Rheinland-Pfälzische
Technische Universität
Kaiserslautern
Landau

Essays in Operations Research

Vom Fachbereich Wirtschaftswissenschaften
der Rheinland-Pfälzischen Technischen Universität Kaiserslautern-Landau
zur Verleihung des akademischen Grades Doctor rerum politicarum (Dr. rer. pol.)
genehmigte

D i s s e r t a t i o n

vorgelegt von

Daniel Schermer

Tag der mündlichen Prüfung:	06. Mai 2024
Dekan:	Prof. Dr. Jan Wenzelburger
Vorsitzender:	Prof. Dr. Timo Gschwind
Berichterstatter:	Prof. Dr. Oliver Wendt
	Prof. Dr. Florian Sahling

D 386
(2024)

Contents

List of Abbreviations

ACP Algorithm Configuration Problem

ALNS Adaptive Large Neighborhood Search

ASP Algorithm Selection Problem

B&B Branch-and-Bound

B&C Branch-and-Cut

B&P Branch-and-Price

BKS Best Known Solution

BPC Branch-Price-and-Cut

CART Classification and Regression Trees

CRP Constraint Related Properties

D&C Divide-and-Conquer

DASP Drone Assignment and Scheduling Problem

DP Dynamic Programming

DS Default Solver

DSP Drone Scheduling Problem

DT Decision Tree

EML Empirical Model Learning

FSTSP Flying Sidekick Taveling Salesman Problem

GRASP Greedy Randomized Adaptive Search Procedure

ILP Integer Linear Programming

LB Lower Bound

MILP Mixed-Integer Linear Program

MINLP Mixed-Integer Non-Linear Programming

MIP Mixed-Integer Programming

ML Machine Learning

MTZ Miller-Tucker-Zemlin

OR Operations Research

PDSTSP Parallel Drone Scheduling Traveling Salesman Problem

RF Random Forest

RUB Route Upper Bound

SA Simulated Annealing

SAS Solver found by means of Algorithm Selection

SBS Single-Best Solver

SE String Exchange

SR String Relocation

SRP Size Related Properties

SVR Support Vector Regression

TS Tabu Search

TSDSLP Traveling Salesman Drone Station Location Problem

TSP Traveling Salesman Problem

TSPD Traveling Salesman Problem with Drone

TSPDRS Drone-Assisted Traveling Salesman Problem with Robot Stations

TSPDS Traveling Salesman Problem with Drone Station

VBS Virtual-Best Solver

VDRPTW Vehicle Drone Routing Problem with Time Windows

VIEQ Valid Inequality

VNS Variable Neighborhood Search

VRP Vehicle Routing Problem

VRPD Vehicle Routing Problem with Drones

VRPDERO Vehicle Routing Problem with Drones and En Route Operations

List of Algorithms

List of Figures

List of Tables

1 Background, Organization, and Contribution

Section 1.1 provides the background and motivation for the problems addressed in this thesis. The structure of the following chapters, along with the specific contributions, is outlined in Section 1.2.

1.1 Background

As pointed out by Ghiani et al. (2013), the continued growth of e-commerce and the perceived immediate availability of a wide variety of commodities has molded a very demanding customer. This is manifested by the general expectation of same- or next-day direct-to-consumer-deliveries, which confronts logistics providers with several challenges. They are required to handle more transactions, each typically involving smaller quantities, within shorter timeframes, at reduced costs, and with heightened service levels, all while striving for increased operational efficiency and sustainability. Clearly, last-mile delivery has a crucial role when it comes to addressing these challenges. On the one hand, it accounts for a significant share of the overall logistics costs; and, on the other hand, it influences the perceived customer service level to a great extent.

To address these unique challenges, there has been a strong interest in investigating the potential opportunities that may be seized by logistics providers when adopting emerging technologies (Savelsbergh and Van Woensel, 2016). For instance, with regard to environmental sustainability, prominent studies focus on the reduction of carbon emissions by transitioning to a fleet of vehicles that utilizes more efficient automotive technology. This involves battery electric vehicles as a replacement for traditional, fossil-fueled trucks (see, e. g., (Goeke, 2018; Hof, 2019; Pelletier et al., 2016)). In terms of operational efficiency, addressing the surge in parcel volumes, logistics providers are looking for new ways to maximize their delivery rate. Generally, this performance indicator is largely governed by the customer density and the transport infrastructure along each route. Thus, optimizing the routing without adjusting the operational dynamics of the logistics system might prove futile, particularly when considering external factors like increased traffic congestion.

Nevertheless, changes to the ways in which deliveries are performed may open up new opportunities, and one emerging technology that may radically transform how the last-mile operates is the use of autonomous auxiliary vehicles. Such auxiliary vehicles encompass

wheeled robots (Azadeh et al., 2019; Fragapane et al., 2021) or unmanned aerial vehicles (commonly referred to as drones) (Macrina et al., 2020; Otto et al., 2018). Due to the limited operational range and carrying capacity of such vehicles, it is generally conceived for them to be used either for the purpose of intralogistics or in close conjunction with traditional delivery trucks (vans). In scenarios involving trucks and auxiliary vehicles like drones, the latter can exploit their freedom from road restrictions and congestion to move swiftly between locations. Furthermore, the lightweight nature of auxiliary vehicles and their payloads results in lower energy consumption for transit between points. However, the carrying capacity of drones or (small-sized) robots is inherently limited to just one or few parcels. Moreover, given their reliance on relatively small batteries, their endurance is more restricted compared to a commercial delivery truck. Consequently, due to the complementary nature of trucks and auxiliary vehicles, the integration of the latter into existing logistics systems may yield synergistic benefits.

Undoubtedly, the transportation sector is one of the fields that has benefited significantly from *Operations Research* (OR). To be more precise, the industry's current scale of operation relies extensively on decision support systems, incorporating a great variety of optimization models and algorithms that drive them (see, e. g., (Petropoulos et al., 2023) and references therein). In this context, the *Traveling Salesman Problem* (TSP) and the *Vehicle Routing Problem* (VRP) stand out as two of the most fundamental problems in the scientific literature, and these problems have already been studied for several decades (see, e. g., (Dantzig et al., 1954; Dantzig and Ramser, 1959)). In the general form of the TSP, the problem involves a set of locations and the distance between each pair. The objective is to determine the shortest route that visits each location exactly once before returning to the starting point. In contrast, the VRP is traditionally concerned with routing a fleet of capacitated trucks from a single depot, ensuring all demands are satisfied while minimizing the total cost incurred by the routing decisions. In the decades following Dantzig's seminal works, numerous variants of these problems have been proposed. Among the most famous variants are the imposition of time windows or the integration of pickup and delivery (for a more comprehensive account refer to (Toth and Vigo, 2002; Toth and Vigo, 2014) and references therein).

However, at the time of starting the work on this thesis and to the best of our knowledge, very little was known in the scientific literature about the possible operational benefits of combining trucks with drones or robots for short-haul transportation. In particular, formal models that allow for a cascading structure of vehicles, operating collaboratively as synchronized working units – such as trucks as a drone- or robot-carrying platform – had only been scarcely investigated (see, e. g., (Murray and Chu, 2015)). For such problems, it is possible to establish a distant relation to multi-echelon routing problems, which are

not an unknown phenomenon from an OR perspective (see, eg, (Crainic et al., 2009; Guastaroba et al., 2016)). The most commonly-studied variant consist of two distinct echelons: In the first echelon so called 'satellite' locations are served from a central depot; subsequently, deliveries can be carried out from satellites to final customer destinations.

Nevertheless, the works considered in this thesis differ from such problems. On the one hand, there is no distinct separation of satellites and customers (Sluijk et al., 2023). Instead, in its most challenging variant, the auxiliary vehicles are assumed to be carried by a delivery truck, and customers may be served by either the truck or one of the auxiliary vehicles that it carries; moreover, in such scenarios, customer locations also coincide as feasible satellite locations for supporting the operations of auxiliary vehicles (Otto et al., 2018). Even in the more simplistic case, i. e., when the auxiliary vehicles are assumed to be stationed in dedicated micro-depots, the same consideration applies. A micro-depot can be, e. g., a garage, the loading dock of a shop, or other infrastructure that may accommodate auxiliary vehicles (Boysen et al., 2021); and on its route the truck may opt to directly serve customers or visit micro-depots to mobilize the stationed vehicles, which may take some of the load and aid the truck in its task. On the other hand, in such scenarios, there is a frequent need for the vehicles to synchronize in order to synergize. In more detail, such problems are governed by extended task, operation, movement, load, and resource synchronization requirements (Drexl, 2012b). In contrast, e. g., in traditional capacitated VRPs synchronization concerns only which vehicle is assigned to which customer.

Consequently, there is a clear need for new decision models that formalize the possibilities of drone- or robot-assisted deliveries. As the resulting combinatorial optimization problems are shown to be $\mathcal{NP}$-hard, it is unlikely that an efficient algorithm for solving these problems is ever going to exist (unless $\mathcal{P} = \mathcal{NP}$). Therefore, heuristic solution approaches are imperative for solving larger instances effectively. Essentially, the aforementioned considerations enable the evaluation of potential benefits of using auxiliary vehicles in last-mile delivery. Such understanding may be crucial to making informed decisions concerning the design and operations of future delivery fleets. This, in turn, empowers us to contribute to establishing the groundwork for future decision support systems.

This thesis aims to address the problems outlined in the previous paragraphs by leveraging and advancing established techniques of OR (more details follow in Section 1.2). In OR, it is customary – and to some extent, an art – to tailor an algorithm in a way such that it exploits the structure of the underlying problem, a practice which we also attempt throughout this thesis. For some $\mathcal{NP}$-hard problems, this yields impressive algorithms that define the state-of-the-art and perform remarkably well for practical instances of these problems (see, e. g., (Applegate et al., 2007)).

Nevertheless, due to their widespread use and applicability, general-purpose *Mixed-Integer Programming* (MIP) solvers have established themselves as a cornerstone of OR (Jünger et al., 2010; Petropoulos et al., 2023). MIP solvers integrate refined algorithmic techniques, developed over several decades of research, in a *Branch-and-Cut* (B&C) scheme (Padberg and Rinaldi, 1991). Fundamentally, B&C involves solving linear programming problems (Dantzig et al., 1954) as relaxations and tightening these relaxations by using cutting planes (Gomory, 1963). These procedures are embedded in a *Branch-and-Bound* (B&B) framework (Land and Doig, 1960). State-of-the-art MIP solvers enrich this scheme by, e. g., cleverly devised branching strategies, more sophisticated cutting planes, effective presolve routines, and a variety of primal heuristics, all of which contribute to considerable speedups (see (Achterberg and Wunderling, 2013; Jünger et al., 2010; Koch et al., 2022) and references therein). Due to the complex interactions of these components, although complete and exact in the nature of their search, MIP solvers remain largely heuristic during their algorithmic execution (Lodi, 2013): This concerns, e. g., when to apply primal heuristics or which branching strategy to pursue.

State-of-the-art MIP solvers expose parameters that allow their users to control the intrinsic components on a coarse- or fine-grained level (see, e. g., (Fair Isaac Corporation, 2023; Gurobi Optimization, LLC, 2023)). However, the default configurations of such solvers are generally assumed to be expertly-tuned, enabling them to work well on a wide range of problems. Consequently, it is exceedingly rare to find OR publications in which the default parameters have been adjusted in any meaningful way, apart from enforcing resource limitations (typically, a time limit) – admittedly, this also applies to a large portion of the works presented in this thesis. This raises the question if it is possible to adjust the default parameters of such solvers in any reasonable way, as this may allow for superior performance on given problem (or a collection of problems that are of interest to a particular user). Any effective answer to this question may have wide-ranging consequences and be of great use when it comes to addressing almost any OR problem, including the logistics challenges outlined above. Although MIP solvers typically include integrated parameter tuning tools (see, e. g., (Gurobi Optimization, LLC, 2023)), such tools have considerable limitations. On the one hand, they often focus on tuning parameters for a single instance; and, on the other hand, there is no way to generalize to unseen models.

From a broader perspective, the challenge of selecting the optimal solution approach for a given problem can be considered as one of the "unsolved problems in problem solving" (Ackoff, 1962). Originally, Ackoff considered this question largely unsolved, emphasizing that any elected approach is mainly a result of the researcher's style rather than the outcome of a rigorous selection procedure – and that sentiment may have some truth even to this day. Clearly, this fundamental problem is of a general nature, and may arise in a

wide variety of situations beyond those described above, in the context of MIP solvers.

Nearly two decades later and coming from a different angle, a formalized version of this problem can be found in the work of Rice (1976). The *Algorithm Selection Problem* (ASP) asks us to find an efficient procedure to generate a mapping from problems to algorithms, ensuring optimal performance when applying the selected algorithm to a given problem. Rice recognized early on that a key component of any competitive approach to the ASP must make use of cheaply computable, informative features. Although the ASP is very general and seems widely applicable, its direct application has remained relatively obscure in the scientific literature for many years that followed.

On a related note, the well-known no-free lunch theorems provide a theoretical framework for algorithms to optimization problems (Wolpert and Macready, 1997). Briefly speaking, this framework establishes that that no black-box algorithm can be expected to perform better than any other on *all* optimization problems. While the no-free lunch theorem is subject of some controversy (see, e. g., (Adam et al., 2019) and references therein), it is crucial to stress that Wolpert and Macready reiterated on the importance of incorporating problem-specific knowledge into algorithm design: This might include cheaply computable, informative features – as required in the ASP – and may yield algorithms that perform better than others on *some* problems.

When addressing challenges (such as adjusting the control parameters of MIP solvers on a per-instance basis) from the perspective of the ASP, it may no longer be sufficient to only rely on established methods from the prescriptive OR toolbox, but instead required to take a more holistic stance (see also (Mingers and White, 2010)). This may be achieved by drawing from methods of other disciplines that converge in the analytics toolbox (Mortenson et al., 2015) (see Figure 1.1.1), especially branches of Artificial Intelligence such as *Machine Learning* (ML). This belief is reinforced by the fact that advancing methods of OR by means of ML constitutes one of the latest developments in the field, which may hold great potential for reshaping parts of the discipline (Bengio et al., 2021; Petropoulos et al., 2023). In recent years, ML and in particular deep learning has received widespread scientific attention and made remarkable advancements (see, e. g., (Goodfellow et al., 2016; Jordan and Mitchell, 2015; Mitchell, 1997)). It is mainly thanks to these advancements that the ASP and its related variants have also gained increased traction (see, e. g., (Hutter et al., 2011; Kerschke and Trautmann, 2019; Kotthoff, 2016)). However, even with good predictions it is rarely trivial to prescribe good decisions (Bertsimas and Kallus, 2020).

In revisiting the question that Ackoff (1962) sketched more than 60 years ago in the context of MIP, we also aim to contribute to the continued coalescence of prescriptive and predictive analytics on a methodological level. Any result in this context may not only be

of great use when it comes to dealing with the logistics challenges, which we outlined at the beginning of this section, but also to the broader discipline of OR.

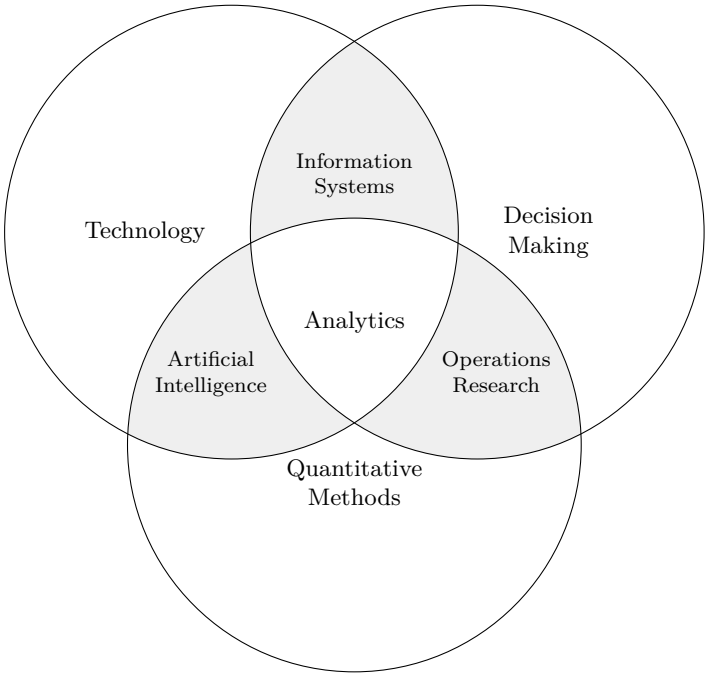

Figure 1.1.1: A taxonomy of disciplines related to analytics (following Mortenson et al. (2015)).

1.2 Organization and Contribution

The work presented in Chapters 2–7 of this thesis consists of three published journal articles, two papers that appeared in peer-reviewed conference proceedings, and one working paper, which is currently under consideration at a prestigious journal:

- Daniel Schermer, Mahdi Moeini, and Oliver Wendt (2019a). "A hybrid VNS/Tabu search algorithm for solving the vehicle routing problem with drones and en route operations". In: *Computers & Operations Research* 109, pp. 134–158 (Chapter 2),

- Daniel Schermer, Mahdi Moeini, and Oliver Wendt (2019b). "A matheuristic for the vehicle routing problem with drones and its variants". In: *Transportation Research Part C: Emerging Technologies* 106, pp. 166–204 (Chapter 3),

- Daniel Schermer, Mahdi Moeini, and Oliver Wendt (2020c). "The Traveling Salesman Drone Station Location Problem". In: *Optimization of Complex Systems: Theory, Models, Algorithms and Applications*. Advances in Intelligent Systems and Computing 991. Cham: Springer, pp. 1129–1138 (Chapter 4),
- Daniel Schermer, Mahdi Moeini, and Oliver Wendt (2020b). "The Drone-Assisted Traveling Salesman Problem with Robot Stations". In: *Proceedings of the Hawaii International Conference on System Sciences*, pp. 1308–1317 (Chapter 5),
- Daniel Schermer, Mahdi Moeini, and Oliver Wendt (2020a). "A branch-and-cut approach and alternative formulations for the traveling salesman problem with drone". In: *Networks* 76.2, pp. 164–186 (Chapter 6),
- Daniel Schermer (2023). "Leveraging Machine Learning for the Per-Instance Configuration of MIP Solvers: An Algorithm Selection Approach". Preprint. Chair of Business Information Systems & Operations Research, University of Kaiserslautern-Landau (Chapter 7).

In the following chapters of this thesis, these works are presented as listed above, in chronological order of their appearance.

In Chapter 2, we introduce the paper "A hybrid VNS/Tabu search algorithm for solving the vehicle routing problem with drones and en route operations" (Schermer et al., 2019a), co-authored with Mahdi Moeini and Oliver Wendt. With the goal of integrating drones in last-mile delivery, the *Vehicle Routing Problem with Drones* (VRPD) assumes that we are given a fleet of trucks, where each truck is equipped with one or more drones. The goal is to design feasible routes of the trucks and operations of the drones such that a set of customers is served with minimal makespan. In this paper, an extension of the VRPD called the *Vehicle Routing Problem with Drones and En Route Operations* (VRPDERO) is proposed (see also (Marinelli et al., 2018)). In this extended problem, drones may not only be launched and retrieved at customer locations but also on some discrete points that are positioned on the arcs that connect customers. This may be beneficial for improving the utilization of drones with limited flight endurance. The problem is modeled as a *Mixed-Integer Linear Program* (MILP) and the formulation enhanced by the introduction of *Valid Inequalities* (VIEQs). Due to limited performance of a state-of-the-art solver in addressing large-scale instances, an algorithm based on the concepts of Variable Neighborhood Search and Tabu Search is proposed. In order to evaluate the performance of the introduced algorithm as well as the solver in addressing the VRPDERO, we carried out extensive computational experiments. According to the numerical results, the proposed VIEQs and the heuristic have a significant contribution in solving the VRPDERO effectively. In

addition, the consideration of en route operations can increase the utilization of drones and lead to an improved makespan. This paper advances several of the foundations that were laid in (Schermer, 2019; Schermer et al., 2018b).

Chapter 3 is dedicated to the paper "A matheuristic for the vehicle routing problem with drones and its variants" (Schermer et al., 2019b), co-authored with Mahdi Moeini and Oliver Wendt. The focus of this work lies in the investigation of the VRPD in its most common variant. Given a fleet of trucks, where each truck carries a given number of drones, the objective consists in designing feasible routes and drone operations such that all customers are served and minimal makespan is achieved. The VRPD is formulated as a MILP and the formulation enhanced through several VIEQs. Due to limited performance of solvers in addressing large instances, a matheuristic approach is proposed that effectively exploits the problem structure of the VRPD. Integral to this approach is the *Drone Assignment and Scheduling Problem* (DASP) that, given an existing routing of trucks, looks for an optimal assignment and schedule of drones such that the makespan is minimized. Two MILP formulations for the DASP are proposed. In order to evaluate the performance of a state-of-the-art solver in tackling the MILP formulation of the VRPD, the benefit of the proposed VIEQs, and the performance of the matheuristic, an extensive computational study is presented. According to the numerical results, the use of drones can significantly reduce the makespan and the proposed VIEQs as well as the matheuristic approach have a significant contribution in solving the VRPD effectively. This paper differs from the one presented in Chapter 2 in several ways. Notably, the matheuristic approach has several advantages. On the one hand, once the truck routes are fixed, the problem can be decomposed into several DASP subproblems, which may be solved independently. On the other hand, in the scientific literature most of the general assumptions that determine the interactions between trucks and drones are agreed upon. However, several subtle nuances exist that make one problem statement deviate just slightly from another – even though, and confusingly, the problem names are often identical. The matheuristic approach has a unique advantage when it comes to customizability: As is demonstrated in the paper, this approach can easily be adapted to cover the most common assumptions concerning the truck-drone interaction. As such, this work can also be seen as an attempt to harmonize the diverse streams in the fast-growing literature.

Chapter 4 introduces the paper "The Traveling Salesman Drone Station Location Problem" (Schermer et al., 2020c), co-authored with Mahdi Moeini and Oliver Wendt. The *Traveling Salesman Drone Station Location Problem* (TSDSLP) exhibits features of the TSP as well as the Facility Location and Parallel Machine Scheduling problems. The TSDSLP involves a central depot with a truck and a set of customer locations, along with various potential micro-depot locations designed to harbor drones or robots. Such micro-

depots are assumed to be strategically located in close proximity to demand centers. The TSDSLP seeks a feasible routing of the truck and operations of the auxiliary vehicles such that all requests are fulfilled, no more than a given number of micro-depots is used, and the makespan (alternatively, the operational cost) is minimized. This problem falls within the general domain of *Location-Routing Problems* involving multiple echelons (see, e. g., (Min et al., 1998; Nagy and Salhi, 2006; Prodhon and Prins, 2014)). What is special about this problem is that the two resulting echelons (distribution levels) are not free of overlaps, as the truck can supply micro-depots or directly serve customers on its tours (see also (Sluijk et al., 2023)). Once a micro-depot has been supplied by the truck, the autonomous auxiliary vehicles, which are stationed there, are able to visit customers in close proximity by performing round trips. Two MILP formulations of the TSDSLP are provided, and a state-of-the-art solver is used to obtain solutions for small- and medium-sized instances. Overall, this work provides a different perspective on how auxiliary vehicles may be used for enhancing last-mile delivery. In contrast to previous works that follow the tandem approach, we demonstrate that slow-moving autonomous vehicles may already be very effective in this case. From a computational perspective, we demonstrate that the TSDSLP is significantly less challenging, due to the relaxed synchronization requirements.

Chapter 5 introduces the paper "The Drone-Assisted Traveling Salesman Problem with Robot Stations" (Schermer et al., 2020b), co-authored with Mahdi Moeini and Oliver Wendt. Specifically, in the *Drone-Assisted Traveling Salesman Problem with Robot Stations* (TSPDRS) there is a single truck that is equipped with a drone, and one or more potential sites of micro-depots, which might accommodate several robots. As such, the TSPDRS integrates the concepts of the *Traveling Salesman Problem with Drone* (TSPD) and TSDSLP by asking for a valid route of the truck as well as feasible utilization of the drone and robots, such that all customers are served and minimal makespan (or cost) is accomplished. We provide a MILP formulation of the problem and perform a detailed numerical study. Through our numerical results, it is revealed that our formulation can be effectively addressed by a state-of-the-art solver. It is demonstrated that optimizing the makespan coincides largely with reduced costs; and, in contrast, optimizing the operational costs might increase the makespan substantially. Notably, this paper introduces a novel and improved MILP formulation for truck-drone problems, compared to the ones shown in Chapters 2 and 3. Specifically, it is no longer necessary to account for *circumjacent* operations explicitly, resulting in a formulation that requires a significantly fewer variables and constraints, while also providing a tighter linear relaxation. Both of these factors speed up the solution process notably.

