Cordeau, I. Gribkovskaia, and G. Laporte (2007). "Static pickup and delivery problems: A classification scheme and survey". In: *TOP* 15.1, pp. 1–31. DOI: 10.1007/s11750-007-0009-0.

Berbeglia, G., J.-F. Cordeau, and G. Laporte (2010). "Dynamic pickup and delivery problems". In: *European Journal of Operational Research* 202.1, pp. 8–15. DOI: 10.1016/j.ejor.2009.04.024.

Berbeglia, G., J.-F. Cordeau, and G. Laporte (2012). "A hybrid tabu search and constraint programming algorithm for the dynamic dial-a-ride problem". In: *INFORMS Journal on Computing* 24.3, pp. 343–355. DOI: 10.1287/ijoc.1110.0454.

Bertsimas, D., P. Jaillet, and S. Martin (2019). "Online vehicle routing: The edge of optimization in large-scale applications". In: *Operations Research* 67.1, pp. 143–162. DOI: 10.1287/opre.2018.1763.

Bischoff, J., K. Führer, and M. Maciejewski (2019). "Impact assessment of autonomous DRT systems". In: *Transportation Research Procedia* 41, pp. 440–446. DOI: 10.1016/j.trpro.2019.09.074.

Bischoff, J., M. Maciejewski, and K. Nagel (2017). "City-wide shared taxis: A simulation study in Berlin". In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, pp. 275–280. DOI: 10.1109/itsc.2017.8317926.

Bongiovanni, C., M. Kaspi, and N. Geroliminis (2019). "The electric autonomous dial-a-ride problem". In: *Transportation Research Part B: Methodological* 122, pp. 436–456. DOI: 10.1016/j.trb.2019.03.004.

Braekers, K., A. Caris, and G. K. Janssens (2014). "Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots". In: *Transportation Research Part B: Methodological* 67, pp. 166–186. DOI: 10.1016/j.trb.2014.05.007.

Braekers, K., K. Ramaekers, and I. V. Nieuwenhuyse (2016). "The vehicle routing problem: State of the art classification and review". In: *Computers & Industrial Engineering* 99, pp. 300–313. DOI: 10.1016/j.cie.2015.12.007.

Carotenuto, P. and F. Martis (2017). "A double dynamic fast algorithm to solve multi-vehicle dial-a-ride problem". In: *Transportation Research Procedia* 27, pp. 632–639. DOI: 10.1016/j.trpro.2017.12.131.

Chevrier, R., A. Liefooghe, L. Jourdan, and C. Dhaenens (2012). "Solving a dial-a-ride problem with a hybrid evolutionary multi-objective approach: Application to demand responsive transport". In: *Applied Soft Computing* 12.4, pp. 1247–1258. DOI: 10.1016/j.asoc.2011.12.014.

Colorni, A. and G. Righini (2001). "Modeling and optimizing dynamic dial-a-ride problems". In: *International Transactions in Operational Research* 8.2, pp. 155–166. DOI: 10.1111/1475-3995.00256.

Conforti, M., G. Cornuéjols, and G. Zambelli (2014). *Integer Programming*. Springer International Publishing. DOI: 10.1007/978-3-319-11008-0.

Cook, S. A. (1971). "The complexity of theorem-proving procedures". In: *Proceedings of the third annual ACM symposium on theory of computing - STOC '71*. ACM Press, pp. 151–158. DOI: 10.1145/800157.805047.

Cordeau, J.-F. (2006). "A branch-and-but algorithm for the dial-a-ride problem". In: *Operations Research* 54.3, pp. 573–586. DOI: 10.1287/opre.1060.0283.

Cordeau, J.-F. and G. Laporte (2003). "A tabu search heuristic for the static multi-vehicle dial-a-ride problem". In: *Transportation Research Part B: Methodological* 37.6, pp. 579–594. DOI: 10.1016/s0191-2615(02)00045-0.

Cordeau, J.-F. and G. Laporte (2007). "The dial-a-ride problem: Models and algorithms". In: *Annals of Operations Research* 153.1, pp. 29–46. DOI: 10.1007/s10479-007-0170-8.