Chapter 6 presents the paper "A branch-and-cut approach and alternative formulations for the traveling salesman problem with drone" (Schermer et al., 2020a), co-authored with

Mahdi Moeini and Oliver Wendt. Although the improved MILP formulation presented in Chapter 5 may easily be adapted to cover the problems of previous chapters (see also Figure 1.2.1), it still has one major drawback: Many big-M constraints are required for correctly modeling the complex interactions between trucks and drones (see also (Ha et al., 2018; Murray and Chu, 2015; Schermer et al., 2020b)). It is well known that this generally results in poor linear relaxations of the involved model, making even small-sized instances difficult to be solved to optimality (Camm et al., 1990). Therefore, in Chapter 6, two new compact MILP formulations for the TSPD are provided that can be used to address instances with up to ten customers within a few seconds. Notably, a third formulation is introduced that cleverly decomposes the temporal relations of the problem and features an exponential number of constraints. As such, this formulation is suitable to be solved by a B&C algorithm that dynamically separates the exponential number of constraints. Indeed, this approach can be used to find (previously open) optimal solutions for several instances with up to 20 customers within one hour, thus challenging the state-of-the-art in optimally solving the TSPD. A detailed numerical study provides an in-depth comparison on the effectiveness of the proposed formulations. Moreover, further details on the operational characteristics of a drone-assisted delivery system are revealed. By using three different sets of benchmark instances, particular consideration is given to various assumptions that affect, e. g., technological drone parameters and the impact of distance metrics.

Chapter 7 presents the paper "Leveraging Machine Learning for the Per-Instance Configuration of MIP Solvers: An Algorithm Selection Approach" (Schermer, 2023). As is evident from the descriptions pertaining to Chapters 2–6 (see also Figure 1.2.1), MIP solvers play a crucial role in addressing combinatorial optimization problems and serve a dual purpose. First, they enable the determination of optimal solutions to small-sized instances, aiding in the validation of heuristic algorithm performance and the identification of structural properties of optimal solutions. Second, MIP solvers are essential for matheuristic algorithms, which are becoming increasingly popular (see, e. g., (Archetti and Speranza, 2014)). Both academic and commercial MIP solvers have made remarkable advancements (Koch et al., 2022). Due to the complexity and feature-richness of these solvers, in principle, most of them allow the execution of the inherent B&C framework to be configured in various ways that influence the optimization. However, despite the potential for performance gains, it is non-trivial to make such adjustments sensibly a priori. Consequently, it is common practice to rely on a default configuration, which is generally assumed to be expertly-tuned, thus enabling it to work well for a wide range of applications. In Chapter 7, we challenge the commonly accepted assumption that the default configuration of a MIP solver can be expected to work well for a wide range of problems. To be more precise, the problem of determining a configuration for a MIP

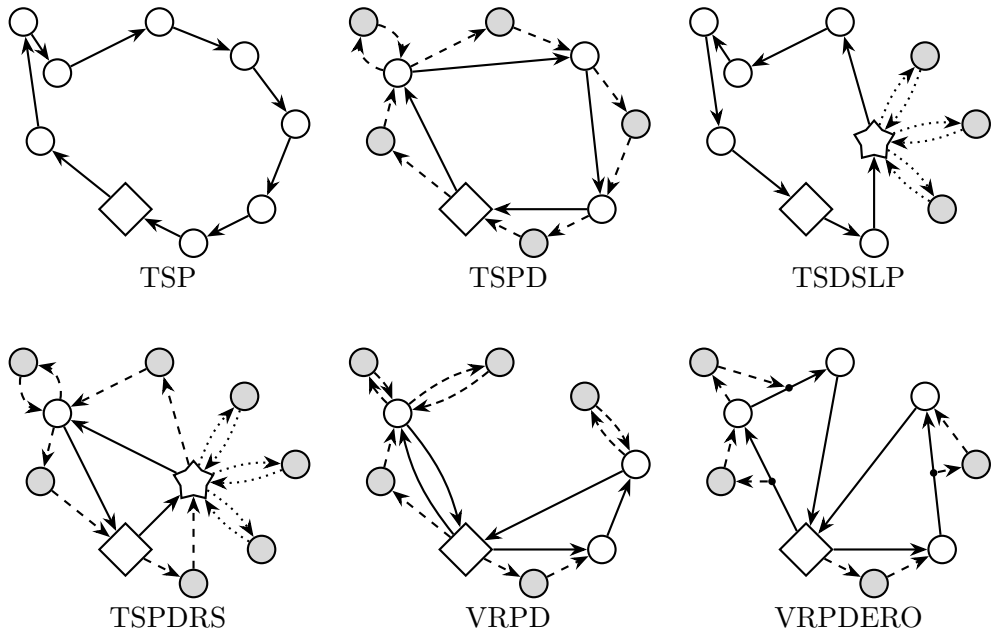

Figure 1.2.1: An illustration of the TSP, TSPD, TSDSLP, TSPDRS, VRPD, and VRPDERO. The paths of the truck, drones, and robots are indicated by solid, dashed, and dotted lines, respectively. In these figures, the depot and micro-depot are indicated by the square and star shapes.

solver on a per-instance basis is addressed from the perspective of the ASP. To that end, a novel methodology is proposed that follows the general idea of Empirical Model Learning (Lombardi et al., 2017). In more detail, *Random Forest* (RF) regression is used as a ML model to learn an approximation of a solver's performance. The ML model helps us to extract the components of a prescriptive optimization problem, which seeks the optimal configuration for a given instance. It is demonstrated that this problem can be efficiently solved by a tailored B&B approach that exploits the structure of the underlying RF ML model. All things considered, this methodology empowers us to compute the (predictably) optimal configuration on a per-instance basis with minimal computational overhead. The methodology is evaluated by using a state-of-the-art solver and the established MIPLIB benchmark instances. The extensive numerical results demonstrate that the performance gap between the default and virtual-best solver can be closed by about half with respect to the average runtime and MIP gap.

We conclude this thesis in Chapter 8 by summarizing its contents, drawing concluding remarks, and providing a future outlook.

1.3 References

Achterberg, Tobias and Roland Wunderling (2013). "Mixed Integer Programming: Analyzing 12 Years of Progress". In: *Facets of Combinatorial Optimization*. Ed. by Michael Jünger and Gerhard Reinelt. Berlin, Heidelberg: Springer, pp. 449–481.

Ackoff, Russell L. (1962). "Some Unsolved Problems in Problem Solving". In: *Operational Research Quarterly* 13.1, pp. 1–11.

Adam, Stavros P. et al. (2019). "No Free Lunch Theorem: A Review". In: *Approximation and Optimization: Algorithms, Complexity and Applications*. Springer Optimization and Its Applications, 145. Cham: Springer, pp. 57–82.

Applegate, David L. et al. (2007). *The Traveling Salesman Problem: A Computational Study*. Princeton University Press.

Archetti, Claudia and M. Grazia Speranza (2014). "A survey on matheuristics for routing problems". In: *EURO Journal on Computational Optimization* 2.4, pp. 223–246.

Azadeh, Kaveh, René De Koster, and Debjit Roy (2019). "Robotized and Automated Warehouse Systems: Review and Recent Developments". In: *Transportation Science* 53.4, pp. 917–945.

Bengio, Yoshua, Andrea Lodi, and Antoine Prouvost (2021). "Machine learning for combinatorial optimization: A methodological tour d'horizon". In: *European Journal of Operational Research* 290.2, pp. 405–421.

Bertsimas, Dimitris and Nathan Kallus (2020). "From Predictive to Prescriptive Analytics". In: *Management Science* 66.3, pp. 1025–1044.

Boysen, Nils, Stefan Fedtke, and Stefan Schwerdfeger (2021). "Last-mile delivery concepts: a survey from an operational research perspective". In: *OR Spectrum* 43.1, pp. 1–58.

Camm, Jeffrey D., Amitabh S. Raturi, and Shigeru Tsubakitani (1990). "Cutting Big M down to Size". In: *Interfaces* 20.5, pp. 61–66.

Crainic, Teodor Gabriel, Nicoletta Ricciardi, and Giovanni Storchi (2009). "Models for Evaluating and Planning City Logistics Systems". In: *Transportation Science* 43.4, pp. 432–454.

Dantzig, George B., 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.

Dantzig, George B. and John. H. Ramser (1959). "The Truck Dispatching Problem". In: *Management Science* 6.1, pp. 80–91.

Drexl, Michael (2012b). "Synchronization in Vehicle Routing—A Survey of VRPs with Multiple Synchronization Constraints". In: *Transportation Science* 46.3, pp. 297–316.

Fair Isaac Corporation (2023). *FICO Xpress Optimizer Reference Manual*.

Fragapane, Giuseppe et al. (2021). "Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda". In: *European Journal of Operational Research* 294.2, pp. 405–426.

Ghiani, Gianpaolo, Gilbert Laporte, and Roberto Musmanno (2013). *Introduction to logistics systems management*. 2nd ed. Wiley series in operations research and management science. Chichester: John Wiley & Sons, Ltd.

Goeke, Dominik (2018). "Emerging trends in logistics". Ph.D. thesis. Technische Universität Kaiserslautern.

Gomory, Ralph E (1963). "An algorithm for integer solutions to linear programs". In: *Recent advances in mathematical programming* 64.14, pp. 260–302.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep learning*. Adaptive computation and machine learning. Cambridge: The MIT Press.

Guastaroba, G., M. G. Speranza, and D. Vigo (2016). "Intermediate Facilities in Freight Transportation Planning: A Survey". In: *Transportation Science* 50.3, pp. 763–789.

Gurobi Optimization, LLC (2023). *Gurobi Optimizer Reference Manual*.

Ha, Quang Minh et al. (2018). "On the min-cost Traveling Salesman Problem with Drone". In: *Transportation Research Part C: Emerging Technologies* 86, pp. 597–621.

Hof, Julian (2019). "Metaheuristics for Vehicle-Routing Problems Arising in Sustainable Logistics". Ph.D. thesis. Technische Universität Kaiserslautern.

Hutter, Frank, Holger H. Hoos, and Kevin Leyton-Brown (2011). "Sequential Model-Based Optimization for General Algorithm Configuration". In: *Learning and Intelligent Optimization*. Lecture Notes in Computer Science 6683. Berlin, Heidelberg: Springer, pp. 507–523.

Jordan, Michael I. and Tom M. Mitchell (2015). "Machine learning: Trends, perspectives, and prospects". In: *Science* 349.6245, pp. 255–260.

Jünger, Michael et al., eds. (2010). *50 Years of Integer Programming 1958-2008*. Berlin, Heidelberg: Springer.

Kerschke, Pascal and Heike Trautmann (2019). "Automated Algorithm Selection on Continuous Black-Box Problems by Combining Exploratory Landscape Analysis and Machine Learning". In: *Evolutionary Computation* 27.1, pp. 99–127.

Koch, Thorsten et al. (2022). "Progress in mathematical programming solvers from 2001 to 2020". In: *EURO Journal on Computational Optimization* 10, p. 100031.

Kotthoff, Lars (2016). "Algorithm Selection for Combinatorial Search Problems: A Survey". In: *Data Mining and Constraint Programming*. Lecture Notes in Computer Science 10101. Cham: Springer, pp. 149–190.

Land, A. H. and A. G. Doig (1960). "An Automatic Method of Solving Discrete Programming Problems". In: *Econometrica* 28.3, pp. 497–520.

Lodi, Andrea (2013). "The Heuristic (Dark) Side of MIP Solvers". In: *Hybrid Metaheuristics*. Ed. by El-Ghazali Talbi. Studies in Computational Intelligence 434. Berlin, Heidelberg: Springer, pp. 273–284.

Lombardi, Michele, Michela Milano, and Andrea Bartolini (2017). "Empirical decision model learning". In: *Artificial Intelligence* 244, pp. 343–367.

Macrina, Giusy et al. (2020). "Drone-aided routing: A literature review". In: *Transportation Research Part C: Emerging Technologies* 120, p. 102762.

Marinelli, Mario et al. (2018). "En route truck–drone parcel delivery for optimal vehicle routing strategies". In: *IET Intelligent Transport Systems* 12.4, pp. 253–261.

Min, Hokey, Vaidyanathan Jayaraman, and Rajesh Srivastava (1998). "Combined location-routing problems: A synthesis and future research directions". In: *European Journal of Operational Research* 108.1, pp. 1–15.

Mingers, John and Leroy White (2010). "A review of the recent contribution of systems thinking to operational research and management science". In: *European Journal of Operational Research* 207.3, pp. 1147–1161.

Mitchell, Tom M. (1997). *Machine Learning*. McGraw-Hill series in computer science. New York: McGraw-Hill.

Mortenson, Michael J., Neil F. Doherty, and Stewart Robinson (2015). "Operational research from Taylorism to Terabytes: A research agenda for the analytics age". In: *European Journal of Operational Research* 241.3, pp. 583–595.

Murray, Chase C. and Amanda G. Chu (2015). "The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery". In: *Transportation Research Part C: Emerging Technologies* 54, pp. 86–109.

Nagy, Gábor and Saïd Salhi (2006). "Location-routing: Issues, models and methods". In: *European Journal of Operational Research* 177.2, pp. 649–672.

Otto, Alena et al. (2018). "Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey". In: *Networks* 72.4, pp. 411–458.

Padberg, Manfred and Giovanni Rinaldi (1991). "A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems". In: *SIAM Review* 33.1, pp. 60–100.

Pelletier, Samuel, Ola Jabali, and Gilbert Laporte (2016). "50th Anniversary Invited Article—Goods Distribution with Electric Vehicles: Review and Research Perspectives". In: *Transportation Science* 50.1, pp. 3–22.

Petropoulos, Fotios et al. (2023). "Operational Research: Methods and Applications". In: *arXiv:2303.14217 [math.OC]*.

Prodhon, Caroline and Christian Prins (2014). "A survey of recent research on location-routing problems". In: *European Journal of Operational Research* 238.1, pp. 1–17.

Rice, John R. (1976). "The Algorithm Selection Problem". In: *Advances in Computers*. Vol. 15. Elsevier, pp. 65–118.

Savelsbergh, Martin and Tom Van Woensel (2016). "50th Anniversary Invited Article—City Logistics: Challenges and Opportunities". In: *Transportation Science* 50.2, pp. 579–590.

Schermer, Daniel (2019). "Integration of Drones in Last-Mile Delivery: The Vehicle Routing Problem with Drones". In: *Operations Research Proceedings 2018*. Operations Research Proceedings. Cham: Springer, pp. 17–22.

Schermer, Daniel (2023). "Leveraging Machine Learning for the Per-Instance Configuration of MIP Solvers: An Algorithm Selection Approach". Preprint. Chair of Business Information Systems & Operations Research, University of Kaiserslautern-Landau.

Schermer, Daniel, Mahdi Moeini, and Oliver Wendt (2018b). "Algorithms for Solving the Vehicle Routing Problem with Drones". In: *Intelligent Information and Database Systems*. Lecture Notes in Computer Science 10751. Cham: Springer, pp. 352–361.

Schermer, Daniel, Mahdi Moeini, and Oliver Wendt (2019a). "A hybrid VNS/Tabu search algorithm for solving the vehicle routing problem with drones and en route operations". In: *Computers & Operations Research* 109, pp. 134–158.

Schermer, Daniel, Mahdi Moeini, and Oliver Wendt (2019b). "A matheuristic for the vehicle routing problem with drones and its variants". In: *Transportation Research Part C: Emerging Technologies* 106, pp. 166–204.

Schermer, Daniel, Mahdi Moeini, and Oliver Wendt (2020a). "A branch-and-cut approach and alternative formulations for the traveling salesman problem with drone". In: *Networks* 76.2, pp. 164–186.

Schermer, Daniel, Mahdi Moeini, and Oliver Wendt (2020b). "The Drone-Assisted Traveling Salesman Problem with Robot Stations". In: *Proceedings of the Hawaii International Conference on System Sciences*, pp. 1308–1317.

Schermer, Daniel, Mahdi Moeini, and Oliver Wendt (2020c). "The Traveling Salesman Drone Station Location Problem". In: *Optimization of Complex Systems: Theory, Models, Algorithms and Applications*. Advances in Intelligent Systems and Computing 991. Cham: Springer, pp. 1129–1138.

Sluijk, Natasja et al. (2023). "Two-echelon vehicle routing problems: A literature review". In: *European Journal of Operational Research* 304.3, pp. 865–886.

Toth, Paolo and Daniele Vigo (2002). *The vehicle routing problem*. 1st ed. SIAM Series on Discrete Mathematics and Applications. Philadelphia: SIAM.

Toth, Paolo and Daniele Vigo, eds. (2014). *Vehicle routing: problems, methods, and applications*. 2nd ed. MOS-SIAM series on optimization. Philadelphia: SIAM.

Wolpert, D.H. and W.G. Macready (1997). “No free lunch theorems for optimization”. In: *IEEE Transactions on Evolutionary Computation* 1.1, pp. 67–82.

2 A Hybrid VNS/Tabu Search Algorithm for Solving the Vehicle Routing Problem with Drones and En Route Operations

Abstract

With the goal of integrating drones in last-mile delivery, the *Vehicle Routing Problem with Drones* (VRPD) uses a fleet of trucks, each of them equipped with a set of drones, for serving a set of customers with minimal makespan. In this paper, we propose an extension of the VRPD that we call the *Vehicle Routing Problem with Drones and En Route Operations* (VRPDERO). Here, in contrast to the VRPD, drones may not only be launched and retrieved at vertices but also on some discrete points that are located on each arc. We formulate the problem as a *Mixed-Integer Linear Program* (MILP) and introduce some *Valid Inequalities* (VIEQs) that enhance the performance of the MILP solvers. Furthermore, due to limited performance of the solvers in addressing large-scale instances, we propose an algorithm based on the concepts of *Variable Neighborhood Search* (VNS) and *Tabu Search* (TS). In order to evaluate the performance of the introduced algorithm as well as the solver in solving the VRPDERO instances, we carried out extensive computational experiments. According to the numerical results, the proposed VIEQs and the heuristic have a significant contribution in solving the VRPDERO effectively. In addition, the consideration of en route operations can increase the utilization of drones and lead to an improved makespan.

2.1 Introduction

Minimum cost and fast pickup and delivery of products are important problems in logistics. In this context, e. g., several variants of the classical *Vehicle Routing Problem* (VRP) have been studied and solved using a wide range of heuristic and exact algorithms (see (Toth and Vigo, 2002), and references therein).

Recently, there has been an interest in the integration of drones in civil applications and, particularly, in last-mile delivery. Drones have been successfully applied in many public and private sectors and are slowly becoming a proven technology (see, e. g., (Otto et al., 2018)). As a result, in an effort to lower costs and reduce delivery times, several optimization problems have been proposed in the academic literature that integrate drones into last-mile delivery (see, e. g., (Agatz et al., 2018; Murray and Chu, 2015; X. Wang et al.,

2017)). In these problems, which might be considered as variants of the VRP or *Traveling Salesman Problem* (TSP), the shared idea of a tandem that consists of traditional trucks and drones is commonly accepted. Hence, a drone might be launched from a truck, makes an autonomous flight to deliver a parcel to an identified customer, and is then retrieved by the truck at a different location. Generally, we might assume that a drone can move faster between two locations than a truck as drones are not exposed to the restriction of following the road network structure or congestion. Additionally, drones are lightweight, which causes them to consume less energy for the movement between two points. However, their carrying capacity is more restricted than that of a truck. Moreover, as drones rely on relatively small batteries for powering their flight, their range of operation is more tightly bounded than that of a truck powered by a fossil fuel or a much larger battery. Consequently, due to the complementary features of trucks and drones, the latter might have a supplementary role in last-mile delivery. Classically, it is commonly assumed that the launch and retrieval of drones is restricted to the locations of customers and the depot (Agatz et al., 2018; Murray and Chu, 2015). However, several researchers have suggested the relaxation of this assumption and promoted the idea of launching and retrieving drones at arbitrary locations (Marinelli et al., 2018; Poikonen et al., 2017). In particular, Marinelli et al. (2018) propose *en route* operations, i. e., the possibility to construct *arc-based* truck-drone operations. Indeed, as one of many companies that are interested in the integration of drones in last-mile delivery, Amazon Inc. has patented the possibility of incorporating drone docking stations into existing structures such as cell towers, light and power poles, and buildings that might be used as intermediate docks for supporting en route operations (Gentry et al., 2016).

In this article, we propose a new problem and we call it the VRPDERO, which is an extension of the VRPD proposed by X. Wang et al. (2017). In fact, compared to the VRPD, the VRPDERO assumes that drones might also be launched and retrieved at some discrete locations on each arc. The contributions of our work are as follows:

1. We formulate the VRPDERO as a MILP. Consequently, the problem might be solved by any standard MILP solver. Furthermore, by considering the definition and mathematical structure of the VRPDERO, we introduce some sets of VIEQs that can be beneficial in enhancing the performance of the MILP solvers. Through our numerical results, we show that our proposed VIEQs have a significant impact on the solver's performance in finding optimal solutions to small instances.

2. In the VRPDERO, the MILP formulation can only be used on small-scale problems to obtain optimal solutions in a reasonable amount of time. Therefore, in order to address larger instances of the problem, we propose a heuristic that combines compo-

nents from VNS and TS. Furthermore, our algorithm incorporates a problem-specific drone insertion operator as local search method that is designed according to the structure and characteristics of the VRPDERO. This operator is embedded within a scalable *Divide-and-Conquer* (D&C) approach. We carried out extensive computational experiments on VRPDERO instances, and we show that our algorithm provides high-quality solutions in reasonable computation time.

3. Across our numerical study, we demonstrate that, with respect to the VRPD, the consideration of en route operations can be beneficial for improving the utilization of drones, thus improving the makespan. Albeit, the introduction of en route operations significantly increases the computational complexity of the problem.

The remainder of this paper is organized as follows. We begin in Section 2.2 by providing a brief overview on the existing academic literature related to the application of drones in last-mile delivery. Then, Section 2.3 presents the notation, the mathematical model, and the proposed VIEQs for the VRPDERO. In Section 2.4, we introduce a heuristic approach that is based on the concepts of VNS and TS. This section also contains a detailed description of the drone insertion procedure and the associated D&C approach. Detailed results on the numerical studies, which were performed using both the MILP formulation and the heuristic approach, are presented in Section 2.5. Finally, Section 2.6 is devoted to the concluding remarks and future research implications.

2.2 Related Research Works

Otto et al. (2018) provide a comprehensive overview on the use of drones in civil applications. However, for the sake of completeness, we present a brief overview of some of the research works that are most relevant to this paper.

Murray and Chu (2015) introduced two new $\mathcal{NP}$-hard problems, called the *Flying Sidekick Taveling Salesman Problem* (FSTSP) and the *Parallel Drone Scheduling Traveling Salesman Problem* (PDSTSP), that are related to delivery applications, where a truck is working in tandem with a drone. In the FSTSP, a set of customers is given, each of whom must be visited exactly once by either a drone or a truck. The truck-drone-tandem starts from a common depot and must return to the same depot at the end of the tour. At any customer location, the drone may be launched from the truck, makes an autonomous delivery to another customer location, and is then retrieved by the truck at a different location. In the PDSTSP, it is assumed that the depot is in close proximity to most of the customers. In this case, it can be beneficial to let the drone make its deliveries independently of the truck, while the truck serves customers that are located far away from

the depot. Whereas the truck continues on a predefined tour, the drone will continuously return to the depot to pick up a new parcel after each delivery. Both problems, i. e., the FSTSP and PDSTSP, share the objective of minimizing the makespan. However, the latter has significantly reduced *synchronization* requirements (Drexl, 2012b). Murray and Chu (2015) proposed two simple heuristic methods for solving the FSTSP and PDSTSP, respectively, as well as a MILP formulation for either problem. They show that a significant makespan reduction is possible through the introduction of drones and highlight that the FSTSP is more computationally challenging.

The *Traveling Salesman Problem with Drone* (TSPD), introduced by Agatz et al. (2018), is another problem that shares most features with the FSTSP. A key difference to the FSTSP is the relaxation that the TSPD permits a drone to be launched and retrieved at the same vertex. Agatz et al. (2018) provide theoretical insights and propose a MILP formulation as well as several new heuristics for solving the TSPD. They confirm that significant savings are possible and conclude by suggesting the extension of their model to consider multiple trucks or drones as well as adding different possibilities of recharging the drones.

X. Wang et al. (2017) introduced the VRPD which might be considered as a generalization of the FSTSP and TSPD. In the VRPD, we assume that a depot, a set of customers, and a fleet of trucks is given such that each truck is equipped with a predefined number of drones. The objective is to minimize the makespan, i. e., the time required to serve all customers such that the given vehicle fleet (i. e., trucks and drones) has returned to the depot at the end of the mission. Wang et al. provided several theoretical insights. In particular, they introduced some upper bounds on the potential reductions in makespan by employing drones (compared to a case where no drones are given). The upper bounds are obtained by investigating the structure of optimal solutions and depend on the relative velocity of the drones compared to the trucks and the number of drones carried by each truck. Poikonen et al. (2017) refined the work of X. Wang et al. (2017) by further considering the impact on the previously introduced bounds when integrating limited battery life, different distance metrics, and operational expenditures of deploying drones and trucks in the objective function. Furthermore, Poikonen et al. (2017) proposed extending the VRPD model similar to the close-enough TSP. More precisely, the authors suggest the possibility of launching and recovering drones from arbitrary locations instead of being restricted to the locations of customers and the depot. They conclude their work by proposing that heuristics and benchmark instances should be developed as a next step in researching the VRPD from a computational point of view.

In (Schermer et al., 2018a), based on the work by Di Puglia Pugliese and Guerriero (2017), we introduced a MILP formulation for the VRPD along with several VIEQs.

In particular, we provided a heuristic approach based on VNS and two heuristic drone insertion operators for solving the VRPD. Through an extensive numerical study, we showed the effectiveness of our proposed VIEQs and heuristic for solving small- and large-scale instances of the VRPD. Finally, we could demonstrate that a truck-drone-tandem allows for a significantly reduced makespan compared to classic truck delivery.

Marinelli et al. (2018) allow drones to be launched and retrieved from arbitrary locations that are located on a predefined path of the truck. More precisely, they propose a novel variant of the TSPD in which the drones can be launched and retrieved on arcs by the truck, which they call an *en route operation*. They state that this procedure can be beneficial in increasing the utilization of drones and in overcoming their limited range of operation. In order to solve the extended TSPD, Marinelli et al. (2018) propose a heuristic which consists of three distinct steps: First, they find a routing for the truck using the well-known Lin-Kernighan heuristic (Lin and Kernighan, 1973). Second, given the routing of the truck, they build a list of all feasible drone operations that yield a reduced cost. Third, the initial list is improved through the consideration of en route operation and the moves are inserted according to their reduced cost. Through their numerical study, they show that en route operations can yield improvements in the order of 0 to 10% over the non-en route case with regard to the makespan.

2.3 The Vehicle Routing Problem with Drones and En Route Operations

In this paper, we introduce and study the VRPDERO, which might be considered an extension to the VRPD. We recall that in the VRPD, the drone sorties, i. e., the operation of launching from a truck, serving a customer, and returning to the truck, might only be done at the vertices (i. e., customer locations and the depot). However, in the VRPDERO, the operations might also start and end at some discrete points which are located on the arcs that are traversed by the trucks. In other words, the VRPDERO admits *en route operations*. In fact, en route operations might be useful in two cases as highlighted in Figure 2.3.1. On the one hand, if a drone is much faster than the truck, then it might be able to perform multiple operations during a single transition of the truck. On the other hand, if a drone has a low endurance, then it might be possible to perform operations, that would otherwise not be feasible. Hence, the VRPDERO can be considered as a natural extension of the VRPD.

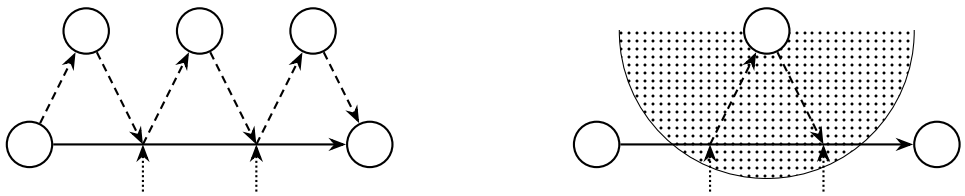

Figure 2.3.1: The paths of the truck and drone are shown using solid and dashed lines, respectively. The dotted arrows on the solid arcs indicate locations at which en route operations are possible. On the left-hand side, one fast-moving drone is able to perform multiple operations during a single transition of the truck. On the right-hand side, we assume that the customer might only be served by drone, if the drone is launched and retrieved within the feasible region (indicated by the dotted pattern). Here, through en route operations, a drone is able to perform an operation that would otherwise not be possible.

2.3.1 Notation and Mathematical Model

Suppose that a set of customer locations, each with an equal demand, and a fleet of homogeneous trucks, each carrying a specified number of identical drones, are given. In the VRPDERO, we look for minimizing the time required to serve all customers using the trucks and the drones such that, by the end of the mission, all trucks and drones must be at the depot. Furthermore, we make the following assumptions (Murray and Chu, 2015; Poikonen et al., 2017; X. Wang et al., 2017):

- A drone can serve exactly one customer per operation and has a limited endurance of $\mathcal{E}$ distance units. After returning to the truck, the battery of the drone is recharged instantaneously (e. g., by swapping it).
- While the trucks are restricted to the road network, the drones might be able to use a different trajectory for traveling between locations. In addition, without loss of generality, the average velocity of each truck is assumed to be equal to 1 and the relative velocity of each drone is assumed to be $\alpha \in \mathbb{R}^+$ times the velocity of the truck. Due to the absence of congestion or necessary stops (e. g., traffic lights and stop signs) in the route of the drone, we assume that $\alpha \geq 1$.
- The service time required to serve a customer by truck or drone and the time required to launch and retrieve a drone are assumed to be negligible.
- Drones may only be launched and retrieved either at vertices (i. e., at the depot and any other customer location) or at predefined discrete points which are located on arcs that are traversed by trucks. Furthermore, a drone may not be picked up at the same location from which it was launched. Finally, a drone must always return to the same truck from which it was launched.

In order to present the mathematical model of the VRPDERO, we use the following notation which is similar to that in (Di Puglia Pugliese and Guerriero, 2017; Schermer et al., 2018a).

Assume that a complete and symmetric graph $\mathcal{G} = (V, A)$ is given, where V is the set of vertices and A is the set of arcs. The set V contains n vertices associated with the customers, named $V_N = \{1, \ldots, n\}$ and (in order to simplify the notation and the mathematical formulation) two extra vertices 0 and $n+1$ that both represent the same depot location at the start and end of each tour, respectively. Hence, $V = \{0\} \cup V_N \cup \{n+1\}$, where $0 \equiv n+1$. Each vertex $V_i \in V$ is defined by a tuple (x_i, y_i) in the Euclidean space. We refer to $V_L = V \setminus \{n+1\}$ as the subset of vertices in V from which drones may be launched. Furthermore, $V_R = V \setminus \{0\}$ denotes the subset of vertices where the drones may be retrieved.

In order to define the discrete points on the arcs, let $S = \{0, 1, \ldots, s_n\}$ be a set of predefined integer numbers, where $s_n \in \mathbb{Z}^+$, $|S| \geq 2$. Given two vertices $V_i \in V_L, V_j \in V_R$, the location of a discrete point $\vec{V_{ij}^s}$ that is located at step $s \in S$ on arc $\vec{A_{ij}} \in A$ (connecting $\vec{V_i}$ to $\vec{V_j}$) can be calculated through formula (2.1):

$$\vec{V_{ij}^s} = \vec{V_i} + \frac{s}{|S|-1}(\vec{V_j} - \vec{V_i}) = \vec{V_i} + \frac{s}{|S|-1}\vec{A_{ij}} \tag{2.1}$$

We assume that for $s = 0$ and $s = s_n$, $\vec{V_{ij}^s}$ coincides with $\vec{V_i}$ and $\vec{V_j}$, respectively. Further, we use the sets $S_L = S \setminus \{s_n\}$ and $S_R = S \setminus \{0\}$ for defining the discrete points on each arc from which a drone might be launched and retrieved.

The parameters d_{ij} and $\overline{d}_{ij}$ define the required distance to travel from vertex i to vertex j by truck and drone, respectively. As $\mathcal{G}$ is assumed to be symmetric, $d_{ij} = d_{ji}$ and $\overline{d}_{ij} = \overline{d}_{ji} \, \forall i, j \in V$. Furthermore, by definition, $d_{ii} = \overline{d}_{ii} = 0 \, \forall i \in V$. Additionally, we introduce $\overline{d}_{ij}^{w,s}$ as the distance between V_{ij}^s and V_w. As the drone may not be limited to the road network, we can assume that $\overline{d}_{ij} \leq d_{ij} \, \forall i, j \in V$. Naturally, the triangle inequality is assumed to hold for these distances.

We use the parameters v and $\overline{v}$, where $v = 1$ and $\overline{v} = \alpha \cdot v$, to indicate the velocity of the truck and drone, respectively. We assume that both, the truck and the drone, move at a constant velocity. Hence, the time required to traverse edge (i, j) is defined as $t_{ij} = d_{ij}$ and $\overline{t}_{ij} = \overline{d}_{ij}/\alpha$ for the trucks and drones, respectively. As we assumed that $\alpha \geq 1$ and $\overline{d}_{ij} \leq d_{ij} \, \forall i, j \in V$, we can conclude that $\overline{t}_{ij} \leq t_{ij} \, \forall i, j \in V$. Analogously, the time required to reach (or return from) a customer $w \in V_N$ from (or to) a discrete point V_{ij}^s, where $V_i \in V_L, V_j \in V_R$ and $s \in S$, by a drone is given by $\overline{t}_{ij}^{w,s} = \overline{d}_{ij}^{w,s}/\alpha$.

As in the VRPD (Poikonen et al., 2017; X. Wang et al., 2017), we suppose that we have a given fleet that consists of a limited set $K = \{1, \ldots, K_N\}$ ($K_N \in \mathbb{Z}^+, |K| \geq 1$) of trucks

such that each truck $k \in K$ holds an equal number of $D \in \mathbb{Z}^{\geq 0}$ identical drones. Each drone may travel a maximum distance of $\mathcal{E}$ units per operation, where a *drone-operation* is characterized by a tuple (i, s_l, w, s_r, j) or, alternatively, two tuples: (i, w, s_l) for deliveries and (w, j, s_r) for retrievals. More precisely, the drone is launched from a truck at $s_l \in S_L$ en route from $i \in V_L$ to serve a customer $w \in V_N$. Afterwards, the drone is retrieved by the truck, from which the drone was launched, at $s_r \in S_R$ en route to $j \in V_R$.

Table 2.3.1: Decision variables used in the VRPDERO formulation.

$\tau \in \mathbb{R}^{\geq 0}$	Continuous variable that indicates the makespan.
$x_{ij}^k \in \{0,1\}$ $\forall k \in K, i \in V_L, j \in V_R$	Variables that state whether arc (i,j) belongs to the route of truck k.
$y_{iw}^{ks_l} \in \{0,1\}$ $\forall k \in K, i \in V_L, w \in V_N, s_l \in S_L$	Variables that indicate if a drone that belongs to truck k is launched at s_l en route from vertex i to serve w.
$\tilde{y}_{wj}^{ks_r} \in \{0,1\}$ $\forall k \in K, w \in V_N, j \in V_R s_r \in S_R$	Variables that specify if a drone that belongs to truck k is retrieved at s_r en route to vertex j after a delivery to w.
$s_{iwj}^k \in \{0,1\}$ $\forall k \in K, i \in V_L, w \in V_N, j \in V_R$	Variables that state if a drone that belongs to truck k performs a delivery to w that occurs on arc (i,j) at any feasible combination of steps $s_l \in S_L$ and $s_r \in S_R$.
$u_i^k \in \mathbb{Z}^{\geq 0}$ $\forall k \in K, i \in V$	Continuous variables that indicate the position of vertex i in the route of truck k (if customer i is served by truck k).
$p_{ij}^k \in \{0,1\}$ $\forall k \in K, \forall i \in V_L, j \in V_R$	Variables that specify whether i is served before (but not necessarily adjacent to) j by truck k.
$z_{aiwje}^k \in \{0,1\}$ $\forall k \in K, a, i, w, j, e \in V$	Variables that state whether any drone delivery that targets w en route from (or at) a and ends en route to (or at) e is performed circumjacent to (i,j), i. e., launched before the truck k has reached i and retrieved after the truck has reached j.
$a_i^k \in \mathbb{R}^{\geq 0}$ $\forall k \in K, i \in V$	Continuous variables that indicate the earliest time at which vertex i is reached by the truck and any drone recovered directly at i. If i is served by a drone, this variable simply marks the earliest time at which i is served.
$a_j^{ks} \in \mathbb{R}^{\geq 0}$ $\forall k \in K, j \in V_R, s \in S_R$	Continuous variables that indicate the earliest time at which step s en route to j is reached by the truck k and any drone that must be recovered here.

We use the decision variables in Table 2.3.1 for modeling the VRPDERO (see also Figure 2.3.2). Using this notation, the MILP formulation of the VRPDERO is given by (2.2)–(2.25):

$$\min \quad \tau, \tag{2.2}$$

$$\text{s. t.} \quad a_{n+1}^k \leq \tau \quad \forall k \in K, \tag{2.3}$$

$$\sum_{j \in V_N} x_{0j}^k \leq 1 \quad \forall k \in K, \tag{2.4}$$

$$\sum_{j \in V_N} x_{0j}^k - \sum_{i \in V_N} x_{i,n+1}^k = 0 \quad \forall k \in K, \tag{2.5}$$

$$\sum_{i \in V_L} x_{ih}^k - \sum_{j \in V_R} x_{hj}^k = 0 \quad \forall k \in K, h \in V_N, \tag{2.6}$$

$$\sum_{k \in K} \sum_{i \in V_L} x_{iw}^k + \sum_{k \in K} \sum_{i \in V_L} \sum_{s_l \in S_L} y_{iw}^{ks_l} = 1 \quad \forall w \in V_N, \tag{2.7}$$

$$\sum_{w \in V_N} \sum_{s_l \in S_L} y_{iw}^{ks_l} \leq M \sum_{j \in V_R} x_{ij}^k \quad \forall k \in K, i \in V_L, \tag{2.8}$$

$$\sum_{w \in V_N} \sum_{s_r \in S_R} \tilde{y}_{wj}^{ks_l} \leq M \sum_{i \in V_L} x_{ij}^k \quad \forall k \in K, j \in V_R, \tag{2.9}$$

$$\sum_{i \in V_L} \sum_{s_l \in S_L} y_{iw}^{ks_l} - \sum_{j \in V_R} \sum_{s_r \in S_R} \tilde{y}_{wj}^{ks_r} = 0 \quad \forall k \in K, w \in V_N, \tag{2.10}$$

$$u_i^k - u_j^k + 1 \leq (n+1)(1 - x_{ij}^k) \quad \forall k \in K, i \in V_L, j \in V_R, \tag{2.11}$$

$$u_i^k - u_w^k + 1 \leq (n+1)(1 - y_{iw}^{ks_l}) \quad \forall k \in K, i \in V_L, w \in V_N, s_l \in S_L, \tag{2.12}$$

$$u_w^k - u_j^k + 1 \leq (n+1)(1 - \tilde{y}_{wj}^{ks_r}) \quad \forall k \in K, j \in V_R, w \in V_N, s_r \in S_R, \tag{2.13}$$

$$u_i^k - u_j^k + 1 \leq (n+1)(1 - p_{ij}^k) \quad \forall k \in K, i \in V_L, j \in V_R, \tag{2.14}$$

$$u_i^k - u_j^k \geq 1 - (n+1)p_{ij}^k \quad \forall k \in K, i \in V_L, j \in V_R, \tag{2.15}$$

$$\begin{aligned} &4z_{aiwje}^k - (p_{ai}^k + p_{je}^k) \\ &\leq \sum_{s_l \in S_L} y_{aw}^{ks_l} + \sum_{s_r \in S_R} \tilde{y}_{we}^{ks_r} \quad \forall k \in K, a \in V_L, i, w, j \in V_N, e \in V_R, \end{aligned} \tag{2.16}$$

$$\begin{aligned} &3 + z_{aiwje}^k - (p_{ai}^k + p_{je}^k) \\ &\geq \sum_{s_l \in S_L} y_{aw}^{ks_l} + \sum_{s_r \in S_R} \tilde{y}_{we}^{ks_r} \quad \forall k \in K, a \in V_L, i, w, j \in V_N, e \in V_R, \end{aligned} \tag{2.17}$$

$$a_j^{ks_n} = a_j^k \quad \forall k \in K, j \in V_R, \tag{2.18}$$

$$M(x_{ij}^k - 1) + a_i^k + \frac{t_{ij}}{|S| - 1} \leq a_j^{k1} \quad \forall k \in K, i \in V_L, j \in V_R, \tag{2.19}$$

$$M(x_{ij}^k - 1) + a_j^{ks} + \frac{t_{ij}}{|S| - 1} \leq a_j^{k,s+1} \quad \forall k \in K, i \in V_L, j \in V_R, s \in S_L \cap S_R, \tag{2.20}$$

$$M(x_{ij} + y_{iw}^{k0} - 2) + a_i^k + \bar{t}_{iw} \leq a_w^k \quad \forall k \in K, i \in V_L, w \in V_N, j \in V_R, s_l \in S_L, \tag{2.21}$$

$$M(x_{ij} + y_{iw}^{ks} - 2) + a_j^{ks} + \bar{t}_{ij}^{ws} \leq a_w^k \quad \forall k \in K, i \in V_L, w \in V_N, j \in V_R, s \in S_L \cap S_R, \tag{2.22}$$

$$M(x_{ij} + \tilde{y}_{wj}^{ks_r} - 2) + a_w^k + \bar{t}_{ij}^{ws_r} \leq a_j^{ks_r} \quad \forall k \in K, i \in V_L, w \in V_N, j \in V_R, s_r \in S_R, \tag{2.23}$$

$$\begin{aligned} &\overline{d}_{ij}^{ws_l}(x_{ij} + y_{iw}^{ks_l} - 1) \\ &+\overline{d}_{gh}^{ws_r}(x_{gh} + \tilde{y}_{gh}^{ks_r} - 1) \leq \mathcal{E} \quad \forall k \in K, i, g \in V_L, w \in V_N, \\ &\qquad j, h \in V_R, s_l \in S_L, s_r \in S_R, \end{aligned} \tag{2.24}$$

$$2 + s_{iwj}^k \geq x_{ij}^k + \sum_{s_l \in S_L} y_{iw}^{ks_l} + \sum_{s_r \in S_R} \tilde{y}_{wj}^{ks_r} \quad \forall k \in K, i \in V_L, w \in V_N, j \in V_R, \tag{2.25}$$

$$3s_{iwj}^k \leq x_{ij}^k + \sum_{s_l \in S_L} y_{iw}^{ks_l} + \sum_{s_r \in S_R} \tilde{y}_{wj}^{ks_r} \quad \forall k \in K, i \in V_L, w \in V_N, j \in V_R, \tag{2.26}$$

$$\begin{aligned} &\sum_{a \in V_L} \sum_{w \in V_N} \sum_{e \in V_R} z_{aiwje}^k \\ &+ \sum_{w \in V_N} \sum_{\substack{s_l \in S_L, \\ s_l \leq s}} y_{iw}^{ks_l} + \sum_{w \in V_N} \sum_{\substack{s_r \in S_R, \\ s_r > s}} \tilde{y}_{wj}^{ks_r} \\ &- \sum_{w \in V_N} s_{iwj}^k + M(x_{ij}^k - 1) \leq D \quad \forall k \in K, i \in V_L, j \in V_R, s \in S_L. \end{aligned} \tag{2.27}$$

In the VRPDERO, the objective is to minimize the makespan τ. Based on the arrival time at the depot, i. e., a_{n+1}^k, constraints (2.3) provide lower bounds on the makespan for each truck k (and its assigned drones). The set of constraints (2.4)–(2.6) form the preservation of flow for the trucks. More precisely, constraints (2.4) state that each truck $k \in K$ may depart from the depot at most once. For any truck that has departed, constraints (2.5) guarantee that the truck must return exactly once. Constraints (2.6) ensure the preservation of flow for each truck. According to constraints (2.7), each customer must be served exactly once by either a truck or a drone. Synchronization between the route of a truck and drone operations is required through constraints (2.8)–(2.9). More precisely, these constraints guarantee that drone operations can only occur at discrete points associated with vertices that are visited by the respective truck. For each customer that is served by a drone, constraints (2.10) state that the flow of drones must be preserved. Constraints (2.11)–(2.13) are Miller-Tucker-Zemlin type constraints (Miller et al., 1960) that link the variables u_i^k to the decision variables x_{ij}^k, $y_{iw}^{ks_l}$, and $\tilde{y}_{iw}^{ks_l}$. Similarly, constraints (2.14)–(2.15) guarantee the right precedence relations through variables p_{ij}^k with regard to u_i^k and u_j^k. Constraints (2.16)–(2.17) determine if circumjacent deliveries occur (see also Figure 2.3.1). For a circumjacent delivery to occur, a drone must be launched before a truck k has reached i and retrieved after the same truck has reached j. Constraints (2.18)–(2.23) determine valid bounds on the temporal variables a_i^k and a_j^{ks}. Here, constraints (2.18) state that the arrival time en route to j at $s_r = s_n$ equals the time a_j^k at the vertex itself. Furthermore, for any arc that is used by the truck, constraints (2.19) and (2.20) provide lower bounds at each discrete step. Similarly, constraints (2.21)–(2.22)

determine arrival times for any customer that is served by a drone. Finally, constraints (2.23) guarantee valid bounds whenever a drone is retrieved by a truck. Constraints (2.24) impose a maximum distance (endurance) constraint on every possible drone operation. For drone operations that involve a single arc (i, j), constraints (2.25)–(2.26) force the variables s^k_{iwj} to take value 1. Finally, constraints (2.27) guarantee that during the transition of a truck on arc (i, j), at each discrete step, a maximum of D drones may be in operation simultaneously. To this end, we account for the following:

- circumjacent operations,
- the launch of any drone that is performed on the same arc up to step s,
- and any outstanding future recovery of a drone on the same arc.

To make this accounting correct, we must prevent counting the same operation twice (once as launch up to s and once as a future recovery). Therefore, we subtract any operation s^k_{iwj} that starts and ends on this arc (i, j) (see also Figure 2.3.2 and Table 2.3.2).

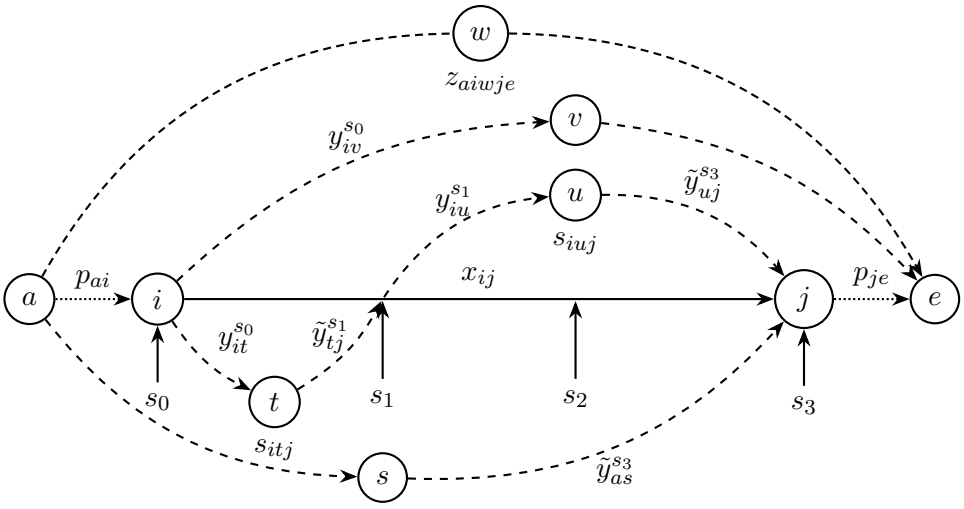

Figure 2.3.2: A visualization of some of the decision variables for a given truck k. The solid and dotted lines refer to the path of the truck. The dashed lines indicate deliveries made by drones, which are assigned to the truck.

2.3.2 Valid Inequalities

In solving a challenging combinatorial optimization problem, a well-known approach for enhancing the performance of a MILP solver consists in using suitable VIEQs. In this context, (Schermer et al., 2018a) provides several sets of effective VIEQs for solving the VRPD. Due to their success, we adapt them to the VRPDERO and improve them for

Table 2.3.2: This table illustrates how the number of operations is accounted for based on constraint (2.27) and Figure 2.3.2. The first column shows the step s on arc (i, j). The remaining columns show the number of circumjacent deliveries surrounding (i, j) ($\sum z$), launches up to s_i on (i, j) ($\sum y$), future recoveries after s_i on (i, j) ($\sum \tilde{y}$) and the negative number of operations that involve the single arc (i, j) ($\sum s_{iwj}$). The last column shows the left-hand side (LHS) according to (2.27), i. e., the summation over columns 2 to 5 in the respective row of the table. In this case, the truck needs to be equipped with $D \geq 4$ drones to perform the operations shown in Figure 2.3.2.

s	$\sum z$	$\sum y$	$\sum \tilde{y}$	$-\sum s_{iwj}$	LHS according to (2.27)
s_0	1	2	3	-2	4
s_1	1	3	2	-2	4
s_2	1	3	2	-2	4

providing sharper lower bounds that strengthen the relaxation of the formulation and restrict the solution space of the problem while excluding none of the optimal solutions.

Proposition 2.1. In the MILP (2.2)–(2.27), τ is bounded by the time spent traveling by each truck.

Proof: Let τ^k be the value which indicates the total time that the truck $k \in K$ spends traveling on arcs. It can be computed as follows:

$$\tau^k = \sum_{i \in V_L} \sum_{j \in V_R} x_{ij}^k t_{ij} \quad \forall k \in K. \tag{2.28}$$

In any VRPDERO solution, a_{n+1}^k, i. e., the earliest possible arrival time of the truck k and all drones that are assigned to it at depot, is bounded by the arrival of the truck itself. For each truck k, this value is a lower bound on τ according to constraints (2.3). Indeed, since it might happen that the truck k waits at some vertices (or at discrete points on arcs) for the arrival of a drone, we can use τ^k as a lower bound on a_{n+1}^k. Consequently, we can obtain the following VIEQs:

$$\sum_{i \in V_L} \sum_{j \in V_R} x_{ij}^k t_{ij} \leq \tau \quad \forall k \in K, \tag{2.29}$$

which completes the proof of Proposition 2.1. ■

In the MILP (2.2)–(2.27), drone operations are characterized implicitly. More precisely, the variables $y_{iw}^{ks_l}$ (respectively, $\tilde{y}_{wj}^{ks_l}$) state that a drone operation starts at s_l en route from i to serve a customer w (respectively, ends at s_r en route to j after a delivery to w). However, only through the consideration of the binary variables x_{ij} do we know which vertices are visited after i and before j. This information is crucial for the determination of the position of discrete steps and, therefore, travel times of drones. Hence, for the

general case of $s_n > 2$, we propose the inclusion of auxiliary variables and constraints to the MILP (2.2)–(2.27) as it is suggested by the following Lemma.