Cordeau, J.-F., G. Laporte, M. W. Savelsbergh, and D. Vigo (2007). "Vehicle routing". In: *Transportation*. Ed. by C. Barnhart and G. Laporte. Vol. 14. Handbooks in Operations Research and Management Science. Elsevier, pp. 367–428. DOI: https://doi.org/10.1016/S0927-0507(06)14006-2.

Cortés, C. E., M. Matamala, and C. Contardo (2010). "The pickup and delivery problem with transfers: Formulation and a branch-and-cut solution method". In: *European Journal of Operational Research* 200.3, pp. 711–724. DOI: 10.1016/j.ejor.2009.01.022.

Coslovich, L., R. Pesenti, and W. Ukovich (2006). "A two-phase insertion technique of unexpected customers for a dynamic dial-a-ride problem". In: *European Journal of Operational Research* 175.3, pp. 1605–1615. DOI: 10.1016/j.ejor.2005.02.038.

Cubillos, C., E. Urra, and N. Rodríguez (2009). "Application of genetic algorithms for the DARPTW problem". In: *International Journal of Computers Communications & Control* 4.2, p. 127. DOI: 10.15837/ijccc.2009.2.2420.

Dandl, F., B. Bracher, and K. Bogenberger (2017). "Microsimulation of an autonomous taxi-system in Munich". In: *2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pp. 833–838. URL: https://api.semanticscholar.org/CorpusID:25700161.

Dantzig, G., R. Fulkerson, and S. Johnson (1954). "Solution of a large-scale traveling-salesman problem". In: *Journal of the Operations Research Society of America* 2.4, pp. 393–410. DOI: 10.1287/opre.2.4.393.

Dantzig, G. B. and J. H. Ramser (1959). "The truck dispatching problem". In: *Management Science* 6.1, pp. 80–91. DOI: 10.1287/mnsc.6.1.80.

Dantzig, G. (1963). *Linear Programming and Extensions*. Princeton University Press. DOI: 10.1515/9781400884179.

Das, I. and J. E. Dennis (1997). "A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems". In: *Structural Optimization* 14.1, pp. 63–69. DOI: 10.1007/bf01197559.

Desrochers, M. and G. Laporte (1991). "Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints". In: *Operations Research Letters* 10.1, pp. 27–36. DOI: 10.1016/0167-6377(91)90083-2.

Desrosiers, J., Y. Dumas, F. Soumis, S. Taillefer, and D. Villeneuve (1991). "An algorithm for mini-clustering in handicapped transport". In: *Les Cahiers du GERAD, G-91-02, HEC Montréal*.

Detti, P., F. Papalini, and G. Z. M. de Lara (2017). "A multi-depot dial-a-ride problem with heterogeneous vehicles and compatibility constraints in healthcare". In: *Omega* 70, pp. 1–14. DOI: 10.1016/j.omega.2016.08.008.

Dumas, Y., J. Desrosiers, and F. Soumis (1989). "Large scale multi-vehicle dial-a-ride problems". In: *Les Cahiers du GERAD, G-89-30, HEC Montréal*.

Dumez, D., C. Tilk, S. Irnich, F. Lehuédé, and O. Péton (2021). "Hybridizing large neighborhood search and exact methods for generalized vehicle routing problems with time windows". In: *EURO Journal on Transportation and Logistics* 10, p. 100040. DOI: 10.1016/j.ejtl.2021.100040.

Ehrgott, M. (2005). *Multicriteria Optimization*. Springer. DOI: 10.1007/3-540-27659-9.

Faraway, J. J. (2016). *Extending the Linear Model with R*. Chapman & Hall/CRC Texts in Statistical Science Series. Boca Raton: CRC Press. DOI: 10.1201/9781315382722.

Fischetti, M., F. Glover, and A. Lodi (2005). "The feasibility pump". In: *Mathematical Programming* 104.1, pp. 91–104. DOI: 10.1007/s10107-004-0570-3.

Foljanty, L. (2020). "Mapping the global on-demand ridepooling market". Accessed on December 4, 2023. URL: https://lukas-foljanty.medium.com/mapping-the-global-on-demand-ridepooling-market-f8318de1c030.