Lemma 2.1. Assume that y^{ks}_{iwj}, where $k \in K, i \in V_L, w \in V_N, j \in V_R, s \in S_L \cap S_R$, are binary variables stating whether a drone that belongs to the truck k travels from s on (i, j) to w or in reverse. Then, the following inequalities (2.30)–(2.31) are valid for the MILP (2.2)–(2.27):

$$2y^{ks}_{iwj} \leq x^k_{ij} + y^{ks}_{iw} + \tilde{y}^{ks}_{wj} \quad \forall i \in V_L, w \in V_N, j \in V_R, s \in S_L \cap S_R, \tag{2.30}$$

$$1 + y^{ks}_{iwj} \geq x^k_{ij} + y^{ks}_{iw} + \tilde{y}^{ks}_{wj} \quad \forall i \in V_L, w \in V_N, j \in V_R, s \in S_L \cap S_R. \tag{2.31}$$

Proof: These inequalities guarantee that y^{ks}_{iwj} is forced to value 1 if two conditions apply. First, arc (i, j) must be a part of the truck k's route. Second, either a delivery to w must be initiated by a drone or a drone must be retrieved after a delivery to w en route at s on arc (i, j). These constraints are valid with respect to MILP (2.2)–(2.27), because a drone launched to customer w cannot be retrieved at the same location from which it was launched (see Section 2.3.1 and MILP (2.2)–(2.27)). Therefore, the sum over y^{ks}_{iw} and $\tilde{y}^{ks}_{wj}$ can be at most one. Furthermore, an optimal solution vector x^* to MILP (2.2)–(2.27) is guaranteed to be feasible subset of an optimal solution vector to MILP (2.2)–(2.27), (2.30)–(2.31). Thus, the value of the optimal solution does not change. This completes the proof of Lemma 2.1. ■

Proposition 2.2. In the MILP (2.2)–(2.27), τ is bounded by the average time spent traveling by drones.

Proof: We distinguish two cases: whether $|S| = 2$ ($s_n = 1$), i. e., there is no en route operation, and $|S| \geq 3$ ($s_n \geq 2$), i. e., in presence of en route operations.

(i) For $|S| = 2$, we have the following case: Since $|S| = 2$, hence, $S = \{0, 1\}$ and there are no en route operations. We define τ^k_i as the total time that the i-th drone, which is assigned to truck $k \in K$, spends traveling on arcs of the underlying graph. Then, we can compute the total time spent traveling by the D drones as follows:

$$\sum_{i=1}^{D} \tau^k_i = \Big(\sum_{i \in V_L} \sum_{w \in V_N} \bar{t}_{iw} y^{ks_0}_{iw} + \sum_{w \in V_N} \sum_{j \in V_R} \bar{t}_{wj} \tilde{y}^{ks_n}_{wj}\Big) \quad \forall k \in K. \tag{2.32}$$

Let $\tau^k_{max} = max(\tau^k_1, \ldots, \tau^k_D)$, i. e., τ^k_{max} is the maximum time that a single drone, associated to a truck $k \in K$, spends traveling. Since a drone might also have to wait at a vertex or an arc in order to be retrieved by the truck, the makespan τ, at which all drones

and the truck have returned to the depot, cannot be smaller than τ^k_{max}, that is:

$$\tau^k_{max} \leq \tau \quad \forall k \in K. \tag{2.33}$$

As we do not explicitly account for the travel time of each drone in MILP (2.2)–(2.27), instead, we consider $\overline{\tau}^k_D$, i. e., the average arrival time of the set of drones D associated to truck k. Then, in the given set of D drones, $\overline{\tau}^k_D$ must always be less than or equal to the earliest arrival time τ^k_{max}, that is:

$$\overline{\tau}^k_D = \frac{1}{D}\sum_{i=1}^{D} \tau^k_i \leq \tau^k_{max} \leq \tau \quad \forall k \in K. \tag{2.34}$$

If we divide both sides of (2.32) by D, i. e., the number of drones, and take into account the inequalities (2.34), we obtain the following VIEQs that provide a lower bound on the makespan τ:

$$\frac{1}{D}\Big(\sum_{i\in V_L}\sum_{w\in V_N} \bar{t}_{iw} y^{ks_0}_{iw} + \sum_{w\in V_N}\sum_{j\in V_R} \bar{t}_{wj} \tilde{y}^{ks_n}_{wj}\Big) \leq \tau \quad \forall k \in K. \tag{2.35}$$

(ii) For $|S| \geq 3$, the more general case applies: Here, the relaxation provided by (2.35) becomes weak because we only account for operations that occur directly on vertices, i. e., at $s \in \{0, s_n\} \subset S$. Through the introduction of the binary variables y^{ks}_{iwj} and the inequalities (2.30) and (2.31) in Lemma 2.1, we can improve the bounds given in (2.35). More precisely, for any drone operation that either starts or ends at a discrete step $s \in S_L \cap S_R$ and does not coincide with a vertex, we account for the time spent traveling by all drones as follows for each truck:

$$\sum_{i\in V_L}\sum_{w\in V_N}\sum_{j\in V_R}\sum_{s\in S_L\cap S_R} \bar{t}^{ws}_{ij} y^{ks}_{iwj} \quad \forall k \in K. \tag{2.36}$$

By analogously following the structure of proof shown for $|S| = 2$, we can consider (2.36) as an additional term to (2.35) that tracks the total time spent performing en route drone operations. This yields the following more general VIEQs:

$$\frac{1}{D}\sum_{w\in V_N}\Big(\sum_{i\in V_L} \bar{t}_{iw} y^{ks_0}_{iw} + \sum_{j\in V_R} \bar{t}_{wj} \tilde{y}^{ks_n}_{wj} + \sum_{i\in V_L}\sum_{j\in V_R}\sum_{s\in S_L\cap S_R} \bar{t}^{ws}_{ij} y^{ks}_{iwj}\Big) \leq \tau \quad \forall k \in K, \tag{2.37}$$

which is the desired conclusion of Proposition 2.2. ■

2.4 Variable Neighborhood Search for Solving the VRPDERO

As we will see in more detail in Section 2.5, only small-sized problem instances can be solved to optimality by a commercial state-of-the-art solver within reasonable runtime. In order to overcome this issue, in this section, we introduce an algorithm which exhibits features of VNS and TS for solving the VRPDERO.

VNS is a metaheuristic framework that was introduced by Mladenović and Hansen (1997) for solving (combinatorial) optimization problems. Given an incumbent solution x, the strategy of VNS consists in systematically visiting increasingly distant k-th neighborhoods of the current incumbent solution x, i. e., $\mathcal{N}_k(x)$. The new neighborhoods are then explored by a local search method in order to identify the local optima. A new solution is accepted if and only if an improvement is made. Several extensions to the basic VNS have been proposed and an overview of the proposed methods is given by Gendreau and Potvin (2010) and Pierre Hansen et al. (2017).

TS was introduced by Glover (1986) as a metaheuristic for guiding and controlling heuristics. The basic idea is to identify certain moves as forbidden, i. e., *tabu*, to prevent cycling. Glover (1986) differentiates *strong inhibition*, i. e., if a move is identified as tabu then it will never be chosen in the future (e. g., found in branch-and-bound methods) from *weak inhibition*, which consists of a short-term memory.

The basic structure of our proposed procedure is shown in Algorithm 2.1. A detailed explanation of all components is given in the subsequent sections. Indeed, this framework structurally reflects the basic VNS by Mladenović and Hansen (1997) that consists of an *initialization*, a *shaking* step that moves the current solution x to the k-th Neighborhood $\mathcal{N}_k(x)$, and a *local search* procedure. For adapting VNS to the VRPDERO, we propose decomposing the problem into two natural components: *route generation* and *drone operation insertion*. In our case, the local search procedure is a problem specific operator that, given a routing x (that consists of a route for each truck), generates a valid VRPDERO solution y through the insertion of drone operations (line 9 of Algorithm 2.1). Furthermore, in order to avoid cycling, a tabu list is implemented in the algorithm (lines 6-7 of Algorithm 2.1).

Finally, the *neighborhood change* operator shown in Algorithm 2.2 handles value updates during each iteration. In case of an improvement (or an equally good objective value), we store the incumbent solution y^* and the routing x associated with this solution. Furthermore, as we will see in Sections 2.4.2 and 2.4.3, the local search operator is a deterministic procedure. Therefore, we add x to a tabu list that will prevent future evaluations of the same routing. Note that, in principle, different routings x may lead to the same VRPDERO solution y after the drone insertion procedure. Therefore, an equally

good objective value also triggers a change in the incumbent value to enable an effective exploration of the search space.

Algorithm 2.1 Adaptation of basic VNS.

```
1: x ← Initialization()                                  ▷ refer to Section 2.4.1
2: while The termination criterion is not met do
3:     k ← 1
4:     while k < k_max do
5:         x' ← Shake(x, k)
6:         if TabuList.contains(x') then
7:             Continue
8:         end if
9:         y ← LocalSearch(x')                           ▷ refer to Sections 2.4.2–2.4.3
10:        x, y, k ← NeighborhoodChange(x', y, k)        ▷ refer to Algorithm 2.2
11:    end while
12: end while
13: return y
```

Algorithm 2.2 An overview of the neighborhood change.

```
1: if f(y) ≤ f(y*) then
2:     y* ← y
3:     x ← x'
4:     k ← 1
5:     TabuList.add(x)
6: else
7:     k ← k + 1
8: end if
```

In what follows, the integrated components of Algorithm 2.1 will be described in more detail. More precisely, in Section 2.4.1 we demonstrate the procedure that is used for the *initialization*, *local search*, and *shaking*. Section 2.4.2 is devoted to the presentation of the drone insertion procedure. As the drone insertion procedure is a deterministic search operator, we embed it within a D&C approach that is well-suited for parallel computing architectures. This approach is presented in Section 2.4.3. We conclude this part in Section 2.4.4 with further remarks on the heuristic procedure and on some potential pitfalls that might be encountered.

2.4.1 Route generation

During the initialization procedure, first, a set of routes is generated through the parallel savings heuristic (Clarke and Wright, 1964). As trucks are by definition not associated

with a capacity in the VRPDERO, we generate the routes in such a way that each route may contain at most $\lceil |V_N|/|K| \rceil$ customers.

Then, for a limited number of iterations, a local search procedure is applied. Within this procedure, we employ the *2-opt*, *String Relocation* (SR), and *String Exchange* (SE) operators (Breedam, 1994; Lin and Kernighan, 1973). More precisely, the 2-opt operator inverts a *randomly* chosen subsequence of a route, SR involves a relocation of a *random* customer from one route to another route and SE invokes an exchange of two *random* customers from two distinct routes. When a vertex is moved through SR or SE, we insert it at the position that will add the least additional length to the affected route. A 2-opt move is only applied, if the move will decrease the length of the original route. Similarly, SR and SE are only applied if the move leads to a reduction of the maximum length of the routes that are involved in the move.

For shaking, we also use the 2-opt, SR, and SE operators that we described above. More precisely, for k iterations, first, a random move is selected. Then, one route (or two routes, in case of SR or SE) is selected and the move applied. However, as opposed to local search, we also accept an increase in the route length after the application of any move.

2.4.2 Drone Insertion

So far, we did not consider drone operations during the construction of the routes, local search, or shaking. At this point, we are interested in a procedure that transforms a set of routes into a VRPDERO solution through introduction of drone operations. The drone operations should be inserted within each route in a manner, such that the makespan is minimized. For the insertion of drones in an existing route, we apply the following procedure for each route π represented by the vector x (see Algorithm 2.1). Assume that each route $\pi = \{0, \ldots, n+1\}$ is given as a permutation of vertices. Set $V_L = \pi \setminus \{n+1\}$, $V_R = \pi \setminus \{0\}$, and $V_N = V_L \cap V_R$. Then, we perform a greedy search strategy that iteratively looks for a drone operation (i, s_l, w, s_r, j) (see Section 2.3.1 for its definition) that satisfies the following criteria:

1. A significant reduction in the length of the truck's route is achieved through the assignment of a customer, which is currently served by the truck, to a drone.

2. The *temporal delay*, that is incurred at the retrieval location through the synchronization requirement, is minimized.

The first criterion can be evaluated very quickly and is guided by the following formula, where d_{ij} is the distance between nodes i and j (see Section 2.3.1), w is the customer associated with the drone operation, and w_p and w_s are the direct predecessor and successor

of w in the route π:

$$\gamma(w) \coloneqq d_{w_p,w_s} - d_{w_p,w} - d_{w,w_s} \tag{2.38}$$

Clearly, the first criterion only depends on the customer w associated with the drone operation (i, s_l, w, s_r, j). Therefore, the second criterion integrates the launch and retrieval locations (i, s_l) and (s_r, j), respectively. More precisely, the second criterion considers the temporal difference in arrival time between the truck and drone at the retrieval location (that is defined by (s_r, j) in the current operation) after the assignment of customer w to a drone. In order to calculate this value, given the starting time defined at (i, s_l), we calculate the arrival time of the truck and drone, denoted by $a_j^{s_r}$ and $\overline{a}_j^{s_r}$, respectively. The temporal delay is then calculated as an absolute value as follows:

$$\lambda(i, s_l, w, s_r, j) = |a_j^{s_r} - \overline{a}_j^{s_r}| \tag{2.39}$$

Based on this formula, a large value indicates that either the drone must wait for the truck or the truck must wait for the drone for a long time. In the first case, if a drone arrives much earlier than a truck, it is unlikely that a good utilization will be achieved as the drone spends much time waiting. For such cases, it can be beneficial to look for an earlier retrieval location such that the temporal delay is reduced and, in this sense, fast synchronization achieved. In the second case, if a truck arrives much earlier than a drone, it must wait for the drone to arrive before it can continue its tour. Therefore, it can be worthwhile to investigate if the drone cannot be retrieved at a later point in the tour such that time spent waiting by the truck is turned into time spent moving.

Given an initial route, these criteria are then embedded in an algorithm as follows. First, we look for $w^* = \arg\max_w \gamma(w)$. Note that w^* must be feasible, i. e., it does not serve as a launch or retrieval location for any previously chosen operation. Afterwards, the operation $(i, s_l, w, s_r, j)^* = \arg\min_{(i,s_l,w,s_r,j)} \lambda(i, s_l, w, s_r, j)$ is identified (for the domain of (i, s_l, w, s_r, j), refer to Section 2.3.1). This operation must be feasible with respect to the drone's endurance, the number of drones in operation at any point, and any other links between the route of the truck and drone operations that are implied by MILP (2.2)–(2.27). As we will see in the following section, an operation may also be prohibited, i. e., tabu. If the operation is feasible and not tabu, we add it to the list of operations and iterate until no more feasible or improving operation can be found.

2.4.3 Divide and Conquer

The drone insertion procedure that was introduced in the previous section is summarized in Algorithm 2.3. This procedure is a very fast and deterministic operator that performs

Algorithm 2.3 A summary of the drone insertion heuristic.

```
1: while the termination criterion is not met do
2:     for customer w with maximum γ(w) in the route π do
3:         for operation (i, s_l, w, s_r, j) with minimum λ(i, s_l, w, s_r, j) that is feasible do
4:             if the operation is not tabu (see Section 2.4.3) then
5:                 insert (i, s_l, w, s_r, j) and remove w from π
6:             end if
7:         end for
8:     end for
9: end while
```

a greedy search that is guided by the criteria γ and λ. While the results returned by this procedure are solid in many cases, it may be of interest to perform a more thorough and in-depth exploration of possible drone operations. To this end, we propose a search space, called $\mathcal{N}(1)$ search space, where the neighborhood relationship is a single drone operation, i. e., two neighboring solutions differ through exactly one drone operation. For the exploration of the $\mathcal{N}(1)$ search space, we embed the drone insertion procedure in a D&C approach (for a general description of a D&C algorithm, see, e. g., (Cormen et al., 2014) and references therein).

In order to describe our D&C approach, consider the rooted tree shown in Figure 2.4.1. In this tree, the root node is associated with the route π of a single truck where no drone operations occur. The children of each node then refer to operations that are inserted into π in a sequential manner. The depth of a node directly corresponds to the number of operations that have been inserted.

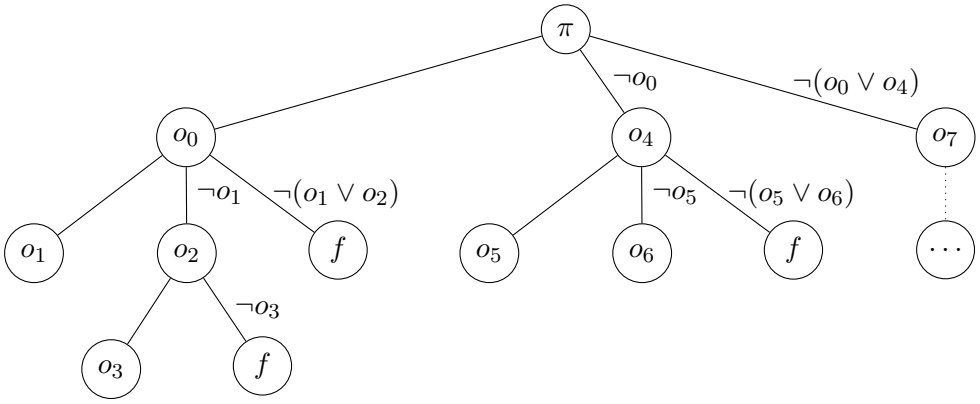

Figure 2.4.1: A sample divide-and-conquer search tree in an $\mathcal{N}(1)$ search space.

Within the D&C approach, first, we apply the drone insertion procedure (Algorithm 2.3) which generates the leftmost branch. This branch indicates that o_0 is the first operation

inserted in the route, afterwards o_1 is inserted, and then, no more operation can be found. As the leaf node o_1 has no children, we say that this node has been fully *fathomed*. After the generation of the initial branch, we follow a D&C approach and explore the tree in a *depth-first* manner. More precisely, after the initial (leftmost) branch has been generated, we start with node o_1, that has been fathomed, and move to its parent node. Then, we generate a new branch where the operation at depth 2 of the tree is prohibited to be o_1, i. e., $\neg o_1$, which generates the node o_2 and its child o_3 through Algorithm 2.3. This procedure is iterated according to Algorithm 2.4 until a termination criterion is met or the tree fully explored.

Algorithm 2.4 Depth-first search in the $\mathcal{N}(1)$ search space.

```
1: Root r ← π
2: r.addBranch()                                   ▷ refer to Algorithm 2.3
3: Node n = r.getLeaf()
4: while Termination criteria not met do
5:     if n.degree == 0 then
6:         n.fathomed ← true
7:     end if
8:     Node p = n.getParent()
9:     if (p.getRightMostChild().Operation == {}) then
10:        p.fathomed ← true
11:    end if
12:    while p.fathomed == true do
13:        p = p.getParent()
14:    end while
15:    n = p.addRightMostChild()
16:    n.addTabus(n.Siblings)
17:    n.addOperations(n.Ancestors)
18:    n.addBranch()                               ▷ refer to Algorithm 2.3
19: end while
```

2.4.4 Comments

In essence, this D&C procedure is well-suited to implementations that utilize parallel-computing architectures (see, e. g., (Kumar and V. N. Rao, 1987; N. N. Rao and Kumar, 1987)). More precisely, after the initial leftmost branch has been generated, branching can start on multiple nodes. This procedure can then be iterated whenever unexplored nodes are available, yielding a *parallel depth-first search* procedure. However, for the purpose of our preliminary study, we chose a more simple sequential depth-first search approach.

As no nodes are pruned in our procedure, in principle, we greedily search the complete

operation space where the neighboring relation between two nodes is a single operation. This enumeration is easily possible for small instances that can be associated with small (in terms of vertices) routes at the root node. However, for large initial routes and an increase in the maximum permitted steps s_n, the size of the tree may become prohibitively large. In such a case, for providing a heuristic exploration of the tree, it is easily possible to introduce some criteria that limit the effort spent on the search. Possible criteria that limit the overall size of the tree include, e. g., limits on the number of nodes that have been processed in the tree, or a limit on the degree of each node before we consider it to be fathomed. As an alternative to restricting the size of the tree, it is also possible to reduce the time that is spent processing each node in the tree. More precisely, additional logic might be added to Algorithm 2.3 such that only a subset of drone operations is considered in the first place. Given the initial route π, let i_{id}, w_{id}, and j_{id} refer to the indices of vertices i, w, and j in π. Then, one possibility is to only consider those operations, where $|w_{id} - i_{id}|$ and $|j_{id} - w_{id}|$ do not exceed a given threshold. Typically, in optimal solutions, retrieval of drones occurs almost immediately after the launch such that a high utilization is guaranteed. Finally, a more general criterion might simply limit the runtime spent on the exploration of the tree.

It is also important to consider that the $\mathcal{N}(1)$ search space may prevent the (globally) optimal solution to be found in some cases, if the feasibility of an operation is considered during each move in Algorithm 2.3. To illustrate this case, consider Figure 2.4.2, which shows an initial route (left). Assume that we have a single drone with limited endurance and the optimal drone operations are (a, s_0, b, s_1, d) and (a, s_1, c, s_2, d) (right). However, given the initial route and the $\mathcal{N}(1)$ search space, it is not possible to reach the optimal solution without visiting an intermediate solution that is infeasible. Conversely, the insertion of an operation that is feasible by itself can render existing operations, that were previously deemed feasible, infeasible. Therefore, careful attention is required when the feasibility of a solution is checked and a full exploration is only guaranteed if the feasibility is verified on a node-level in the rooted tree instead of a per-move basis in Algorithm 2.3. However, such delayed validation of operational feasibility (i. e., on a node instead of per-move basis) will increase the size of the tree.

Any increase in discrete steps has significant influence on number of possible drone operations and consequently, the size of the tree and the potential degree of its nodes. In the MILP (2.2)–(2.27), the number of possible drone operations, which are reflected by combinations of variables $y_{iw}^{ks_l}$ and $\tilde{y}_{wj}^{ks_r}$, grows in the order of $\mathcal{O}(|S|^2)$ with the amount of discrete steps. As the route is assumed to be fixed when solving the drone insertion, given s_n, a route $\pi = \{0, a, \ldots, b, n{+}1\}$ with $\pi \backslash \{0, n{+}1\} \in V_N \backslash \{w\}$, and a vertex $w \in V_N$ that is set to be served by drone, the number of potential operations can be calculated according

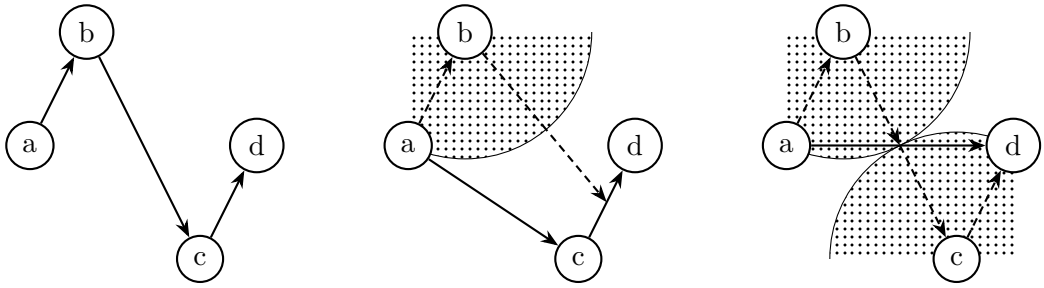

Figure 2.4.2: An initial route (left) and the optimal solution that features two drone operations (right). In this case, the optimal solution can only be constructed by generating an intermediately infeasible solution in the $\mathcal{N}(1)$-neighborhood (center). In this case, we assume that $s_n = 2$, i. e., we have $S = \{0, 1, 2\}$ which implies an available discrete point at the center of each arc.

to formula (2.40), in which the first underbrace accounts for the drone operations that are performed on single arcs and the second underbrace accounts for all drone operations that involve multiple arcs. As an example, Figure 2.4.3 visualizes the number of possible operations for a route $\pi^k = \{g, h, j\}$ with just three vertices and $s_n = 2$. In this case, for $s_n = 1$, we have three possible operations that are highlighted in red. For $s_n = 2$, the number of possible operations already increases to 10 and, clearly, this amount grows not just with an increase in s_n but also with respect to the value of $|\pi^k|$.

$$\sum_{i=1}^{|\pi|-1} \underbrace{\Big(\sum_{s_l \in S_L} \sum_{\substack{s_r \in S_R \\ s_r > s_l}} 1\Big)}_{\text{single arc}} + \sum_{i=1}^{|\pi|-2} \underbrace{\big((|\pi| - 1 - i) \sum_{s_l \in S_L} \sum_{s_r \in S_R} 1\big)}_{\text{multiple arcs}} \tag{2.40}$$

2.5 Computational Experiments and their Numerical Results

We carried out two main classes of computational experiments, on small- and large-scale instances, and we present the numerical results in this section. More precisely, in Section 2.5.1 we discuss the performance of Gurobi Optimizer in solving MILP (2.2)–(2.27) and compare its performance to our proposed heuristic. Afterwards, Section 2.5.2 shows that the heuristic can also be used to solve larger instances.

All experiments were performed on single compute nodes in an Intel® Xeon® Gold 6126 cluster, where each node operated at 2.6 GHz with 16 GB of RAM (hyper-threading disabled). We implemented our algorithms in Java SE 8 and for solving MILPs, we used Gurobi Optimizer 8.1.0 (Gurobi Optimization, LLC, 2023). The solver was allowed to utilize all cores of the processors, whereas our heuristic ran single-threaded. Since there is

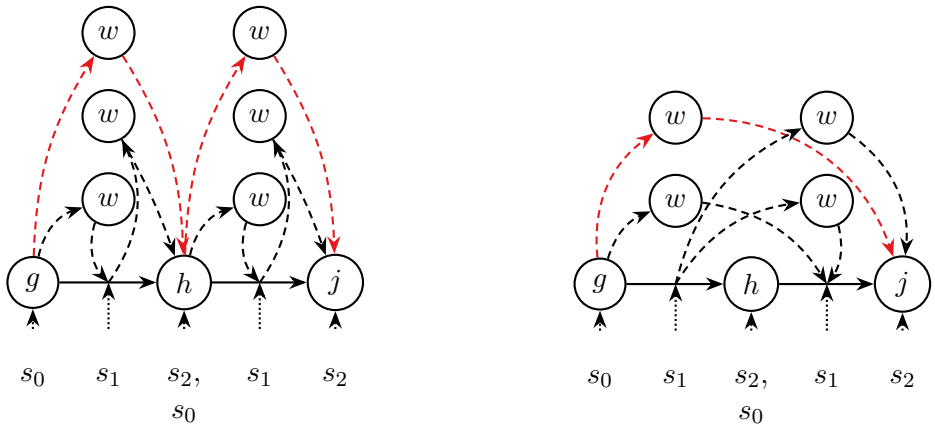

Figure 2.4.3: All possible drone operations for a given route $\pi^k = \{g, h, j\}$, a customer w that is served by drone and $s_n = 2$. The left-hand and right-hand side represent the first and second under brace in formula (2.40), respectively. The operations that are possible for $s_n = 1$ are highlighted in red.

no commonly accepted benchmark for the VRPD, we performed our experiments on the benchmark instances used by Agatz et al. (2018). The instances are specified by a graph $\mathcal{G}(V, E)$ (that determines the location of the depot and customers), the relative velocity α, and the endurance $\mathcal{E}$. In particular, through a *relative endurance* parameter $\mathcal{E}_r$, the endurance $\mathcal{E}$ can be calculated as follows, where $max(E)$ refers to the edge with maximum weight in the graph $\mathcal{G}$:

$$\mathcal{E} \coloneqq \mathcal{E}_r \cdot \max(E), \quad \mathcal{E}_r \in [0, 2]. \tag{2.41}$$

In this case, $\mathcal{E}_r = 0$ guarantees that no drone operation is possible while $\mathcal{E}_r = 2$ marks any operation as feasible. Furthermore, the trucks and drones are assumed to both follow the Euclidean distance metric. For different numbers of vertices, Agatz et al. (2018) provide 10 instances with a uniform distribution of customers. Each instance is associated with a value $\alpha \in \{1, 2, 3\}$ and a value $\mathcal{E}_r \in \{0.2, 0.4, 0.6, 1.0, 1.5, 2.0\}$ yielding a total of $10 \cdot 3 \cdot 6 = 180$ different instances for a given size (i. e., $|V|$). Furthermore, by definition of Poikonen et al. (2017) and X. Wang et al. (2017), the number of trucks and drones per truck are parameters in the VRPD (and therefore, also in the VRPDERO). As a meaningful selection, we limit ourselves to $|K| \in \{1, 2\}$ and $|D| \in \{1, 2\}$, i. e., we either have one or two trucks that are equipped with either one or two drones each. In addition, to investigate the benefits of en route operations, we consider $s_n \in \{1, 2\}$, i. e., we either have 0 or 1 discrete steps on each arc, which can be used for en route operations. Figure 2.5.1 shows a sample VRPDERO solution.

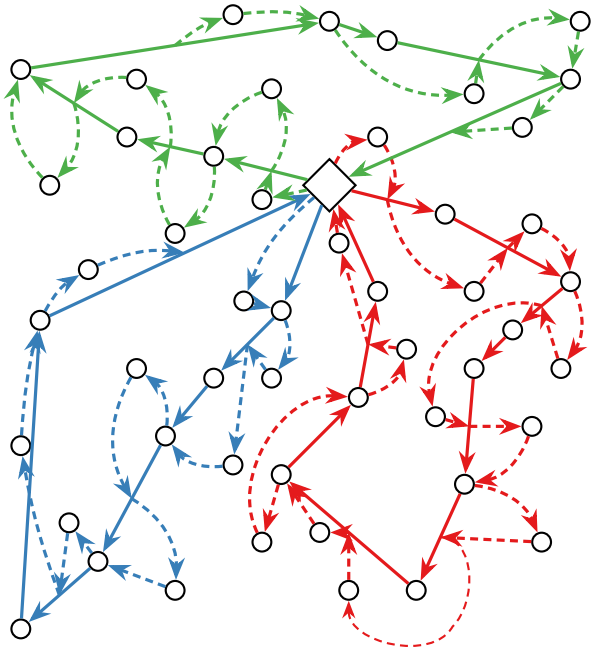

Figure 2.5.1: A sample VRPDERO solution on a problem instance with 50 customers, $|K| = 3$, $D = 1$, and $s_n = 2$. The solid and dashed lines represent the paths of the vehicles and drones, respectively.

2.5.1 Small Instances

For the purpose of studying the performance of the MILP solver and a comparison to our heuristic, we used instances with $|V| \in \{8, 9, 10\}$ vertices. For each size, we have ten different instances each of which is associated with three different values for the relative velocity α and six different values for the relative endurance $\mathcal{E}_r$. In total, this yields $3 \cdot 180 = 540$ instances. Based on the additional VRPDERO-specific problem parameters, i. e., the number of trucks, drones, and discrete steps, we considered a total of $2^3 \cdot 540 = 4320$ problem instances in our experiments. Sections 2.5.1.1 and 2.5.1.2 are dedicated to detailed numerical results that were obtained by solving these instances through the MILP solver and our heuristic.

2.5.1.1 Results obtained through the MILP Formulation

In this section, we provide the results that were obtained by solving the MILP introduced in Section 2.3.1 using the solver Gurobi Optimizer 8.1.0 (Gurobi Optimization, LLC, 2023). More precisely, we solved these 4320 instances in two different ways:

1. We solved the MILP (2.2)–(2.27) while we included the VIEQs (2.29)–(2.35) that

provide bounds on the makespan. For $s_n = 1$, we only included VIEQs (2.29) and (2.35) in the model. For the general case, where $s_n \geq 2$, we also took the variables y^k_{iwj} into consideration, enabling us to add constraints (2.30)–(2.31) to the model. Through these additional variables and constraints, the VIEQs (2.35) were replaced with the more general VIEQs (2.37).

2. In MILP (2.2)–(2.27), the cardinality of rows required for constraints (2.16)–(2.17) that bind the values of the z^k_{aiwje} variables is very large. Furthermore, given the underlying set of these variables, the typical subset of variables in the solution vector with binary value 1 is very sparse and thus, they rarely influence constraints (2.27). Therefore, these constraints might be a good candidate for a row-generation procedure. Following this logic, we also considered a case where these constraints were treated as *lazy constraints*. More precisely, whenever the solver detects a new candidate incumbent integer-feasible solution, we determine if any of constraints (2.16)–(2.17) are violated. If this is the case, the respective constraints are *lazily* added to the model, i. e., not before needed or only as necessary, and then, the solver's internal branch-and-cut procedure continues. For this case, we also included the VIEQs according to the previous case (as we previously described).

When solving the VRPDERO as MILP, we limited the runtime of the solver to 10 minutes. Note that, in principle, we could have also solved MILP (2.2)–(2.27) without the presence of any VIEQ. However, for all but trivial cases, this will lead to *Mixed-Integer Programming* (MIP) gaps that are close or equal to 100% after runtime. As this is not very revealing, we decided to exclude them from the final experiments and analysis.

For the analysis of our results, we introduce the following metric. The value Δ indicates the relative change in makespan through the introduction of drones and is calculated as follows:

$$\Delta \coloneqq 100\% - \frac{\text{objective value after runtime}}{\textit{optimal} \text{ objective value with } D = 0} \tag{2.42}$$

Note that the optimal objective value for $D = 0$ can be obtained within a few seconds by solving MILP (2.2)–(2.27).

The results are visualized in Figures 2.5.2 and 2.5.3; however, for the sake of completeness, the detailed numerical results are available in the Appendix 2.A (refer to Tables 2.A.1–2.A.6). Overall, the problem parameters such as, e. g., the relative endurance $\mathcal{E}_r$, the number of trucks $|K|$, the number of drones D, and the instance size have a significant influence on the runtime of the solver. A small relative endurance (i. e., $\mathcal{E}_r \in \{0.2, 0.4, 0.6\}$) will often allow the solver to find the optimal solution within the allocated runtime. Moreover, a change in the discrete steps might increase the runtime significantly. As an exam-

ple, even for the smallest instance ($n = 8$), the introduction of a discrete step on each arc can cause a nearly tenfold increase in runtime to find the optimal solution.

In general, treating constraints (2.16)–(2.17) as lazy constraints has some benefits in the sense that most instances can be solved faster to proven optimality or with smaller gaps within the runtime limit. Significant speed-ups are gained when there is no en route possibility, i. e., $s_n = 1$. However, this procedure quickly approaches the quality of the non-lazy case as the instance size increases and, as a result, the runtime limit is steadily approached by both approaches.

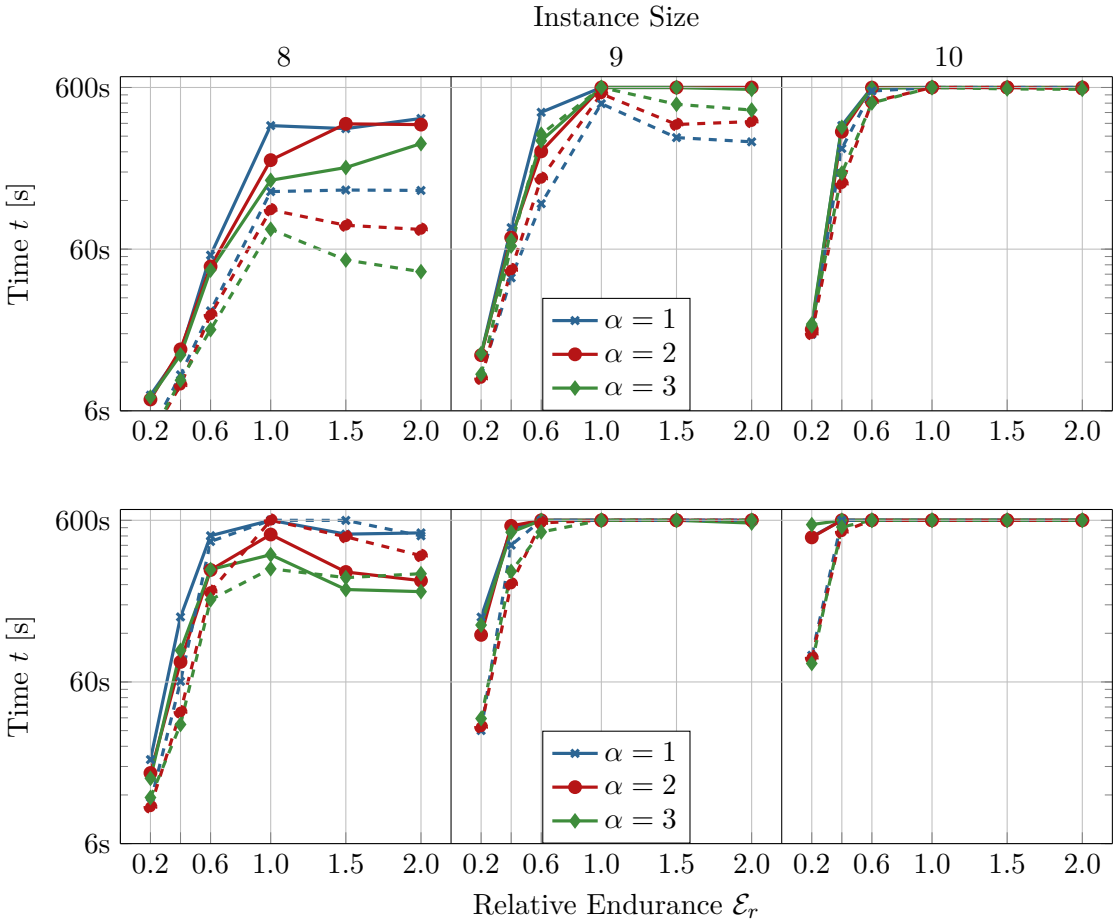

Figure 2.5.2: The average time t (shown on a logarithmic scale) required by the solver for different instance sizes, relative endurances $\mathcal{E}_r$, and relative velocities α. The lines are shown for $s_n = 1$ (upper figure) and $s_n = 2$ (lower figure) and a single truck equipped with a single drone (based on Tables 2.A.1 - 2.A.6). The solid lines represent the basic MILP and the dashed lines represent the case where lazy constraints were used.

Figure 2.5.2 visualizes the runtime of the solver for both approaches that were described at the beginning of this section. Furthermore, Figure 2.5.3 highlights the remaining MIP

gap after runtime. These figures show the cases of a single truck that is equipped with a single drone for $s_n = 1$ and $s_n = 2$, respectively.

A peak can often be detected for the runtime and MIP gap at $\mathcal{E}_r = 1$. This behavior is interesting as $\mathcal{E}_r = 1$ implies a much more constrained solution space than $\mathcal{E}_r \in \{1.5, 2.0\}$. Nevertheless, the structure of optimal solutions can differ in these cases and, therefore, so can the runtime to determine them.

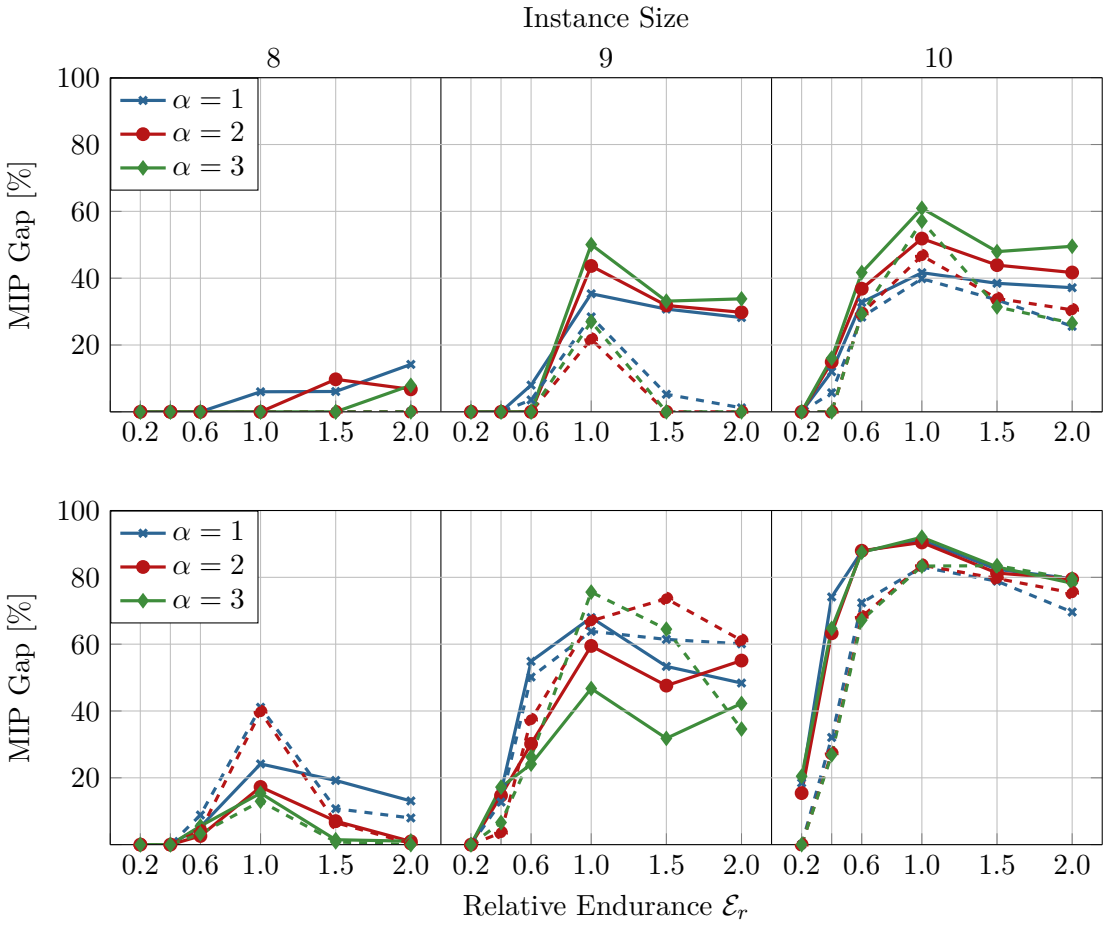

Figure 2.5.3: The average remaining relative MIP gap after runtime returned by the solver for different instance sizes, relative endurances $\mathcal{E}_r$, and relative velocities α. The lines are shown for $s_n = 1$ (upper figure) and $s_n = 2$ (lower figure) and a single truck equipped with a single drone (based on Tables 2.A.1 - 2.A.6). The solid lines represent the basic MILP and the dashed lines represent the case where lazy constraints were used.

Similar observations can be made as the number of drones per truck D or the number of trucks $|K|$ is increased. Typically, a single truck equipped with two drones can be tackled easier by the solver than the case of two trucks that are equipped with one drone each. Moreover, the case of two trucks equipped with two drones each has proven itself to be

the most difficult case. In all of these cases, generally, the average runtime and MIP gaps increase tremendously compared to the basic case of $|K| = D = 1$ (see Tables 2.A.1–2.A.6). Clearly, the number of columns and rows of MILP (2.2)–(2.27) are independent of the number of drones, i. e., D (only the right-hand side in constraints (2.27) is affected). Opposed to this, an increase in the number of trucks increases both, the number of variables and constraints, significantly.

To sum up, based on the recorded Δ values in Tables 2.A.1–2.A.6, notable makespan-savings are possible through the introduction of drones. These values vary significantly with the drone parameters and hence, care should be taken when designing a drone fleet in practice. Moreover, the introduction of a discrete step on each arc typically reduces the makespan by an additional 2 to 3% with respect to the non-en route cases. Compared to this, a second drone allows for significantly increased savings that typically fall in the range of 5 to 15%. In both cases, the increased savings depend strongly on the relative velocity and endurance of drones. This is also illustrated in Figure 2.5.4.

2.5.1.2 Results obtained through the Heuristic

In this section, we provide the numerical results that were obtained by solving the small instances through the heuristic introduced in Section 2.4. We begin by explaining the parameters that were used for the implementation of the heuristic. For the initialization procedure described in Section 2.4.1, after the initial generation of routes, $|V|^3$ iterations were spent on local search. We ran the heuristic five times, each time using a different random seed, for at most 10 minutes. If the incumbent solution remained unchanged for one minute, the heuristic terminated ahead of time. The value $k_{\max}$, that determines the maximum depth of the neighborhood within our VNS procedure, was set to $k_{\max} = 6$.

For the determination of feasible moves in the drone insertion heuristic (Algorithm 2.3), we only considered the limit on the number of drones currently in operation and the endurance with respect to the current operation. Recall that, as mentioned in Section 2.4.4, the insertion of a feasible operation can render previously feasibly inserted operations infeasible. The feasibility of a solution (defined by multiple operations) was then determined whenever a new branch was added to the rooted tree (Section 2.4.3). For the depth-first search of this tree, we impose a maximum degree of 30 on all nodes (including the root), before considering it to be fathomed. Alternatively, in each iteration of Algorithm 2.1, we canceled the exploration of the tree after 5000 calls of Algorithm 2.3.

The detailed results can be found in Tables 2.A.7–2.A.9 (see the Appendix). With respect to the results achieved by the MILP solver, overall, our heuristic demonstrated desirable behavior. In what follows, we will compare the results of our heuristic to the

best results achieved by the MILP solver through either approach (best results from Tables 2.A.1–2.A.3 and Tables 2.A.4–2.A.6). For the 4320 different problem instances and 5 runs per heuristic, the heuristic found a solution that was as good as or better than the one provided by the solver in 18569 out of 21600, i. e., 85.96% of all cases. Given that not every instance was solved to optimality by Gurobi after runtime, the results returned by the heuristic were on average 3.6% better in terms of the objective value. Therefore, it is fair to conclude, as backed up by the detailed results, that for the cases where the heuristic performed worse than the solver, the solution quality did not differ much. The average runtime for the heuristic was 106.56 seconds, i. e., on average the final solution was detected within $106.56 - 60 = 44.76$ seconds. Note that our neighborhood change (see Algorithm 2.2) is also triggered when the same objective value is found through a different routing (for an explanation, see Section 2.4). Therefore, the same solution (that is constructed through a different routing) might often be found earlier. As was the case for the MILP, runtime behavior was significantly influenced by the discrete steps, the number of drones, and endurance of drones. Clearly, all of these parameters influence the size of the tree explored during D&C.

Figure 2.5.4 visualizes the relative change in the makespan Δ (defined by formula (2.42)) for the cases that were recorded for the heuristic. These figures provide an insight in the savings through allowing an en route operation and the influence of the relative endurance $\mathcal{E}_r$ and relative velocity α. A look at this figure confirms the observations that were already mentioned in the previous section. More precisely, the introduction of a discrete step increases the savings by typically 2 to 3%. The introduction of a second drone is often much more effective, albeit, the relative endurance and velocity must be sufficiently large. Overall, the relative improvement diminishes as α or $\mathcal{E}_r$ are increased. Nevertheless, fast drones benefit noticeably more than slow drones from a further increase in endurance. Conversely, large endurance is only of limited usefulness in presence of slow-moving drones. These observations also apply to the cases where multiple trucks are present. For these small instances, in the presence of two trucks, the average number of customers served by each truck is very small. Therefore, drones often seem very effective in these cases. Care must be taken when comparing the cases of one truck to those where two trucks are present as the denominator when calculating Δ also changes.

2.5.2 Large Instances

In this section, we show that our heuristic is also suitable for tackling larger VRPDERO instances. To this end, we tested on reasonably-sized instances with either 20 or 50 vertices that are within the scope of those that were used for related problems (see, e. g., (Agatz

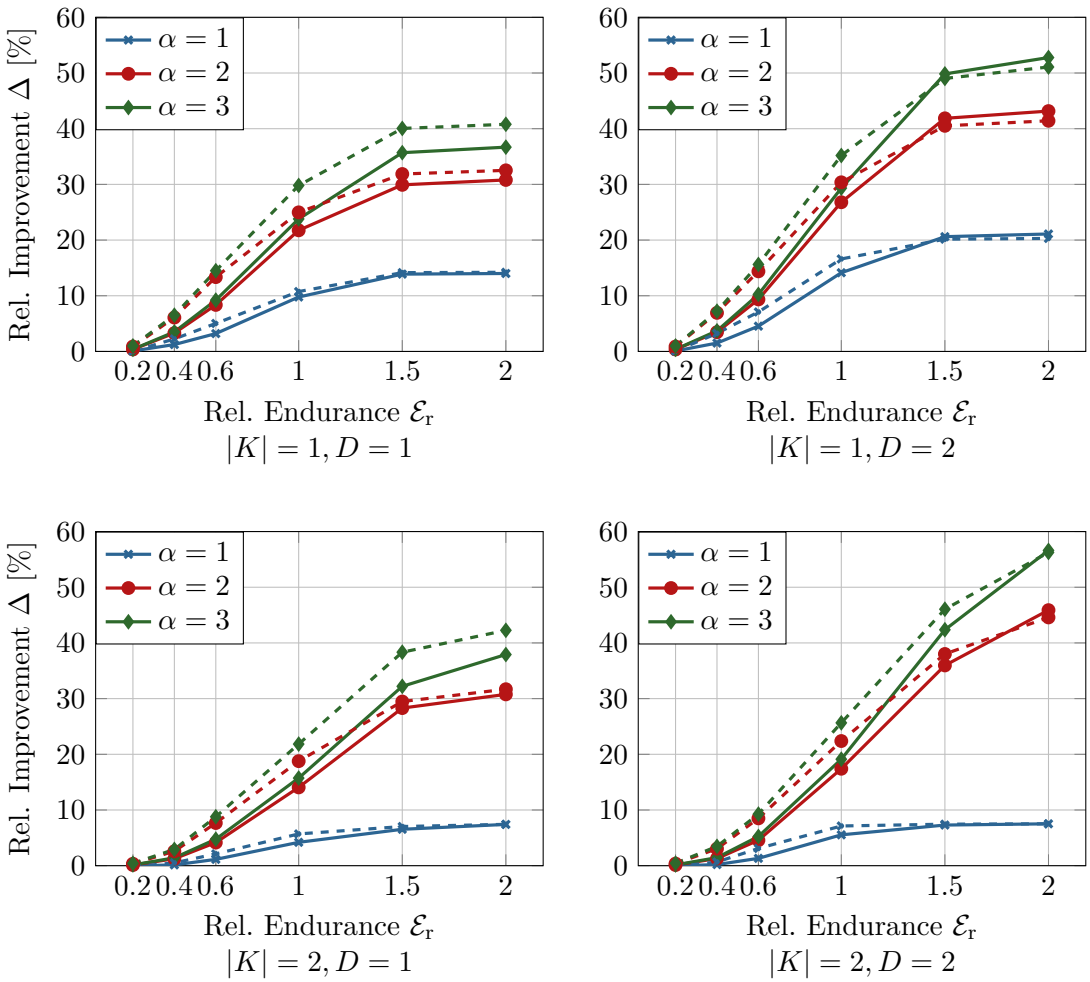

Figure 2.5.4: The average relative makespan reduction Δ depending on the number of trucks $|K|$, the number of drones per truck D, the relative endurance $\mathcal{E}_r$, and the relative velocity α (based on Tables 2.A.7–2.A.9). Lines shown for $s_n = 1$ (solid) and $s_n = 2$ (dashed).

et al., 2018; Bouman et al., 2018a; Murray and Chu, 2015)). On these instances, we tested according to the specifications given at the beginning of Section 2.5. Deviating from that, we only used $\mathcal{E}_r \in \{0.2, 0.4, 0.6, 1.0\}$ as any larger value on the relative endurance had no significant impact (see also Figure 2.5.4). Hence, in total, we tested on 1920 medium-sized instances. On each of these instances, we ran our heuristic five times with the time restrictions that were introduced previously.