Furtado, M. G. S., P. Munari, and R. Morabito (2017). "Pickup and delivery problem with time windows: A new compact two-index formulation". In: *Operations Research Letters* 45.4, pp. 334–341. DOI: 10.1016/j.orl.2017.04.013.

Garey, M. R. and D. S. Johnson (1979). *Computers and Intractability. A Guide to the Theory of NP-Completeness*. W. H. Freeman.

Gaul, D. (2022a). "Event-based MILP for the DARP". Git repository. URL: https://git.uni-wuppertal.de/dgaul/event-based-milp-for-darp.

Gaul, D. (2022b). "Rolling-horizon algorithm for the dynamic DARP". Git repository. URL: https://git.uni-wuppertal.de/dgaul/rolling-horizon-algorithm-for-dynamic-darp.

Gaul, D. (2023). "A tight formulation for the DARP". Git repository. URL: https://git.uni-wuppertal.de/dgaul/a-tight-formulation-for-the-darp.

Gaul, D., K. Klamroth, C. Pfeiffer, A. Schulz, and M. Stiglmayr (2023). "A tight formulation for the dial-a-ride problem". *Submitted to European Journal of Operational Research*. DOI: 10.48550/ARXIV.2308.11285.

Gaul, D., K. Klamroth, and M. Stiglmayr (2021). "Solving the dynamic dial-a-ride problem using a rolling-horizon event-based graph". In: *21st Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2021)*. Ed. by M. Müller-Hannemann and F. Perea. Vol. 96. Open Access Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 8:1–8:16. DOI: 10.4230/OASIcs.ATMOS.2021.8.

Gaul, D., K. Klamroth, and M. Stiglmayr (2022). "Event-based MILP models for ridepooling applications". In: *European Journal of Operational Research* 301.3, pp. 1048–1063. DOI: 10.1016/j.ejor.2021.11.053.

Glover, F. and C. McMillan (1986). "The general employee scheduling problem: An integration of MS and AI". In: *Computers & Operations Research* 13.5, 563–573. DOI: 10.1016/0305-0548(86)90050-X.

Gschwind, T. and M. Drexl (2019). "Adaptive large neighborhood search with a constant-time feasibility test for the dial-a-ride problem". In: *Transportation Science* 53.2, pp. 480–491. DOI: 10.1287/trsc.2018.0837.

Gschwind, T. and S. Irnich (2015). "Effective handling of dynamic time windows and its application to solving the dial-a-ride problem". In: *Transportation Science* 49.2, pp. 335–354. DOI: 10.1287/trsc.2014.0531.

Guerreiro, P. M. M., P. J. S. Cardoso, and H. C. L. Fernandes (2020). "A comparison of multiple objective algorithms in the context of a dial-a-ride problem". In: *Lecture Notes in Computer Science*. Springer International Publishing, pp. 382–396. DOI: 10.1007/978-3-030-50436-6_28.

Guerriero, F., M. E. Bruni, and F. Greco (2013). "A hybrid greedy randomized adaptive search heuristic to solve the dial-a-ride problem". In: *Asia-Pacific Journal of Operational Research* 30.1, p. 1250046. DOI: 10.1142/s0217595912500467.

Guo, S., I. Dayarian, J. Li, and X. Qian (2022). "A branch-cut-and-price algorithm for a dial-a-ride problem with minimum disease-transmission risk". DOI: 10.48550/ARXIV.2205.05324.

Hanne, T., T. Melo, and S. Nickel (2009). "Bringing robustness to patient flow management through optimized patient transports in hospitals". In: *Interfaces* 39.3, pp. 241–255. DOI: 10.1287/inte.1080.0379.

Hastie, T., R. Tibshirani, J. H. Friedman, and J. H. Friedman (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Vol. 2. Springer New York. DOI: 10.1007/978-0-387-84858-7.

Ho, S. C., W. Szeto, Y.-H. Kuo, J. M. Leung, M. Petering, and T. W. Tou (2018). "A survey of dial-a-ride problems: Literature review and recent developments". In: *Transportation Research Part B: Methodological* 111, pp. 395–421. DOI: 10.1016/j.trb.2018.02.001.