With respect to the description of the parameters that were used for small instances (see Section 2.5.1.2), we adjusted some of them as follows. In order, to perform a more focused exploration of the search space, the maximum depth of the neighborhood has been reduced to $k_{\max} = 3$. Furthermore, to achieve a better tradeoff between diversification

(route generation) and intensification (exploration of the rooted tree), some parameters for the D&C approach were adjusted. More precisely, a maximum degree of 10 on the root node and 5 on all other nodes was imposed, before we considered them to be fathomed. Alternatively, we canceled the search after 10^4 calls of Algorithm 2.3.

The detailed results are available in Table 2.A.10 of the appendix. In general, our heuristic still performs well on these instances. On average, the final solution is found within just below two minutes of runtime (after 111 seconds). As there are no other results to compare ourselves to, for this section, we focus on the performance differences of our heuristic in solving these instances with regard to en route operations. We have seen in Section 2.5.1 that en route operations significantly increase the computational complexity of the problem. Therefore, we are interested to know if our heuristic is able to process larger instances as well.

Figure 2.5.5 compares the average objective values achieved through the heuristic while en route operations are allowed ($s_n = 2$) to the cases where they are not allowed ($s_n = 1$). More precisely, for each combination of trucks and drones, this figure contains a plot for $|V| \in \{20, 50\}$ and the different combinations of drone parameters.

Overall, the much more difficult case of en route operations is still handled well by our heuristic. As we already observed in the previous section, enabling en route operations can in some cases allow saving in the order of typically 2 to 3% and up to 10%. Typically, such savings are achieved if the relative velocity is large and the relative endurance small. However, it is also evident that the solutions found for the en route case are in some cases of slightly worse quality than those found for the basic case. As was already evident on small instances, the consideration of en route operations increases the size of the search space tremendously. As a result, much more effort has to be spent during the D&C approach to identify the best drone operations in the en route case, as the rooted tree might become significantly larger. For these cases, an implementation that provides a parallel depth-first search or prunes parts of the search tree might yield a significant speed-up.

In a similar manner, Figure 2.5.6 gives some insights into the structural differences when drone operations are allowed. More precisely, this figure visualizes the increase (decrease) of drone operations. This figure highlights that solutions differ noticeably at a small relative endurance values (i. e., $\mathcal{E}_r \in \{0.2, 0.4\}$), where the number of operations can be larger in the en route case. In particular, there is no large variation in these plots with a change in the relative velocity α. However, as we have seen in Figure 2.5.5, an increase in α can nevertheless reduce the makespan significantly.

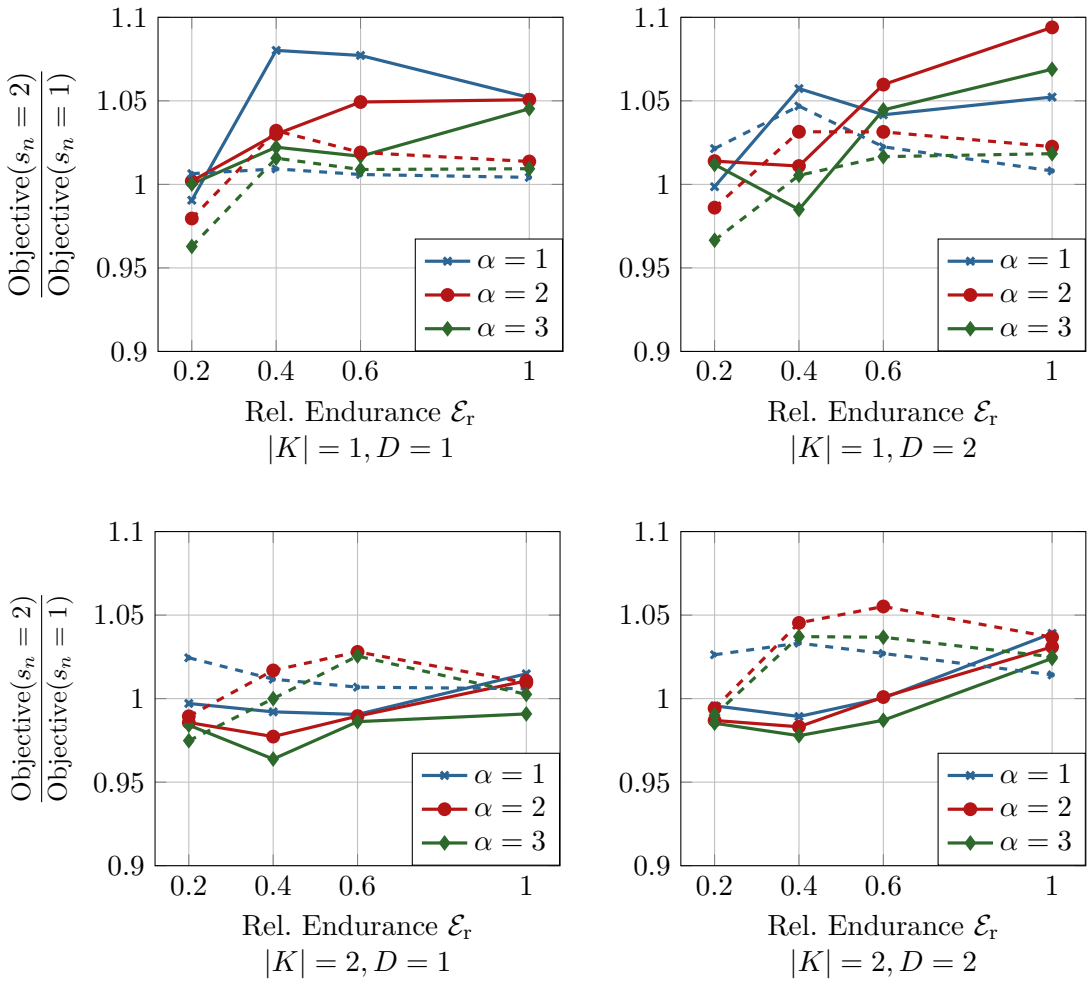

Figure 2.5.5: The average makespan shown as relative values for the en route and basic case depending on the number of trucks $|K|$, the drones per truck D, the relative endurance $\mathcal{E}_r$, and the relative velocity α (based on Table 2.A.10). Lines shown for $|V| = 20$ (solid) and $|V| = 50$ (dashed).

2.6 Conclusion

In this paper, we have introduced the VRPDERO, which consists of a combination of multiple trucks and drones in a last-mile delivery setting. The distinguishing feature between the VRPDERO and VRPD is that the former allows drones to be launched and retrieved on arcs by the truck; more precisely, drones are allowed to be launched and retrieved on certain discrete steps on each arc. We formulated the VRPDERO as a MILP. Then, we proposed VIEQs in order to enhance the performance of the solver and showed the effectiveness of the VIEQs through computational experiments. However, we are only

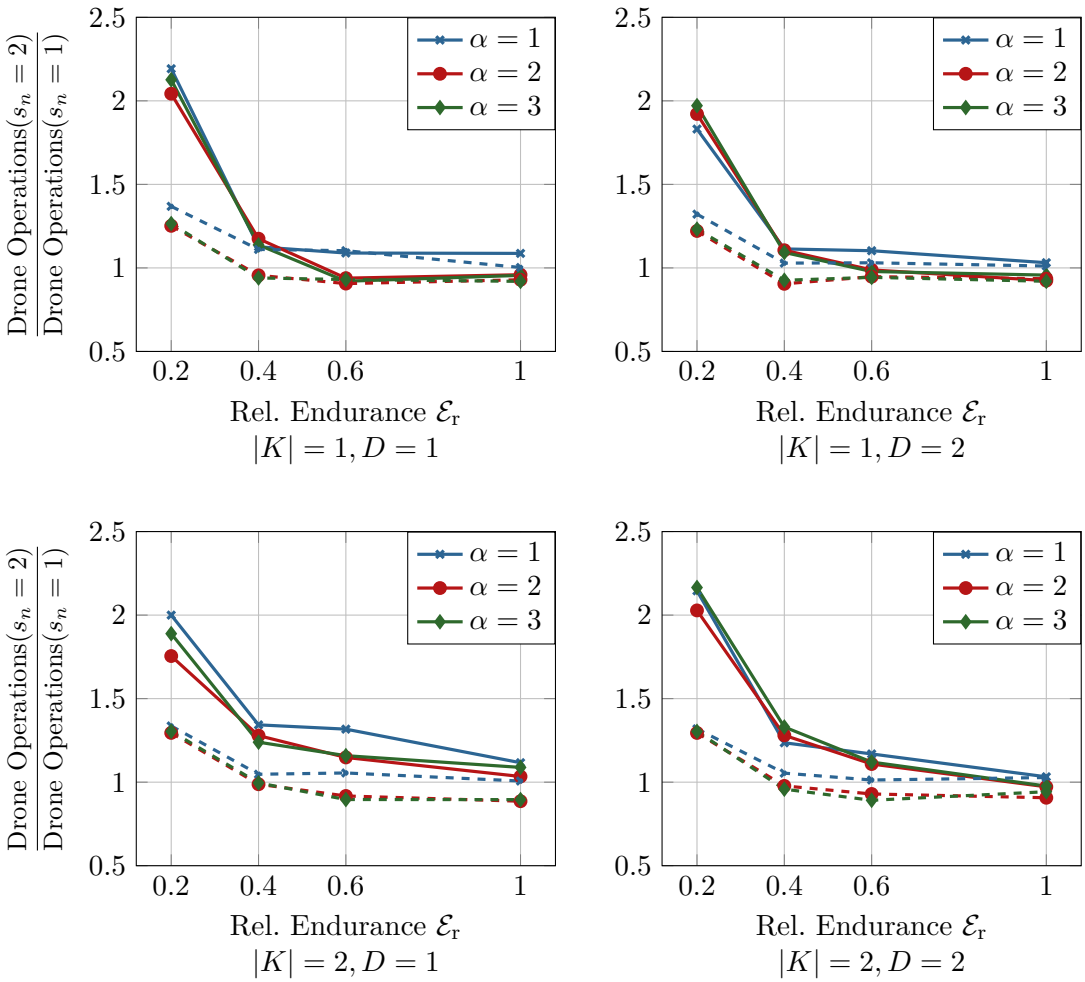

Figure 2.5.6: The average number of drone operations shown as relative values for the en route and basic case depending on the number of trucks $|K|$, the drones per truck D, the relative endurance $\mathcal{E}_r$, and the relative velocity α (based on Table 2.A.10). Lines shown for $|V| = 20$ (solid) and $|V| = 50$ (dashed).

able to solve some small-scale problems to optimality through a state-of-the-art solver within reasonable runtime. Therefore, we introduced a heuristic, which is based on the concepts of VNS and TS. Within this heuristic, as a local search operator, we introduced a drone insertion procedure that was embedded within a scalable D&C approach. We provided an extensive computational study that considered the impact of several problem parameters on, e. g., the makespan and runtime.

Overall, through our numerical results, we are able to show the potential of a reduced makespan and a higher utilization of drones if we consider the possibility of launching and retrieving drones en route. Notably, en route operations seem most beneficial when either

the drone endurance is small or the relative velocity is sufficiently large. In the former case, the drone is able to perform operations that would otherwise not be feasible. In the latter case, the drone is able to perform multiple operations in a most efficient way. However, we have also seen that the introduction of en route operations increases the computational complexity of the problem significantly. As a next step, it might be worth to refine our drone insertion operators. Naturally, it might be worthwhile to investigate the performance of a parallel depth-first search procedure. Moreover, a procedure to prune parts of the search tree might significantly increase the performance, in particular in solving larger instances. Finding alternative and possibly compact MILP formulations might also be an interesting research direction.

In this paper, we only considered a basic model with possibility of en route operations and assumed that the battery is recharged instantaneously after each delivery and discharged only if the drone is moving. Furthermore, possible effects of service times were neglected. In a practical setting, we might also be interested to investigate more realistic battery life considerations, recharge models, and the impact of potentially stochastic service times. For the moment, we leave them as future research plans.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments and suggestions that helped us in improving the quality of this paper.

2.7 References

Agatz, Niels, Paul Bouman, and Marie Schmidt (2018). "Optimization Approaches for the Traveling Salesman Problem with Drone". In: *Transportation Science* 52.4, pp. 965–981.

Bouman, Paul, Niels Agatz, and Marie Schmidt (2018a). "Dynamic programming approaches for the traveling salesman problem with drone". In: *Networks* 72.4, pp. 528–542.

Breedam, Alex van (1994). "An Analysis of the Behavior of Heuristics for the Vehicle Routing Problem for a selection of problems with Vehicle-related, Customer-related, and Time-related Constraints Contents". Ph.D. thesis. University of Antwerp.

Clarke, Geoff and John W. Wright (1964). "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points". In: *Operations Research* 12.4, pp. 568–581.

Cormen, Thomas H. et al. (2014). *Introduction to Algorithms*. 3rd ed. Cambridge: MIT Press.

Di Puglia Pugliese, Luigi and Francesca Guerriero (2017). "Last-Mile Deliveries by Using Drones and Classical Vehicles". In: *Optimization and Decision Science: Methodologies and Applications*. Springer Proceedings in Mathematics & Statistics 217. Cham: Springer, pp. 557–565.

Drexl, Michael (2012b). "Synchronization in Vehicle Routing—A Survey of VRPs with Multiple Synchronization Constraints". In: *Transportation Science* 46.3, pp. 297–316.

Gendreau, Michel and Jean-Yves Potvin (2010). *Handbook of Metaheuristics*. 2nd ed. Cham: Springer.

Gentry, Nicholas Kristofer, Raphael Hsieh, and Luan Khai Nguyen (2016). "Multi-use UAV docking station systems and methods". US9387928B1 (United States Patent).

Glover, Fred (1986). "Future paths for integer programming and links to artificial intelligence". In: *Computers & Operations Research* 13.5, pp. 533–549.

Gurobi Optimization, LLC (2023). *Gurobi Optimizer Reference Manual*.

Hansen, Pierre et al. (2017). "Variable neighborhood search: basics and variants". In: *EURO Journal on Computational Optimization* 5.3, pp. 423–454.

Kumar, Vipin and V. Nageshwara Rao (1987). "Parallel depth first search. Part II. Analysis". In: *International Journal of Parallel Programming* 16.6, pp. 501–519.

Lin, S. and B. W. Kernighan (1973). "An Effective Heuristic Algorithm for the Traveling-Salesman Problem". In: *Operations Research* 21.2, pp. 498–516.

Marinelli, Mario et al. (2018). "En route truck–drone parcel delivery for optimal vehicle routing strategies". In: *IET Intelligent Transport Systems* 12.4, pp. 253–261.

Miller, C. E., A. W. Tucker, and R. A. Zemlin (1960). "Integer Programming Formulation of Traveling Salesman Problems". In: *Journal of the ACM* 7.4, pp. 326–329.

Mladenović, N and P Hansen (1997). "Variable Neighborhood Search". In: *Computers & Operations Research* 24.11, pp. 1097–1100.

Murray, Chase C. and Amanda G. Chu (2015). "The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery". In: *Transportation Research Part C: Emerging Technologies* 54, pp. 86–109.

Otto, Alena et al. (2018). "Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey". In: *Networks* 72.4, pp. 411–458.

Poikonen, Stefan, Xingyin Wang, and Bruce Golden (2017). "The vehicle routing problem with drones: Extended models and connections". In: *Networks* 70.1, pp. 34–43.

Rao, Nageshwara N. and Vipin Kumar (1987). "Parallel depth first search. Part I. Implementation". In: *International Journal of Parallel Programming* 16.6, pp. 479–499.

Schermer, Daniel, Mahdi Moeini, and Oliver Wendt (2018a). "A Variable Neighborhood Search Algorithm for Solving the Vehicle Routing Problem with Drones". Technical

Report. Business Information Systems & Operations Research, Technische Universität Kaiserslautern.

Toth, Paolo and Daniele Vigo (2002). *The vehicle routing problem*. 1st ed. SIAM Series on Discrete Mathematics and Applications. Philadelphia: SIAM.

Wang, Xingyin, Stefan Poikonen, and Bruce Golden (2017). "The vehicle routing problem with drones: several worst-case results". In: *Optimization Letters* 11.4, pp. 679–697.

Appendix

2.A Detailed Numerical Results

This appendix contains the detailed numerical results. All experiments were performed on the uniformly distributed instances used by Agatz et al. (2018).

For each respective instance size and problem parameters, the values provided in the tables are shown as average values over 10 instances. Tables 2.A.1–2.A.6 show the results that were obtained through Gurobi Optimizer. More precisely, Tables 2.A.1–2.A.3 show the results that were obtained by solving MILP (2.2)–(2.27). For $s_n = 1$, VIEQs (2.29) and (2.35) were added to the model. For $s_n = 2$, we also added the variables y^k_{iwj}, enabling us to include constraints (2.30)–(2.31) to the model. In this case, the VIEQs (2.35) were replaced with the more general VIEQs (2.37). Tables 2.A.4–2.A.6 show the results that were obtained by treating constraints (2.16)–(2.17) as lazy constraints. Apart from the problem parameters such as the number of trucks, number of drones, discrete steps, relative velocity and relative endurance, the columns show the average savings Δ (see formula 2.42), the average runtime t and the average remaining MIP gap after runtime.

Furthermore, Tables 2.A.7–2.A.9 and 2.A.10 show the results that were obtained by solving the small and large instances through the proposed heuristic. These tables also show the average savings Δ, the average runtime t, and the average number of drone operations (Ops) that were present in the solution. As no optimal solutions for the non-drone case were generated for larger instances, Table 2.A.10 shows the objective value (Obj.) as real numbered values.

Table 2.A.1: MILP solver results on instances with 8 vertices.

			$\|K\| = 1$						$\|K\| = 2$					
			$D = 1$			$D = 2$			D=1			D=2		
s_n	α	$\mathcal{E}_r$	Δ	t	Gap	Δ	t	Gap	Δ	t	Gap	Δ	t	Gap
		0.2	0.1%	7.5	0.0%	0.1%	7.9	0.0%	0.0%	23.9	0.0%	0.0%	21.0	0.0%
		0.4	0.9%	13.8	0.0%	1.0%	13.1	0.0%	0.0%	51.7	0.0%	0.0%	66.0	0.0%
		0.6	3.0%	55.1	0.0%	3.5%	48.4	0.0%	0.7%	553.0	23.4%	0.8%	459.4	9.0%
	1	1	9.4%	348.6	6.0%	12.6%	423.5	14.8%	3.0%	603.9	46.3%	5.0%	601.0	49.8%
		1.5	13.1%	333.9	6.1%	19.9%	546.3	33.4%	6.3%	603.4	44.6%	6.6%	601.8	51.8%
		2	13.1%	385.9	14.2%	20.4%	556.5	35.8%	7.0%	603.5	43.3%	7.2%	602.0	61.0%
		0.2	0.5%	7.1	0.0%	0.5%	8.4	0.0%	0.4%	27.0	0.0%	0.4%	23.0	0.0%
		0.4	2.2%	14.4	0.0%	2.5%	11.2	0.0%	1.5%	45.1	0.0%	1.7%	58.3	0.0%
		0.6	5.3%	46.9	0.0%	6.0%	38.7	0.0%	3.9%	450.4	18.9%	4.7%	386.9	13.2%
1	2	1	20.2%	213.2	0.0%	24.2%	174.2	0.0%	12.0%	601.3	39.2%	13.4%	493.5	24.9%
		1.5	31.9%	357.4	9.7%	42.8%	188.4	5.8%	31.4%	474.2	32.5%	36.1%	340.5	8.2%
		2	33.2%	353.6	6.7%	45.6%	175.5	0.0%	37.2%	469.7	23.8%	47.5%	385.4	23.9%
		0.2	0.6%	7.3	0.0%	0.6%	6.9	0.0%	0.5%	21.2	0.0%	0.5%	24.2	0.0%
		0.4	2.5%	13.3	0.0%	2.9%	9.9	0.0%	1.8%	68.0	0.0%	1.9%	50.7	0.0%
		0.6	5.6%	44.5	0.0%	6.2%	41.5	0.0%	4.1%	427.2	18.1%	4.5%	400.9	6.8%
	3	1	22.5%	160.0	0.0%	26.3%	168.9	5.1%	16.0%	520.9	28.0%	17.6%	507.8	25.8%
		1.5	41.1%	191.9	0.0%	49.5%	104.6	0.0%	34.9%	396.8	28.2%	42.3%	358.0	12.3%
		2	41.8%	269.7	7.9%	53.8%	132.2	0.0%	43.4%	382.0	13.3%	63.2%	263.5	0.0%
Average			13.7%	156.9	2.8%	17.7%	147.6	5.3%	11.3%	351.3	20.0%	14.1%	313.6	15.9%
		0.2	0.2%	19.9	0.0%	0.3%	15.0	0.0%	0.0%	162.5	0.0%	0.0%	163.0	2.6%
		0.4	1.7%	151.5	0.0%	2.2%	91.8	0.0%	0.4%	546.8	28.7%	0.3%	521.6	25.4%
		0.6	4.3%	481.6	5.3%	5.6%	521.5	5.5%	1.5%	554.9	41.4%	1.2%	551.1	46.1%
	1	1	7.6%	604.0	24.2%	13.0%	604.1	36.5%	5.8%	601.5	47.1%	6.1%	602.6	52.8%
		1.5	12.6%	492.6	19.2%	19.8%	600.4	41.2%	6.8%	602.8	44.6%	7.2%	603.9	54.2%
		2	13.3%	501.5	13.1%	20.5%	600.2	30.2%	7.2%	603.0	41.0%	7.2%	603.6	61.7%
		0.2	0.7%	16.4	0.0%	0.7%	13.9	0.0%	0.5%	165.3	0.0%	0.5%	114.2	0.0%
		0.4	4.4%	79.8	0.0%	5.3%	83.0	0.0%	-0.5%	471.1	24.3%	2.1%	444.2	32.1%
		0.6	11.0%	295.6	2.6%	11.3%	278.1	0.9%	7.3%	488.7	26.7%	7.3%	509.6	32.8%
2	2	1	22.0%	490.0	17.3%	27.6%	321.9	16.6%	17.3%	530.1	27.7%	20.2%	480.8	18.6%
		1.5	33.3%	287.3	7.0%	44.8%	224.6	4.5%	34.0%	339.3	12.5%	37.3%	358.1	15.4%
		2	35.0%	254.0	1.0%	47.3%	262.2	1.5%	38.2%	399.4	5.3%	48.1%	433.3	8.9%
		0.2	0.8%	15.2	0.0%	0.8%	14.4	0.0%	0.5%	164.4	0.2%	0.5%	145.0	0.0%
		0.4	4.7%	94.0	0.0%	5.8%	76.0	0.0%	2.3%	426.5	17.5%	0.6%	470.5	25.5%
		0.6	10.1%	297.4	5.6%	11.1%	243.3	2.9%	8.0%	509.1	21.5%	7.0%	500.7	32.3%
	3	1	28.1%	367.4	15.4%	32.9%	280.1	10.2%	20.5%	531.9	22.4%	23.0%	436.1	22.6%
		1.5	43.2%	223.9	1.5%	52.7%	83.1	0.0%	41.7%	287.0	9.0%	45.3%	246.4	3.3%
		2	43.9%	217.1	1.0%	56.8%	104.1	0.0%	48.5%	282.1	4.1%	63.4%	235.7	0.0%
Average			15.4%	271.6	6.3%	19.9%	245.4	8.3%	13.3%	425.9	20.8%	15.4%	412.2	24.1%

Table 2.A.2: MILP solver results on instances with 9 vertices.