Horni, A., K. Nagel, and K. W. Axhausen (2016). *The Multi-Agent Transport Simulation MATSim*. Ubiquity Press. DOI: 10.5334/baw.

Hu, T.-Y. and C.-P. Chang (2014). "A revised branch-and-price algorithm for dial-a-ride problems with the consideration of time-dependent travel cost". In: *Journal of Advanced Transportation* 49.6, pp. 700–723. DOI: 10.1002/atr.1296.

Hu, T.-Y., G.-C. Zheng, and T.-Y. Liao (2019). "Multi-objective model for dial-a-ride problems with vehicle speed considerations". In: *Transportation Research Record: Journal of the Transportation Research Board* 2673.11, pp. 161–171. DOI: 10.1177/0361198119848417.

Hungerländer, P., K. Maier, V. Pachatz, and C. Truden (2021). "Improving sharing rates of a dial-a-ride problem implemented for an austrian mobility provider". In: *Transportation Research Procedia* 52, pp. 525–532. DOI: 10.1016/j.trpro.2021.01.062.

Häll, C. H., M. Högberg, and J. T. Lundgren (2012). "A modeling system for simulation of dial-a-ride services". In: *Public Transport* 4.1, pp. 17–37. DOI: 10.1007/s12469-012-0052-6.

Häll, C. H., J. T. Lundgren, and S. Voß (2015). "Evaluating the performance of a dial-a-ride service using simulation". In: *Public Transport* 7.2, pp. 139–157. DOI: 10.1007/s12469-015-0101-z.

Häll, C. H. and A. Peterson (2013). "Improving paratransit scheduling using ruin and recreate methods". In: *Transportation Planning and Technology* 36.4, pp. 377–393. DOI: 10.1080/03081060.2013.798488.

Häme, L. and H. Hakula (2015). "A maximum cluster algorithm for checking the feasibility of dial-a-ride instances". In: *Transportation Science* 49.2, pp. 295–310. DOI: 10.1287/trsc.2013.0495.

Ioachim, I., J. Desrosiers, Y. Dumas, M. M. Solomon, and D. Villeneuve (1995). "A request clustering algorithm for door-to-door handicapped transportation". In: *Transportation Science* 29.1, pp. 63–78. DOI: 10.1287/trsc.29.1.63.

Irnich, S. and G. Desaulniers (2005). "Shortest path problems with resource constraints". In: *Column Generation*. Springer-Verlag, pp. 33–65. DOI: 10.1007/0-387-25486-2_2.

ITF (2015). "Urban mobility system upgrade. How shared self-driving cars could change city traffic." In: *International Transport Forum Policy Papers* 6. DOI: 10.1787/5jlwvzdk29g5-en.

Jaw, J.-J., A. R. Odoni, H. N. Psaraftis, and N. H. Wilson (1986). "A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows". In: *Transportation Research Part B: Methodological* 20.3, pp. 243–257. DOI: 10.1016/0191-2615(86)90020-2.

Johnsen, L. C. and F. Meisel (2022). "Interrelated trips in the rural dial-a-ride problem with autonomous vehicles". In: *European Journal of Operational Research* 303.1, pp. 201–219. DOI: 10.1016/j.ejor.2022.02.021.

Jorgensen, R. M., J Larsen, and K. B. Bergvinsdottir (2007). "Solving the dial-a-ride problem using genetic algorithms". In: *Journal of the Operational Research Society* 58.10, pp. 1321–1331. DOI: 10.1057/palgrave.jors.2602287.

Kirchler, D. and R. Wolfler Calvo (2013). "A granular tabu search algorithm for the dial-a-ride problem". In: *Transportation Research Part B: Methodological* 56, pp. 120–135. DOI: 10.1016/j.trb.2013.07.014.

Knight, K. (1999). *Mathematical Statistics*. New York: Chapman & Hall/CRC. DOI: 10.1201/9780367805319.

Lenstra, J. K. and A. H. G. R. Kan (1981). "Complexity of vehicle routing and scheduling problems". In: *Networks* 11.2, pp. 221–227. DOI: 10.1002/net.3230110211.