			$\|K\| = 1$						$\|K\| = 2$					
			$D = 1$			$D = 2$			D=1			D=2		
s_n	α	$\mathcal{E}_r$	Δ	t	Gap	Δ	t	Gap	Δ	t	Gap	Δ	t	Gap
		0.2	0.0%	13.4	0.0%	0.0%	13.0	0.0%	0.0%	72.6	0.0%	0.0%	76.3	0.0%
		0.4	1.5%	81.5	0.0%	1.8%	61.2	0.0%	0.2%	493.9	17.7%	0.3%	475.7	23.5%
		0.6	3.4%	421.3	8.0%	5.2%	321.3	6.2%	1.4%	603.0	36.3%	1.5%	602.5	32.8%
	1	1	8.9%	604.2	35.3%	13.1%	603.8	48.0%	1.1%	601.3	54.9%	0.8%	600.9	67.9%
		1.5	13.7%	601.9	30.8%	20.0%	602.4	46.6%	5.8%	602.8	52.0%	7.5%	604.1	67.3%
		2	14.3%	601.8	28.2%	20.8%	602.0	43.9%	6.1%	602.3	50.9%	8.1%	604.3	66.3%
		0.2	0.1%	13.3	0.0%	0.1%	13.3	0.0%	0.1%	77.9	0.0%	0.1%	71.5	0.0%
		0.4	4.5%	70.7	0.0%	4.8%	57.2	0.0%	1.5%	450.6	19.9%	1.3%	448.7	28.0%
		0.6	10.5%	241.6	0.0%	11.9%	230.5	0.0%	4.2%	601.9	32.1%	4.6%	601.2	38.3%
1	2	1	19.8%	603.4	43.7%	24.7%	601.5	50.2%	9.0%	600.8	66.4%	11.7%	601.0	76.9%
		1.5	30.0%	602.8	31.8%	41.3%	540.8	19.3%	24.2%	600.7	56.7%	35.0%	600.8	55.0%
		2	30.9%	607.1	29.8%	43.7%	563.0	14.3%	26.1%	600.7	55.3%	43.3%	600.7	49.6%
		0.2	0.1%	13.6	0.0%	0.1%	13.4	0.0%	0.1%	79.0	0.0%	0.1%	75.6	0.0%
		0.4	4.6%	68.1	0.0%	5.0%	51.2	0.0%	0.5%	447.6	27.1%	0.9%	433.8	21.7%
		0.6	11.7%	280.8	0.0%	13.3%	259.0	0.0%	4.8%	602.0	34.8%	6.0%	602.2	44.3%
	3	1	24.0%	601.1	50.1%	26.0%	601.5	47.1%	11.6%	600.8	66.5%	9.8%	600.9	80.5%
		1.5	34.7%	603.6	33.1%	52.3%	424.3	5.3%	30.9%	597.0	49.5%	41.6%	553.9	33.0%
		2	34.4%	583.6	33.8%	53.7%	406.5	4.3%	34.0%	600.7	48.2%	53.4%	539.7	23.4%
Average			13.7%	365.5	17.9%	18.8%	331.4	15.8%	9.0%	490.9	37.1%	12.6%	483.0	39.4%
		0.2	0.1%	151.1	0.0%	0.1%	114.4	0.0%	-2.2%	546.2	33.6%	-5.6%	516.2	30.5%
		0.4	-1.4%	527.7	15.8%	1.8%	508.8	14.6%	-62.7%	601.2	64.8%	-53.5%	600.9	64.6%
		0.6	1.1%	600.8	54.9%	4.8%	601.1	47.3%	-2.2%	601.3	87.5%	-13.8%	601.5	89.3%
	1	1	2.2%	600.9	68.0%	12.7%	600.9	68.7%	-2.9%	602.0	92.8%	-6.6%	601.8	94.0%
		1.5	11.3%	600.9	53.3%	18.9%	602.2	65.4%	1.8%	601.1	82.5%	6.4%	601.2	86.9%
		2	10.4%	601.5	48.4%	20.9%	602.3	56.7%	5.9%	600.8	85.2%	8.1%	600.9	74.8%
		0.2	0.5%	117.3	0.0%	0.5%	127.3	0.0%	-0.1%	556.6	32.3%	-0.3%	538.8	28.1%
		0.4	2.5%	554.0	14.7%	6.9%	465.0	4.3%	-33.1%	601.1	56.2%	-49.3%	601.1	59.3%
		0.6	10.5%	601.0	30.2%	11.0%	593.8	27.4%	0.2%	601.7	86.3%	-11.3%	601.4	83.1%
2	2	1	13.5%	601.3	59.5%	26.8%	577.5	38.6%	-2.7%	602.3	93.5%	2.9%	602.0	93.6%
		1.5	25.1%	601.1	47.6%	38.2%	602.9	35.7%	16.3%	601.2	82.4%	23.3%	601.0	76.8%
		2	22.3%	603.9	55.1%	41.2%	603.1	30.6%	24.6%	600.8	74.9%	45.6%	601.3	26.6%
		0.2	0.6%	135.0	0.0%	0.6%	131.3	0.0%	-41.5%	526.7	28.7%	-1.8%	558.4	36.1%
		0.4	2.9%	506.3	17.2%	-1.0%	436.9	17.8%	-60.1%	600.9	65.3%	-59.4%	601.0	74.0%
		0.6	14.6%	599.4	24.1%	16.2%	597.7	21.0%	-1.2%	601.4	83.0%	-11.2%	601.4	82.6%
	3	1	20.9%	601.9	46.7%	35.3%	560.5	25.5%	-0.7%	602.3	94.2%	6.5%	601.8	92.8%
		1.5	34.1%	601.9	31.8%	53.1%	497.6	9.9%	21.5%	601.2	75.9%	32.8%	580.9	60.7%
		2	31.6%	576.7	42.3%	53.9%	589.9	14.3%	35.6%	601.0	56.2%	59.0%	573.2	12.0%
Average			11.2%	508.2	33.7%	19.0%	489.6	26.5%	-5.8%	591.7	70.9%	-1.0%	587.9	64.7%

Table 2.A.3: MILP solver results on instances with 10 vertices.

			$\|K\| = 1$						$\|K\| = 2$					
			$D = 1$			$D = 2$			D=1			D=2		
s_n	α	$\mathcal{E}_r$	Δ	t	Gap	Δ	t	Gap	Δ	t	Gap	Δ	t	Gap
		0.2	0.1%	19.7	0.0%	0.1%	17.0	0.0%	0.1%	305.7	0.0%	0.1%	280.3	0.0%
		0.4	-0.8%	349.0	12.0%	-2.0%	323.2	15.6%	-0.9%	603.1	36.6%	-16.5%	602.0	54.3%
		0.6	-2.3%	601.9	32.8%	-3.8%	601.0	36.6%	-9.7%	601.1	63.5%	-1.0%	600.9	62.3%
	1	1	5.7%	600.7	41.6%	7.5%	600.9	57.0%	1.1%	601.7	68.6%	-0.3%	602.4	77.0%
		1.5	9.3%	600.8	38.5%	15.1%	600.8	49.5%	3.5%	601.7	66.9%	3.5%	601.2	73.4%
		2	9.5%	601.0	37.2%	16.3%	600.9	47.2%	4.8%	601.8	64.4%	5.9%	601.7	73.1%
		0.2	0.5%	19.1	0.0%	0.5%	20.3	0.0%	0.1%	327.4	1.2%	0.1%	278.1	0.0%
		0.4	-1.2%	319.5	14.9%	0.3%	299.8	15.4%	-4.9%	602.1	51.8%	-25.3%	591.2	64.5%
		0.6	-4.1%	598.9	36.8%	-4.7%	567.7	38.2%	-7.7%	601.0	72.5%	-3.1%	600.9	75.9%
1	2	1	15.9%	601.2	51.9%	21.6%	601.2	63.7%	6.8%	602.9	75.0%	9.1%	601.8	83.5%
		1.5	22.4%	601.2	43.9%	34.8%	600.9	51.1%	13.3%	601.5	71.4%	22.9%	601.4	78.2%
		2	23.1%	600.8	41.7%	35.4%	601.3	50.1%	19.0%	601.6	67.5%	28.1%	601.7	72.9%
		0.2	0.5%	20.6	0.0%	0.5%	20.7	0.0%	0.1%	276.4	0.0%	0.1%	270.2	0.0%
		0.4	-0.9%	340.2	16.1%	0.2%	307.1	15.9%	-15.7%	601.2	53.6%	-22.1%	601.1	53.2%
		0.6	-2.0%	593.2	41.7%	-0.5%	577.2	33.3%	-8.2%	601.0	78.1%	1.0%	599.5	72.3%
	3	1	16.3%	600.7	60.9%	21.1%	600.8	70.4%	4.4%	601.8	81.1%	8.0%	602.3	87.0%
		1.5	28.3%	600.9	47.9%	41.1%	601.1	56.3%	17.7%	601.5	75.5%	26.7%	601.5	81.8%
		2	27.6%	601.1	49.5%	49.1%	601.1	42.2%	19.4%	601.3	73.5%	36.4%	601.9	74.4%
Average			8.3%	461.9	31.7%	12.9%	452.4	35.7%	2.4%	551.9	55.6%	4.1%	546.7	60.2%
		0.2	-2.7%	474.5	18.5%	-11.4%	483.8	21.8%	-125.0%	602.0	85.1%	-86.3%	602.2	79.6%
		0.4	-36.2%	600.9	74.1%	-9.8%	601.1	58.9%	-54.6%	602.5	86.9%	-59.0%	602.3	87.3%
		0.6	-9.3%	601.1	87.7%	-9.9%	601.5	88.3%	-26.6%	602.9	96.2%	-43.0%	603.1	97.2%
	1	1	-9.2%	601.4	91.1%	-4.2%	601.6	92.1%	-33.2%	603.6	95.8%	-24.4%	603.0	96.5%
		1.5	5.1%	601.4	82.4%	11.6%	601.1	82.1%	-7.9%	601.8	91.3%	-1.5%	602.4	91.8%
		2	6.2%	600.9	79.6%	13.0%	601.1	79.3%	-11.6%	602.2	90.6%	1.1%	602.1	90.7%
		0.2	-5.1%	470.4	15.4%	-7.6%	487.1	18.0%	-95.6%	601.5	81.5%	-80.6%	601.3	79.5%
		0.4	-25.5%	601.1	63.3%	-20.7%	601.2	58.0%	-50.9%	602.3	87.3%	-63.5%	603.2	87.2%
		0.6	-5.4%	601.4	88.0%	-1.3%	601.1	85.1%	-38.8%	603.5	96.0%	-24.6%	604.1	96.7%
2	2	1	5.9%	601.4	90.4%	6.2%	601.2	91.1%	-41.5%	603.3	96.6%	-40.2%	604.9	97.3%
		1.5	18.1%	601.2	81.3%	27.6%	601.2	79.5%	-4.4%	601.6	91.9%	20.9%	602.2	91.2%
		2	17.0%	601.0	79.5%	30.9%	601.1	77.7%	1.2%	601.7	90.6%	25.8%	601.6	89.3%
		0.2	-11.2%	564.2	20.5%	-3.8%	480.0	20.1%	-92.4%	601.4	81.1%	-96.8%	601.6	81.7%
		0.4	-32.5%	601.0	64.7%	-32.9%	600.9	64.5%	-52.8%	602.0	90.8%	-93.8%	602.0	89.1%
		0.6	-6.6%	601.1	87.5%	0.2%	601.1	82.5%	-45.3%	602.6	96.7%	-22.1%	603.6	96.5%
	3	1	-1.9%	601.6	92.0%	7.4%	601.5	91.5%	-35.2%	602.8	96.5%	-56.0%	602.7	97.1%
		1.5	14.7%	601.1	83.1%	39.8%	601.7	72.0%	-5.5%	601.9	92.7%	17.5%	601.4	92.5%
		2	24.5%	601.0	78.2%	44.6%	601.3	67.7%	9.9%	602.1	90.5%	37.0%	601.6	88.6%
Average			-3.0%	585.4	71.3%	4.4%	581.6	68.4%	-39.5%	602.3	91.0%	-32.8%	602.5	90.5%

Table 2.A.4: MILP solver results (with lazy constraints) on instances with 8 vertices.

			$\|K\| = 1$						$\|K\| = 2$					
			$D = 1$			$D = 2$			D=1			D=2		
s_n	α	$\mathcal{E}_r$	Δ	t	Gap	Δ	t	Gap	Δ	t	Gap	Δ	t	Gap
		0.2	0.1%	4.4	0.0%	0.1%	4.5	0.0%	0.0%	24.3	0.0%	0.0%	29.3	0.0%
		0.4	0.9%	10.0	0.0%	1.0%	10.6	0.0%	0.0%	52.1	0.0%	0.0%	55.9	0.0%
		0.6	3.0%	24.9	0.0%	3.5%	26.8	0.0%	0.8%	300.3	0.0%	0.8%	256.0	0.0%
	1	1	9.4%	136.2	0.0%	12.7%	209.3	0.0%	5.4%	600.5	31.2%	4.7%	600.6	36.7%
		1.5	13.1%	139.2	0.0%	19.9%	257.5	0.0%	7.1%	517.7	6.0%	7.2%	520.1	9.4%
		2	13.1%	138.5	0.0%	20.5%	263.9	0.0%	7.2%	514.7	5.7%	7.2%	586.9	14.0%
		0.2	0.5%	4.1	0.0%	0.5%	4.6	0.0%	0.4%	23.1	0.0%	0.4%	23.6	0.0%
		0.4	2.2%	8.6	0.0%	2.5%	8.9	0.0%	1.5%	50.3	0.0%	1.7%	44.1	0.0%
1	2	0.6	5.3%	23.5	0.0%	6.0%	21.9	0.0%	3.9%	184.8	0.0%	4.7%	153.2	0.0%
		1	20.2%	104.9	0.0%	24.2%	116.7	0.0%	13.5%	592.1	35.2%	15.7%	501.9	17.4%
		1.5	31.9%	84.3	0.0%	42.8%	41.7	0.0%	33.0%	141.9	0.0%	36.1%	183.7	1.4%
		2	33.5%	79.4	0.0%	45.6%	77.0	0.0%	38.0%	148.5	0.0%	48.1%	161.4	0.0%
		0.2	0.6%	4.1	0.0%	0.6%	4.6	0.0%	0.5%	24.9	0.0%	0.5%	22.4	0.0%
		0.4	2.5%	9.3	0.0%	2.9%	10.0	0.0%	1.8%	43.9	0.0%	1.9%	40.4	0.0%
		0.6	5.6%	19.1	0.0%	6.2%	20.9	0.0%	4.1%	190.9	3.1%	5.1%	196.9	0.0%
	3	1	22.5%	79.7	0.0%	26.3%	127.7	0.0%	16.4%	516.0	25.5%	16.6%	510.3	24.2%
		1.5	41.1%	51.4	0.0%	49.5%	21.1	0.0%	38.0%	140.0	0.0%	42.3%	129.7	0.0%
		2	41.8%	43.5	0.0%	53.8%	21.8	0.0%	44.7%	95.0	0.0%	63.2%	69.5	0.0%
Average			13.7%	53.6	0.0%	17.7%	69.4	0.0%	12.0%	231.2	5.9%	14.2%	227.0	5.7%
		0.2	0.2%	10.8	0.0%	0.3%	11.3	0.0%	0.0%	83.1	0.0%	0.0%	82.5	0.0%
		0.4	1.7%	60.5	0.0%	2.2%	50.5	0.0%	-2.5%	467.2	20.0%	-6.0%	436.0	37.6%
		0.6	3.4%	443.6	8.9%	5.2%	408.5	9.2%	1.7%	553.1	30.8%	1.5%	547.5	31.8%
	1	1	4.7%	602.4	41.1%	10.7%	602.3	48.7%	4.3%	602.3	84.0%	3.1%	602.9	85.8%
		1.5	13.3%	599.2	10.8%	20.2%	598.6	30.9%	5.6%	600.9	83.4%	6.1%	600.8	79.8%
		2	13.4%	480.9	8.0%	20.4%	600.6	18.8%	5.4%	601.0	78.9%	7.1%	600.8	83.8%
		0.2	0.7%	10.1	0.0%	0.7%	11.2	0.0%	0.5%	64.0	0.0%	0.5%	58.0	0.0%
		0.4	4.4%	39.2	0.0%	5.3%	29.6	0.0%	-2.1%	443.7	22.5%	-4.2%	347.7	24.4%
2	2	0.6	10.4%	217.8	4.1%	11.3%	122.2	0.0%	7.1%	383.9	13.3%	-2.4%	455.6	31.3%
		1	17.5%	603.2	39.8%	26.1%	333.0	15.4%	10.4%	601.0	80.2%	14.6%	563.9	74.2%
		1.5	33.3%	474.4	6.6%	44.9%	377.0	0.0%	18.8%	601.3	83.9%	30.7%	601.7	75.4%
		2	35.0%	362.5	0.3%	47.3%	315.6	0.9%	31.8%	601.2	50.3%	45.2%	600.8	68.4%
		0.2	0.8%	11.6	0.0%	0.8%	11.3	0.0%	0.5%	54.1	0.0%	0.5%	49.9	0.0%
		0.4	4.7%	32.8	0.0%	5.8%	27.5	0.0%	1.6%	302.9	5.7%	-4.3%	381.7	26.7%
		0.6	11.2%	193.5	3.3%	11.4%	107.5	0.0%	6.4%	395.9	17.9%	8.4%	449.6	15.4%
	3	1	28.5%	300.7	13.0%	31.5%	284.4	11.8%	12.6%	554.8	64.2%	18.3%	517.8	59.8%
		1.5	43.2%	265.2	0.8%	52.7%	109.8	0.0%	31.1%	477.0	60.7%	39.7%	506.8	53.5%
		2	44.1%	280.3	0.0%	56.8%	233.0	0.0%	39.7%	516.9	57.4%	56.6%	600.4	57.1%
Average			15.0%	277.1	7.6%	19.6%	235.2	7.5%	9.6%	439.1	41.9%	12.0%	444.7	44.7%

Table 2.A.5: MILP solver results (with lazy constraints) on instances with 9 vertices.

			$\|K\| = 1$						$\|K\| = 2$					
			$D = 1$			$D = 2$			D=1			D=2		
s_n	α	$\mathcal{E}_r$	Δ	t	Gap	Δ	t	Gap	Δ	t	Gap	Δ	t	Gap
1	1	0.2	0.0%	10.1	0.0%	0.0%	9.8	0.0%	0.0%	116.8	0.0%	0.0%	118.3	0.0%
		0.4	1.5%	62.7	0.0%	1.8%	59.6	0.0%	0.3%	385.2	11.7%	0.4%	390.4	11.5%
		0.6	3.4%	308.2	3.6%	5.4%	254.2	4.1%	1.4%	600.5	35.4%	0.7%	600.9	34.2%
		1	8.9%	600.3	28.4%	11.7%	600.5	42.2%	0.0%	601.5	60.5%	-1.8%	602.2	70.9%
		1.5	15.6%	472.3	5.3%	21.7%	593.1	15.4%	6.9%	600.6	53.0%	5.3%	600.6	70.9%
		2	15.9%	435.2	1.2%	22.7%	598.0	12.1%	8.0%	600.4	35.1%	8.1%	600.6	44.3%
	2	0.2	0.1%	9.5	0.0%	0.1%	9.9	0.0%	0.1%	113.5	0.0%	0.1%	124.2	0.0%
		0.4	4.5%	44.6	0.0%	4.8%	44.5	0.0%	1.8%	347.4	9.4%	1.8%	350.4	5.9%
		0.6	10.5%	164.3	0.0%	11.9%	149.0	0.0%	4.1%	601.0	42.4%	3.7%	590.6	30.1%
		1	20.8%	548.2	21.7%	25.5%	554.8	22.0%	5.9%	602.2	68.5%	6.8%	603.6	76.9%
		1.5	31.0%	354.2	0.0%	42.9%	300.9	2.6%	28.3%	583.1	30.7%	35.2%	526.3	25.9%
		2	31.6%	369.2	0.0%	44.4%	306.5	0.0%	32.9%	526.2	15.9%	47.5%	515.7	13.2%
	3	0.2	0.1%	9.8	0.0%	0.1%	9.9	0.0%	0.1%	119.9	0.0%	0.1%	145.7	0.0%
		0.4	4.6%	39.9	0.0%	5.0%	47.2	0.0%	2.0%	376.5	4.6%	2.0%	379.8	6.7%
		0.6	11.7%	114.6	0.0%	13.3%	118.0	0.0%	5.8%	585.7	34.2%	5.8%	537.1	31.2%
		1	21.6%	476.3	27.0%	25.8%	512.2	35.2%	-5.2%	602.7	74.7%	8.6%	601.4	83.1%
		1.5	35.9%	293.8	0.0%	52.3%	70.9	0.0%	34.7%	517.8	36.3%	42.6%	427.5	16.0%
		2	36.2%	276.7	0.0%	53.7%	164.2	0.0%	43.0%	376.6	0.4%	55.0%	324.5	11.0%
Average			14.1%	255.0	4.8%	19.1%	244.6	7.4%	9.5%	458.8	28.5%	12.3%	446.6	29.5%
2	1	0.2	0.1%	30.0	0.0%	0.1%	38.1	0.0%	0.1%	403.2	7.6%	0.0%	405.2	11.4%
		0.4	1.8%	421.3	12.7%	2.8%	340.5	9.4%	-25.2%	600.6	53.1%	-31.3%	570.6	51.3%
		0.6	-3.9%	604.4	50.2%	0.8%	603.1	46.9%	-8.9%	601.0	75.9%	-26.9%	601.5	80.6%
		1	-0.3%	608.0	63.8%	6.9%	604.5	70.7%	-0.2%	602.7	90.3%	2.3%	624.0	92.1%
		1.5	5.5%	601.6	61.4%	13.3%	601.2	73.3%	-0.8%	602.0	89.1%	1.6%	601.6	89.9%
		2	5.5%	600.6	60.1%	15.2%	600.5	65.5%	1.8%	601.2	86.1%	3.3%	600.7	87.7%
	2	0.2	0.5%	31.7	0.0%	0.5%	31.4	0.0%	0.1%	380.1	8.4%	0.3%	356.3	8.4%
		0.4	6.3%	242.1	3.6%	7.1%	229.0	1.9%	-38.0%	534.7	53.0%	-21.0%	562.4	44.7%
		0.6	7.4%	578.1	37.4%	11.2%	567.6	27.1%	-13.9%	601.1	74.2%	-12.2%	601.3	80.5%
		1	9.8%	603.9	67.0%	23.3%	570.5	48.8%	6.9%	642.0	90.3%	11.0%	623.2	86.1%
		1.5	12.0%	601.0	73.6%	28.8%	600.9	60.2%	8.8%	601.1	89.2%	14.7%	601.9	89.5%
		2	16.7%	600.7	61.1%	36.0%	600.9	56.6%	8.4%	600.9	87.4%	20.5%	600.5	87.1%
	3	0.2	0.6%	35.6	0.0%	0.6%	37.2	0.0%	0.4%	398.1	7.5%	0.4%	310.7	2.9%
		0.4	6.2%	289.7	6.6%	8.0%	182.8	0.0%	-18.0%	579.1	44.3%	-22.2%	569.6	52.4%
		0.6	10.9%	507.9	26.3%	12.8%	485.2	23.4%	-2.7%	601.2	74.3%	-15.1%	601.7	75.3%
		1	9.3%	607.9	75.6%	29.4%	531.4	37.7%	6.6%	626.3	91.0%	11.0%	636.9	92.2%
		1.5	22.3%	602.4	64.5%	49.2%	600.7	40.8%	12.8%	601.1	89.4%	16.5%	601.6	91.0%
		2	32.0%	600.4	34.6%	53.6%	582.7	33.4%	17.5%	601.3	86.5%	31.1%	602.8	87.9%
Average			7.9%	453.7	38.8%	16.6%	433.8	33.1%	-2.5%	565.4	66.5%	-0.9%	559.6	67.3%

Table 2.A.6: MILP solver results (with lazy constraints) on instances with 10 vertices.