Letchford, A. N. and J.-J. Salazar-González (2005). "Projection results for vehicle routing". In: *Mathematical Programming* 105.2-3, pp. 251–274. DOI: 10.1007/s10107-005-0652-x.

Liebchen, C., M. Lehnert, C. Mehlert, and M. Schiefelbusch (2021). "Betriebliche Effizienzgrößen für Ridepooling-Systeme". In: *Making Connected Mobility Work*. Springer Fachmedien Wiesbaden, pp. 135–150. DOI: 10.1007/978-3-658-32266-3_7.

Liu, M., Z. Luo, and A. Lim (2015). "A branch-and-cut algorithm for a realistic dial-a-ride problem". In: *Transportation Research Part B: Methodological* 81, pp. 267–288. DOI: 10.1016/j.trb.2015.05.009.

Lois, A. and A. Ziliaskopoulos (2017). "Online algorithm for dynamic dial a ride problem and its metrics". In: *Transportation Research Procedia* 24, pp. 377–384. DOI: 10.1016/j.trpro.2017.05.097.

Lourenço, H. R., O. C. Martin, and T. Stützle (2018). "Iterated local search: Framework and applications". In: *International Series in Operations Research & Management Science*. Springer International Publishing, pp. 129–168. DOI: 10.1007/978-3-319-91086-4_5.

Luo, Y. and P. Schonfeld (2011). "Online rejected-reinsertion heuristics for dynamic multi-vehicle dial-a-ride problem". In: *Transportation Research Record: Journal of the Transportation Research Board* 2218.1, pp. 59–67. DOI: 10.3141/2218-07.

Luo, Z., M. Liu, and A. Lim (2019). "A two-phase branch-and-price-and-cut for a dial-a-ride problem in patient transportation". In: *Transportation Science* 53.1, pp. 113–130. DOI: 10.1287/trsc.2017.0772.

Lysgaard, J. (2006). "Reachability cuts for the vehicle routing problem with time windows". In: *European Journal of Operational Research* 175.1, pp. 210–223. DOI: 10.1016/j.ejor.2005.04.022.

Madsen, O. B. G., H. F. Ravn, and J. M. Rygaard (1995). "A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities, and multiple objectives". In: *Annals of Operations Research* 60.1, pp. 193–208. DOI: 10.1007/bf02031946.

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

Marković, N., R. Nair, P. Schonfeld, E. Miller-Hooks, and M. Mohebbi (2015). "Optimizing dial-a-ride services in Maryland: Benefits of computerized routing and scheduling". In: *Transportation Research Part C: Emerging Technologies* 55, pp. 156–165. DOI: 10.1016/j.trc.2015.01.011.

Masmoudi, M. A., K. Braekers, M. Masmoudi, and A. Dammak (2017). "A hybrid genetic algorithm for the heterogeneous dial-a-ride problem". In: *Computers & Operations Research* 81, pp. 1–13. DOI: 10.1016/j.cor.2016.12.008.

Masmoudi, M. A., M. Hosny, E. Demir, K. N. Genikomsakis, and N. Cheikhrouhou (2018). "The dial-a-ride problem with electric vehicles and battery swapping stations". In: *Transportation Research Part E: Logistics and Transportation Review* 118, pp. 392–420. DOI: 10.1016/j.tre.2018.08.005.

Masmoudi, M. A., M. Hosny, E. Demir, and E. Pesch (2019). "Hybrid adaptive large neighborhood search algorithm for the mixed fleet heterogeneous dial-a-ride problem". In: *Journal of Heuristics* 26.1, pp. 83–118. DOI: 10.1007/s10732-019-09424-x.

Masson, R., F. Lehuédé, and O. Péton (2013). "An adaptive large neighborhood search for the pickup and delivery problem with transfers". In: *Transportation Science* 47.3, pp. 344–355. DOI: 10.1287/trsc.1120.0432.

Masson, R., F. Lehuédé, and O. Péton (2014). "The dial-a-ride problem with transfers". In: *Computers & Operations Research* 41, pp. 12–23. DOI: 10.1016/j.cor.2013.07.020.

Mauri, G., L. Antonio, and N. Lorena (2009). "Customers' satisfaction in a dial-a-ride problem". In: *IEEE Intelligent Transportation Systems Magazine* 1.3, pp. 6–14. DOI: 10.1109/mits.2009.934641.

Melachrinoudis, E., A. B. Ilhan, and H. Min (2007). "A dial-a-ride problem for client transportation in a health-care organization". In: *Computers & Operations Research* 34.3, pp. 742–759. DOI: 10.1016/j.cor.2005.03.024.

Mladenović, N. and P. Hansen (1997). "Variable neighborhood search". In: *Computers & Operations Research* 24.11, pp. 1097–1100. DOI: 10.1016/s0305-0548(97)00031-2.

Molenbruch, Y., K. Braekers, and A. Caris (2017a). "Typology and literature review for dial-a-ride problems". In: *Annals of Operations Research* 259.1-2, pp. 295–325. DOI: 10.1007/s10479-017-2525-0.

Molenbruch, Y., K. Braekers, A. Caris, and G. V. Berghe (2017b). "Multi-directional local search for a bi-objective dial-a-ride problem in patient transportation". In: *Computers & Operations Research* 77, pp. 58–71. DOI: 10.1016/j.cor.2016.07.020.

Molenbruch, Y., K. Braekers, P. Hirsch, and M. Oberscheider (2021). "Analyzing the benefits of an integrated mobility system using a matheuristic routing algorithm". In: *European Journal of Operational Research* 290.1, pp. 81–98. DOI: 10.1016/j.ejor.2020.07.060.

Morapitiye, S. and T. Kis (2022). "Strong cuts from compatibility relations for the dial-a-ride problem". In: *Discrete Applied Mathematics* 309, pp. 240–257. DOI: 10.1016/j.dam.2021.12.010.

Morrison, D. R., S. H. Jacobson, J. J. Sauppe, and E. C. Sewell (2016). "Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning". In: *Discrete Optimization* 19, pp. 79–102. DOI: 10.1016/j.disopt.2016.01.005.

Muelas, S., A. LaTorre, and J.-M. Peña (2013). "A variable neighborhood search algorithm for the optimization of a dial-a-ride problem in a large city". In: *Expert Systems with Applications* 40.14, pp. 5516–5531. DOI: 10.1016/j.eswa.2013.04.015.

Muelas, S., A. LaTorre, and J.-M. Peña (2015). "A distributed VNS algorithm for optimizing dial-a-ride problems in large-scale scenarios". In: *Transportation Research Part C: Emerging Technologies* 54, pp. 110–130. DOI: 10.1016/j.trc.2015.02.024.

Narici, L. and E. Beckenstein (1985). *Topological Vector Spaces*. Monographs and Textbooks in Pure and Applied Mathematics. M. Dekker.

Nemhauser, G. and L. Wolsey (1988). *Integer and Combinatorial Optimization*. Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley.

Papadimitriou, C. H. and K. Steiglitz (1998). *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications.

Paquette, J., J.-F. Cordeau, G. Laporte, and M. M. Pascoal (2013). "Combining multicriteria analysis and tabu search for dial-a-ride problems". In: *Transportation Research Part B: Methodological* 52, pp. 1–16. DOI: 10.1016/j.trb.2013.02.007.

Parragh, S. N. (2011). "Introducing heterogeneous users and vehicles into models and algorithms for the dial-a-ride problem". In: *Transportation Research Part C: Emerging Technologies* 19.5, pp. 912–930. DOI: 10.1016/j.trc.2010.06.002.

Parragh, S. N., J.-F. Cordeau, K. F. Doerner, and R. F. Hartl (2010a). "Models and algorithms for the heterogeneous dial-a-ride problem with driver-related constraints". In: *OR Spectrum* 34.3, pp. 593–633. DOI: 10.1007/s00291-010-0229-9.

Parragh, S. N., K. F. Doerner, and R. F. Hartl (2010b). "Variable neighborhood search for the dial-a-ride problem". In: *Computers & Operations Research* 37.6, pp. 1129–1138. DOI: 10.1016/j.cor.2009.10.003.

Parragh, S. N., K. F. Doerner, R. F. Hartl, and X. Gandibleux (2009). "A heuristic two-phase solution approach for the multi-objective dial-a-ride problem". In: *Networks* 54.4, pp. 227–242. DOI: 10.1002/net.20335.

Parragh, S. N. and V. Schmid (2013). "Hybrid column generation and large neighborhood search for the dial-a-ride problem". In: *Computers & Operations Research* 40.1, pp. 490–497. DOI: 10.1016/j.cor.2012.08.004.

Parragh, S. N., J. P. de Sousa, and B. Almada-Lobo (2015). "The dial-a-ride problem with split requests and profits". In: *Transportation Science* 49.2, pp. 311–334. DOI: 10.1287/trsc.2014.0520.

Pfeiffer, C. and A. Schulz (2021). "An ALNS algorithm for the static dial-a-ride problem with ride and waiting time minimization". In: *OR Spectrum* 44.1, pp. 87–119. DOI: 10.1007/s00291-021-00656-7.

Posada, M. and C. H. Häll (2020). "A metaheuristic for evaluation of an integrated special transport service". In: *International Journal of Urban Sciences* 24.3, pp. 316–338. DOI: 10.1080/12265934.2019.1709533.

Pouls, M. (2023). "Real-Time Optimization for Dynamic Ride-Sharing". PhD thesis. Karlsruher Institut für Technologie (KIT). DOI: 10.5445/IR/1000158636.

Psaraftis, H. N. (1980). "A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem". In: *Transportation Science* 14.2, pp. 130–154. DOI: 10.1287/trsc.14.2.130.

Qu, Y. and J. F. Bard (2015). "A branch-and-price-and-cut algorithm for heterogeneous pickup and delivery problems with configurable vehicle capacity". In: *Transportation Science* 49.2, pp. 254–270. DOI: 10.1287/trsc.2014.0524.

Quadrifoglio, L., M. M. Dessouky, and F. Ordóñez (2008). "A simulation study of demand responsive transit system design". In: *Transportation Research Part A: Policy and Practice* 42.4, pp. 718–737. DOI: 10.1016/j.tra.2008.01.018.

Reinhardt, L. B., T. Clausen, and D. Pisinger (2013). "Synchronized dial-a-ride transportation of disabled passengers at airports". In: *European Journal of Operational Research* 225.1, pp. 106–117. DOI: 10.1016/j.ejor.2012.09.008.

Richter, E., M. Friedrich, A. Migl, and J. Hartleb (2019). "Integrating ridesharing services with automated vehicles into macroscopic travel demand models". In: *2019 6th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*. IEEE, pp. 1–8. DOI: 10.1109/mtits.2019.8883315.

Riedler, M. and G. Raidl (2018). "Solving a selective dial-a-ride problem with logic-based Benders decomposition". In: *Computers & Operations Research* 96, pp. 30–54. DOI: 10.1016/j.cor.2018.03.008.

Rist, Y. and M. Forbes (2022). "A column generation and combinatorial Benders decomposition algorithm for the selective dial-a-ride-problem". In: *Computers & Operations Research* 140, p. 105649. DOI: 10.1016/j.cor.2021.105649.

Rist, Y. and M. A. Forbes (2021). "A new formulation for the dial-a-ride problem". In: *Transportation Science* 55.5, pp. 1113–1135. DOI: 10.1287/trsc.2021.1044.

Rockafellar, R. T. (1997). *Convex Analysis*. Princeton University Press.

Ronald, N., R. Thompson, and S. Winter (2015). "Simulating demand-responsive transportation: A review of agent-based approaches". In: *Transport Reviews* 35.4, pp. 404–421. DOI: 10.1080/01441647.2015.1017749.

Ropke, S. and J.-F. Cordeau (2009). "Branch and cut and price for the pickup and delivery problem with time windows". In: *Transportation Science* 43.3, pp. 267–286. DOI: 10.1287/trsc.1090.0272.

Ropke, S., J.-F. Cordeau, and G. Laporte (2007). "Models and branch-and-cut algorithms for pickup and delivery problems with time windows". In: *Networks* 49.4, pp. 258–272. DOI: 10.1002/net.20177.

Ropke, S. and D. Pisinger (2006). "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows". In: *Transportation Science* 40.4, pp. 455–472. DOI: 10.1287/trsc.1050.0135.

Rothvoß, T. (2012). "Some 0/1 polytopes need exponential size extended formulations". In: *Mathematical Programming* 142.1-2, pp. 255–268. DOI: 10.1007/s10107-012-0574-3.

Santos, D. O. and E. C. Xavier (2015). "Taxi and ride sharing: A dynamic dial-a-ride problem with money as an incentive". In: *Expert Systems with Applications* 42.19, pp. 6728–6737. DOI: 10.1016/j.eswa.2015.04.060.

Schrijver, A. (1998). *Theory of Linear and Integer Programming*. Wiley.

Schulz, A. and C. Pfeiffer (2024). "Using fixed paths to improve branch-and-cut algorithms for precedence-constrained routing problems". In: *European Journal of Operational Research* 312.2, pp. 456–472. DOI: 10.1016/j.ejor.2023.07.002.

Shaw, P. (1998). "Using constraint programming and local search methods to solve vehicle routing problems". In: *Principles and Practice of Constraint Programming — CP98*. Springer Berlin Heidelberg, pp. 417–431. DOI: 10.1007/3-540-49481-2_30.

Souza, A. L. S., M. Bernardo, P. H. V. Penna, J. Pannek, and M. J. F. Souza (2021). "Bi-objective optimization model for the heterogeneous dynamic dial-a-ride problem with no rejects". In: *Optimization Letters* 16.1, pp. 355–374. DOI: 10.1007/s11590-020-01698-6.

Souza, A. L. S., J. B. C. Chagas, P. H. V. Penna, and M. J. F. Souza (2020). "A hybrid heuristic algorithm for the dial-a-ride problem". In: *Variable Neighborhood Search*. Springer International Publishing, pp. 53–66. DOI: 10.1007/978-3-030-44932-2_4.

Su, Y., N. Dupin, and J. Puchinger (2023). "A deterministic annealing local search for the electric autonomous dial-a-ride problem". In: *European Journal of Operational Research* 309.3, pp. 1091–1111. DOI: 10.1016/j.ejor.2023.02.012.

Tellez, O., S. Vercraene, F. Lehuédé, O. Péton, and T. Monteiro (2018). "The fleet size and mix dial-a-ride problem with reconfigurable vehicle capacity". In: *Transportation Research Part C: Emerging Technologies* 91, pp. 99–123. DOI: 10.1016/j.trc.2018.03.020.

Toth, P. and D. Vigo (2014). *Vehicle Routing Problems, Methods, and Applications*. Society for Industrial and Applied Mathematics.

Vallee, S., A. Oulamara, and W. R. Cherif-Khettaf (2020). "New online reinsertion approaches for a dynamic dial-a-ride problem". In: *Journal of Computational Science* 47, p. 101199. DOI: 10.1016/j.jocs.2020.101199.

Viana, R. J. S., A. G. Santos, F. V. C. Martins, and E. F. Wanner (2019). "Optimization of a demand responsive transport service using multi-objective evolutionary algorithms". In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO '19. Association for Computing Machinery, 2064–2067. DOI: 10.1145/3319619.3328528.

Wilkes, G., L. Briem, M. Heilig, T. Hilgert, M. Kagerbauer, and P. Vortisch (2021). "Determining service provider and transport system related effects of ridesourcing services by simulation within the travel demand model mobiTopp". In: *European Transport Research Review* 13.1. DOI: 10.1186/s12544-021-00493-3.

Wolsey, L. A. (1998). *Integer Programming*. Wiley.

Wong, K., A. Han, and C. Yuen (2012). "On dynamic demand responsive transport services with degree of dynamism". In: *Transportmetrica A: Transport Science* 10.1, pp. 55–73. DOI: 10.1080/18128602.2012.694491.

Zidi, I., K. Mesghouni, K. Zidi, and K. Ghedira (2012). "A multi-objective simulated annealing for the multi-criteria dial a ride problem". In: *Engineering Applications of Artificial Intelligence* 25.6, pp. 1121–1131. DOI: 10.1016/j.engappai.2012.03.012.