			$\|K\| = 1$						$\|K\| = 2$					
			$D = 1$			$D = 2$			D=1			D=2		
s_n	α	$\mathcal{E}_r$	Δ	t	Gap	Δ	t	Gap	Δ	t	Gap	Δ	t	Gap
1	1	0.2	0.1%	17.9	0.0%	0.1%	17.5	0.0%	0.1%	485.5	11.3%	0.1%	478.3	8.5%
		0.4	1.5%	251.6	5.7%	1.7%	246.0	5.6%	-0.8%	601.3	31.4%	-0.7%	601.5	38.0%
		0.6	1.1%	570.4	28.4%	2.6%	565.3	31.8%	-2.2%	601.2	58.4%	-2.0%	602.1	64.7%
		1	4.6%	601.0	39.9%	4.5%	601.0	55.1%	-2.0%	604.0	64.8%	0.0%	602.3	74.9%
		1.5	7.8%	600.6	33.5%	14.6%	600.5	48.3%	1.2%	601.7	65.7%	0.5%	601.4	74.4%
		2	10.7%	600.5	25.6%	15.5%	600.6	40.1%	1.1%	601.3	64.1%	4.8%	601.1	73.1%
1	2	0.2	0.5%	18.0	0.0%	0.5%	16.9	0.0%	0.1%	475.6	10.3%	0.1%	489.5	10.8%
		0.4	3.3%	152.5	0.0%	3.5%	216.4	0.0%	0.1%	601.4	35.9%	-0.8%	601.9	35.4%
		0.6	5.9%	488.2	29.4%	7.2%	476.0	21.5%	-0.3%	601.7	64.9%	0.9%	601.7	65.9%
		1	13.7%	601.0	46.7%	19.3%	601.3	47.8%	6.5%	602.1	72.3%	9.8%	601.2	79.5%
		1.5	22.0%	600.5	33.9%	31.3%	588.6	36.4%	14.1%	601.7	69.6%	21.8%	601.7	75.1%
		2	22.6%	590.4	30.5%	36.0%	556.7	39.4%	16.2%	600.7	67.6%	26.3%	602.0	71.9%
	3	0.2	0.5%	20.1	0.0%	0.5%	17.6	0.0%	0.1%	489.2	10.2%	0.1%	471.7	10.8%
		0.4	3.4%	177.3	0.0%	3.6%	221.5	2.0%	-0.6%	600.9	37.3%	-0.3%	593.2	36.0%
		0.6	8.3%	481.6	29.3%	6.7%	478.2	24.5%	-1.9%	603.2	70.7%	1.1%	572.1	60.2%
		1	6.3%	601.0	57.1%	21.0%	601.7	53.3%	5.7%	602.7	78.2%	-5.2%	601.6	85.5%
		1.5	29.3%	587.9	31.4%	46.8%	519.3	28.3%	15.1%	601.6	72.5%	26.8%	601.1	75.1%
		2	31.4%	584.3	26.5%	47.7%	560.8	29.7%	23.3%	601.5	69.9%	36.3%	601.7	73.3%
Average			9.6%	419.1	23.2%	14.6%	415.9	25.8%	4.2%	582.1	53.1%	6.6%	579.2	56.3%
2	1	0.2	0.2%	87.7	0.0%	0.2%	109.3	0.0%	-0.7%	594.5	31.9%	-2.7%	599.8	30.1%
		0.4	-2.0%	600.6	32.1%	-1.7%	585.5	31.3%	-68.4%	600.7	89.1%	-87.4%	600.8	84.2%
		0.6	-11.4%	601.1	72.4%	-7.5%	601.1	69.9%	-15.2%	615.0	94.3%	-10.8%	601.3	92.4%
		1	-9.4%	600.8	83.1%	-1.0%	602.1	84.2%	-9.9%	601.4	95.3%	-11.0%	601.9	95.7%
		1.5	-0.9%	601.3	78.9%	7.6%	601.4	82.5%	-3.7%	601.7	90.8%	-0.1%	601.7	91.8%
		2	1.2%	601.0	69.6%	10.5%	601.4	76.4%	-3.4%	601.2	90.1%	3.5%	601.5	90.2%
	2	0.2	1.5%	84.8	0.0%	1.5%	85.2	0.0%	-0.3%	582.8	24.0%	-8.3%	580.1	25.8%
		0.4	1.9%	509.7	27.3%	1.6%	505.1	28.1%	-56.6%	601.3	84.3%	-50.4%	601.1	80.3%
		0.6	-2.8%	600.9	68.2%	5.1%	601.9	57.9%	-16.0%	601.5	94.4%	-2.0%	601.1	93.8%
		1	0.4%	601.8	83.6%	11.2%	601.1	81.4%	-3.6%	601.6	95.7%	2.4%	601.6	95.0%
		1.5	6.9%	601.2	79.7%	17.9%	601.2	79.9%	5.1%	601.4	91.4%	17.3%	602.0	91.2%
		2	11.6%	601.6	75.4%	23.5%	600.8	78.6%	6.5%	601.1	90.2%	20.2%	601.2	89.5%
	3	0.2	1.6%	78.1	0.0%	1.6%	88.8	0.0%	-2.7%	601.5	28.5%	-4.8%	601.6	35.0%
		0.4	2.1%	544.9	26.9%	4.6%	465.7	18.4%	-63.4%	601.2	79.9%	-56.6%	601.5	78.2%
		0.6	-3.6%	602.1	67.2%	6.4%	601.3	60.3%	-11.0%	601.4	93.2%	-13.2%	624.9	93.4%
		1	6.3%	601.0	83.4%	18.2%	601.5	69.7%	2.1%	601.8	94.6%	1.0%	601.5	94.8%
		1.5	7.5%	601.0	83.5%	31.3%	603.3	72.5%	6.6%	602.9	92.0%	20.9%	602.1	91.7%
		2	11.3%	600.8	79.5%	29.4%	602.1	77.9%	15.9%	601.3	89.9%	33.5%	603.2	88.3%
Average			1.3%	506.2	56.0%	8.9%	503.3	53.8%	-11.5%	600.8	80.5%	-7.6%	601.6	80.0%

Table 2.A.7: Heuristic results on instances with 8 vertices.

			$\|K\| = 1$						$\|K\| = 2$					
			$D = 1$			$D = 2$			D=1			D=2		
s_n	α	$\mathcal{E}_r$	Δ	t	Ops	Δ	t	Ops	Δ	t	Ops	Δ	t	Ops
		0.2	0.1%	61.6	0.4	0.1%	61.6	0.4	-0.1%	61.4	0.2	-0.1%	61.4	0.2
		0.4	0.9%	61.8	0.9	1.0%	62.1	1.1	-0.1%	61.5	0.7	-0.1%	61.5	0.7
		0.6	2.9%	62.2	1.4	3.5%	67.6	2.2	0.7%	61.5	1.4	0.8%	62.3	1.7
	1	1	9.4%	63.6	1.8	12.7%	85.3	2.9	3.9%	61.8	1.9	5.5%	63.6	2.6
		1.5	13.1%	64.2	1.9	19.9%	202.3	3.2	6.1%	62.0	2.2	7.1%	63.9	3.4
		2	13.1%	64.5	1.9	20.5%	267.3	3.0	7.2%	62.7	2.0	7.2%	77.3	3.5
		0.2	0.5%	61.7	0.4	0.5%	61.7	0.4	0.3%	61.4	0.2	0.3%	61.4	0.2
		0.4	2.2%	61.8	0.9	2.2%	62.1	1.1	1.5%	61.5	0.7	1.5%	61.6	0.7
		0.6	5.3%	62.3	1.4	5.9%	75.3	2.2	3.5%	61.5	1.6	3.9%	62.0	1.8
2	2	1	20.0%	63.6	2.3	24.2%	103.9	3.5	11.6%	61.9	2.5	14.4%	64.2	3.1
		1.5	31.7%	64.6	2.8	42.8%	238.7	4.2	30.6%	62.3	3.2	36.0%	68.3	4.2
		2	33.2%	65.0	2.9	45.6%	290.0	4.1	35.9%	62.2	3.1	47.4%	67.7	4.4
		0.2	0.6%	61.7	0.4	0.6%	61.7	0.4	0.4%	61.5	0.2	0.4%	61.4	0.2
		0.4	2.5%	61.8	0.9	2.5%	62.1	1.1	1.8%	61.5	0.7	1.8%	61.6	0.7
		0.6	5.5%	62.2	1.3	6.2%	75.9	2.3	3.7%	61.5	1.6	4.1%	62.0	1.8
	3	1	22.1%	63.6	2.4	26.3%	107.6	3.6	14.1%	62.0	2.5	16.0%	64.1	2.9
		1.5	39.7%	64.6	3.1	49.3%	259.4	4.3	35.0%	62.3	3.3	41.7%	69.0	4.2
		2	41.0%	65.0	3.2	53.8%	215.1	4.3	42.6%	62.2	3.3	56.1%	67.2	4.5
Average			13.5%	63.1	1.7	17.7%	131.1	2.5	11.0%	61.8	1.7	13.6%	64.5	2.3
		0.2	0.1%	61.9	0.6	0.1%	62.2	0.7	-0.2%	61.5	0.6	-0.2%	61.6	0.6
		0.4	1.7%	62.9	1.2	2.5%	76.2	2.1	0.3%	61.9	1.4	0.4%	62.7	1.7
		0.6	4.4%	65.9	1.7	5.7%	124.8	2.8	1.9%	62.6	1.9	2.5%	65.5	2.5
	1	1	10.0%	77.3	2.2	16.7%	182.5	3.3	5.8%	63.0	2.2	6.9%	66.9	3.1
		1.5	13.4%	88.5	2.0	20.1%	179.8	3.5	6.7%	63.0	2.1	7.1%	67.5	3.4
		2	13.4%	87.4	2.0	20.2%	186.8	3.4	7.2%	64.5	2.0	7.2%	146.8	3.8
		0.2	0.7%	61.8	0.7	0.7%	62.1	0.9	0.5%	61.5	0.6	0.5%	61.6	0.6
		0.4	4.3%	63.1	1.4	5.3%	82.1	2.3	2.3%	62.0	1.5	3.1%	62.9	1.7
		0.6	10.7%	66.4	2.3	11.3%	157.7	3.5	7.5%	63.1	2.7	7.8%	67.4	2.9
2	2	1	23.4%	87.9	3.3	27.9%	189.0	4.2	16.7%	64.0	3.4	20.2%	97.8	3.8
		1.5	33.6%	106.2	3.2	43.2%	127.0	4.3	32.0%	64.0	3.4	37.7%	80.8	4.0
		2	35.0%	117.5	3.2	45.3%	133.9	4.2	35.7%	64.1	3.3	44.2%	91.7	4.3
		0.2	0.8%	61.8	0.7	0.8%	62.1	0.9	0.5%	61.5	0.6	0.5%	61.6	0.6
		0.4	4.7%	63.3	1.4	5.7%	85.0	2.3	2.4%	62.0	1.5	3.3%	62.7	1.8
		0.6	11.0%	67.7	2.4	11.4%	166.4	3.4	8.8%	63.1	2.6	8.7%	67.5	3.0
	3	1	28.3%	92.9	3.4	32.7%	218.5	4.3	19.3%	64.1	3.3	22.7%	103.4	3.9
		1.5	43.1%	110.5	4.0	50.7%	159.5	4.5	39.0%	64.2	3.6	45.1%	82.1	4.2
		2	44.0%	115.6	3.9	53.9%	151.0	4.5	43.3%	64.4	4.0	53.8%	84.8	4.4
Average			15.7%	81.0	2.2	19.7%	133.7	3.1	12.8%	63.0	2.3	15.1%	77.5	2.8

Table 2.A.8: Heuristic results on instances with 9 vertices.

			$\|K\| = 1$						$\|K\| = 2$					
			$D = 1$			$D = 2$			D=1			D=2		
s_n	α	$\mathcal{E}_r$	Δ	t	Ops	Δ	t	Ops	Δ	t	Ops	Δ	t	Ops
		0.2	0.0%	61.7	0.1	0.0%	61.7	0.1	0.0%	61.6	0.1	0.0%	61.6	0.1
		0.4	1.4%	62.2	1.1	1.8%	62.7	1.3	0.2%	61.6	0.6	0.4%	61.7	0.7
		0.6	3.4%	63.6	1.5	5.4%	65.8	2.2	1.6%	61.8	1.4	1.8%	62.7	1.7
	1	1	10.4%	64.8	2.0	15.3%	196.0	3.5	4.2%	62.5	2.3	6.0%	64.9	3.1
		1.5	15.8%	68.2	2.0	22.5%	317.4	3.7	7.5%	62.8	2.4	8.0%	71.1	3.7
		2	16.1%	69.0	2.0	23.1%	303.2	3.7	7.9%	63.2	2.2	8.1%	123.6	3.8
		0.2	0.1%	61.7	0.2	0.1%	61.7	0.2	0.1%	61.6	0.1	0.1%	61.6	0.1
		0.4	4.4%	62.0	1.1	4.7%	62.7	1.4	1.7%	61.6	0.7	1.7%	61.7	0.7
		0.6	10.5%	63.1	1.4	11.7%	64.9	2.3	5.4%	61.8	1.5	5.8%	62.6	1.8
1	2	1	23.1%	64.6	2.8	27.7%	166.6	4.0	14.0%	62.6	2.8	17.4%	69.3	3.6
		1.5	30.7%	67.2	3.3	43.1%	345.5	4.8	28.0%	63.2	3.5	37.1%	71.6	4.7
		2	31.4%	68.8	3.5	43.3%	232.0	4.9	28.6%	63.1	3.6	45.7%	73.7	5.1
		0.2	0.1%	61.7	0.2	0.1%	61.7	0.2	0.1%	61.6	0.1	0.1%	61.6	0.1
		0.4	4.6%	62.1	1.1	4.9%	62.7	1.4	1.8%	61.6	0.7	1.8%	61.7	0.7
		0.6	11.7%	63.1	1.6	13.2%	65.2	2.5	6.0%	61.8	1.6	6.6%	62.7	1.7
	3	1	25.4%	64.8	2.9	30.0%	183.9	4.2	15.1%	62.6	2.9	19.1%	70.5	3.5
		1.5	34.6%	69.9	3.5	51.2%	335.3	4.9	31.9%	63.1	3.7	42.9%	71.6	4.7
		2	35.6%	70.4	3.7	53.3%	273.8	5.2	36.3%	63.3	3.9	53.9%	72.2	5.0
Average			14.4%	64.9	1.9	19.5%	162.4	2.8	10.6%	62.3	1.9	14.3%	69.2	2.5
		0.2	0.0%	61.9	0.4	0.0%	62.0	0.5	-0.2%	61.7	0.3	-0.2%	61.7	0.3
		0.4	2.6%	64.6	1.4	3.9%	73.7	2.0	0.5%	62.2	1.4	0.6%	63.3	1.7
		0.6	6.0%	76.0	2.0	8.9%	320.9	3.2	2.7%	64.0	2.1	4.1%	68.3	2.9
	1	1	11.8%	151.8	2.2	17.7%	220.3	3.9	5.8%	64.5	2.5	7.7%	79.7	3.6
		1.5	16.1%	246.5	2.3	21.8%	185.9	3.9	7.5%	64.8	2.3	8.1%	78.3	3.9
		2	16.3%	221.8	2.3	22.1%	205.9	3.9	7.9%	65.9	2.3	8.1%	240.1	4.0
		0.2	0.5%	61.9	0.5	0.5%	62.0	0.6	0.4%	61.7	0.3	0.4%	61.7	0.3
		0.4	6.6%	64.2	1.6	7.8%	75.4	2.4	3.5%	62.1	1.7	3.7%	63.4	1.9
		0.6	15.2%	73.0	2.6	17.1%	216.3	3.8	8.3%	64.3	2.8	9.1%	69.2	3.5
2	2	1	27.3%	141.3	3.4	32.7%	296.4	4.7	19.8%	66.1	3.9	23.7%	117.7	4.5
		1.5	33.0%	209.4	4.1	40.7%	158.5	4.7	28.4%	66.5	4.1	38.4%	107.1	4.8
		2	33.5%	203.0	4.1	40.9%	158.1	4.9	29.4%	67.4	3.7	45.0%	91.0	5.0
		0.2	0.6%	61.9	0.5	0.6%	62.0	0.6	0.5%	61.7	0.3	0.5%	61.7	0.3
		0.4	7.1%	64.3	1.5	8.1%	73.6	2.3	4.3%	62.3	1.7	4.5%	63.6	2.0
		0.6	16.6%	72.7	2.7	19.0%	242.7	3.7	9.2%	64.6	2.9	9.9%	77.3	3.1
	3	1	31.9%	164.6	3.8	37.6%	294.8	4.9	23.9%	67.2	4.0	28.1%	168.7	4.7
		1.5	39.9%	244.5	4.8	49.2%	190.1	4.9	37.1%	66.8	4.5	44.8%	96.9	4.9
		2	40.6%	228.6	4.7	50.9%	191.0	5.0	42.0%	66.6	4.6	55.0%	86.0	5.0
Average			17.0%	134.0	2.5	21.1%	171.6	3.3	12.8%	64.5	2.5	16.2%	92.0	3.1

Table 2.A.9: Heuristic results on instances with 10 vertices.

			$\|K\| = 1$						$\|K\| = 2$					
			$D = 1$			$D = 2$			D=1			D=2		
s_n	α	$\mathcal{E}_r$	Δ	t	Ops	Δ	t	Ops	Δ	t	Ops	Δ	t	Ops
		0.2	0.1%	62.0	0.4	0.1%	62.0	0.4	-0.1%	61.7	0.2	-0.1%	61.6	0.2
		0.4	1.5%	63.0	1.1	1.7%	66.8	1.7	0.3%	61.8	1.0	0.3%	62.4	1.2
		0.6	3.3%	64.5	1.9	4.8%	123.4	2.7	1.0%	62.4	1.9	1.4%	63.5	2.4
	1	1	9.5%	73.6	2.3	14.4%	506.7	3.8	4.5%	63.2	2.6	5.1%	70.2	3.8
		1.5	12.8%	87.5	2.2	19.4%	328.2	4.1	6.0%	63.3	2.6	6.8%	78.5	4.0
		2	12.8%	88.6	2.3	19.7%	366.3	4.0	7.2%	65.1	2.5	7.3%	402.1	4.1
		0.2	0.5%	62.0	0.4	0.5%	62.0	0.4	-0.1%	61.7	0.2	-0.1%	61.6	0.2
		0.4	3.3%	63.0	1.2	3.5%	67.0	1.8	0.6%	61.9	1.0	0.7%	62.4	1.1
		0.6	9.3%	64.4	2.0	10.4%	144.9	3.0	3.6%	62.7	2.3	4.0%	64.2	2.5
1	2	1	22.2%	76.0	3.2	28.5%	398.1	4.9	16.6%	63.7	3.4	20.4%	110.8	4.3
		1.5	27.4%	88.2	3.6	39.6%	361.8	5.2	26.3%	63.8	3.9	34.8%	112.9	5.5
		2	27.8%	86.0	3.7	40.5%	374.2	5.4	27.7%	63.9	4.0	44.6%	126.2	5.5
		0.2	0.5%	62.1	0.4	0.5%	62.1	0.4	-0.1%	61.7	0.2	-0.1%	61.7	0.2
		0.4	3.4%	62.9	1.2	3.6%	67.4	1.8	0.6%	61.9	1.1	0.8%	62.4	1.1
		0.6	10.4%	64.5	2.0	11.3%	143.8	3.1	4.6%	62.8	2.3	5.0%	64.2	2.6
	3	1	23.9%	78.8	3.3	31.7%	452.2	5.0	18.0%	63.7	3.7	22.3%	84.9	4.5
		1.5	32.8%	96.9	4.0	49.0%	392.2	5.3	29.7%	63.6	4.1	42.4%	116.5	5.4
		2	33.5%	97.0	4.0	51.2%	358.9	5.4	34.9%	63.7	4.3	59.8%	91.9	5.9
Average			13.0%	74.5	2.2	18.4%	241.0	3.2	10.1%	62.9	2.3	14.2%	97.7	3.0
		0.2	0.1%	62.9	0.6	0.2%	63.4	0.8	-0.2%	61.9	0.6	-0.1%	62.1	0.6
		0.4	2.5%	69.2	1.5	3.3%	218.6	2.7	0.6%	63.4	1.7	1.0%	65.7	2.4
		0.6	4.7%	162.9	2.0	6.6%	345.3	3.6	1.8%	64.8	2.5	2.7%	74.2	3.2
	1	1	10.5%	212.2	2.8	15.4%	262.5	4.1	5.4%	68.1	2.7	6.8%	91.3	4.4
		1.5	12.9%	225.1	2.4	18.6%	214.0	4.2	6.9%	66.7	2.4	7.1%	101.1	4.2
		2	12.9%	228.0	2.4	18.6%	232.5	4.4	7.2%	116.7	2.5	7.3%	551.0	4.3
		0.2	1.5%	62.8	0.7	1.5%	63.3	0.9	0.2%	61.9	0.7	0.3%	62.1	0.8
		0.4	7.4%	67.1	2.1	7.7%	244.2	3.1	2.0%	63.5	2.5	2.4%	68.2	2.8
		0.6	14.1%	142.7	3.0	14.8%	300.5	4.3	7.2%	66.6	3.3	8.6%	127.6	4.3
2	2	1	24.2%	187.7	4.4	30.6%	218.7	5.3	19.8%	81.0	4.4	23.3%	247.1	5.3
		1.5	29.0%	176.2	4.3	37.6%	221.4	5.0	28.0%	77.3	4.4	37.9%	130.3	5.5
		2	29.1%	177.8	4.2	38.1%	231.0	5.0	29.9%	91.7	4.6	44.5%	196.6	5.7
		0.2	1.6%	62.8	0.7	1.5%	63.3	0.9	0.2%	61.9	0.7	0.3%	62.1	0.7
		0.4	7.7%	69.8	2.1	7.8%	238.8	3.2	2.1%	63.5	2.4	2.6%	69.7	2.9
		0.6	15.9%	153.0	3.3	16.5%	290.2	4.4	8.4%	67.5	3.1	9.2%	153.9	4.2
	3	1	29.2%	220.4	4.8	35.3%	231.3	5.1	22.4%	78.3	4.7	26.1%	202.0	5.3
		1.5	37.2%	220.4	5.0	47.2%	207.7	5.1	38.8%	71.8	5.0	48.3%	143.7	5.8
		2	37.8%	249.8	4.7	48.5%	263.8	5.3	41.5%	72.6	5.0	60.0%	158.2	6.1
Average			15.4%	152.8	2.8	19.4%	217.3	3.7	12.4%	72.2	3.0	16.0%	142.6	3.8

Table 2.A.10: Heuristic results on instances with 20 and 50 vertices.

				$K=1$						$K=2$					
				$D=1$			$D=2$			$D=1$			$D=2$		
$\|V\|$	s_n	α	$\mathcal{E}_r$	Obj.	t	Ops	Obj.	t	Ops	Obj.	t	Ops	Obj.	t	Ops
20	1	1	0.2	398.7	68.3	1.0	397.6	79.7	1.5	278.0	63.4	0.9	277.7	63.6	1.1
			0.4	373.0	242.5	3.5	371.0	191.7	5.1	271.6	71.0	2.7	268.7	119.6	4.6
			0.6	368.8	164.5	4.0	350.5	187.3	6.2	266.4	96.0	3.3	259.4	299.9	5.9
			1	362.6	156.3	3.7	332.9	148.6	6.4	250.5	100.2	3.8	239.1	521.1	6.9
		2	0.2	393.3	87.4	2.3	392.4	103.2	2.8	276.2	64.1	2.0	274.9	64.6	2.1
			0.4	367.3	145.2	5.3	353.3	161.1	6.8	262.7	112.6	5.4	256.2	172.8	6.6
			0.6	340.7	143.4	6.5	307.0	196.9	8.4	242.6	183.9	6.8	233.5	414.1	8.8
			1	330.1	159.9	6.4	280.7	200.6	8.9	221.6	340.3	7.6	202.7	511.6	10.8
		3	0.2	392.9	87.4	2.2	392.0	105.1	2.8	276.2	64.1	2.0	274.9	64.5	2.1
			0.4	365.5	153.8	5.5	352.0	173.1	6.9	261.5	116.3	5.6	252.5	177.9	6.4
			0.6	336.6	157.6	6.8	296.0	211.2	8.4	236.9	189.3	7.1	229.3	451.6	9.1
			1	322.0	156.4	6.7	272.9	184.4	8.8	215.4	330.2	7.9	193.1	531.9	11.1
	Average			362.6	143.6	4.5	341.5	161.9	6.1	255.0	144.3	4.6	246.8	282.8	6.3
	2	1	0.2	394.9	156.4	2.3	397.1	166.3	2.8	277.2	66.8	1.8	276.5	71.5	2.4
			0.4	402.9	147.7	3.9	392.2	135.0	5.7	269.5	114.4	3.7	265.9	282.5	5.7
			0.6	397.2	120.7	4.4	365.1	152.9	6.9	263.9	224.9	4.4	259.5	292.3	6.9
			1	381.5	135.3	4.0	350.3	139.0	6.6	254.2	214.0	4.2	248.5	244.4	7.1
		2	0.2	394.1	145.9	4.8	397.8	161.7	5.4	272.2	79.7	3.6	271.3	87.7	4.3
			0.4	378.3	139.8	6.2	357.2	140.0	7.5	256.7	208.7	6.9	251.9	303.8	8.4
			0.6	357.5	130.8	6.1	325.3	137.0	8.3	240.1	326.5	7.8	233.7	361.9	9.8
			1	346.8	142.1	6.1	307.1	143.4	8.3	223.9	250.6	7.9	209.0	286.9	10.5
		3	0.2	393.0	151.9	4.7	396.7	161.7	5.5	271.8	77.8	3.7	270.9	89.7	4.5
			0.4	373.6	140.6	6.3	346.7	157.9	7.6	252.0	215.5	6.9	246.9	313.8	8.5
			0.6	342.3	142.5	6.2	309.2	147.4	8.2	233.7	292.5	8.2	226.4	379.8	10.2
			1	336.6	131.5	6.4	291.7	159.9	8.5	213.5	303.5	8.6	197.8	292.0	10.9
	Average			374.9	140.4	5.1	353.0	150.2	6.8	252.4	197.9	5.6	246.5	250.5	7.4
50	1	1	0.2	642.8	155.8	7.0	638.0	140.5	11.5	376.9	149.3	6.5	379.0	126.4	9.8
			0.4	597.8	163.1	9.2	578.1	127.4	17.4	366.2	129.6	10.1	360.9	122.9	17.4
			0.6	585.7	163.5	8.3	561.0	122.6	15.6	359.1	139.5	9.4	349.2	145.9	17.1
			1	583.6	148.7	7.9	552.9	135.5	14.3	353.1	143.9	8.7	343.2	136.0	15.1
		2	0.2	609.9	118.7	13.0	600.2	128.5	15.1	374.6	115.2	12.5	366.8	135.4	14.3
			0.4	536.9	120.3	16.2	516.3	103.6	22.1	340.5	127.2	17.8	321.6	193.4	23.2
			0.6	546.6	128.8	15.1	509.0	111.4	20.6	332.3	143.1	17.4	314.5	179.4	23.5
			1	556.8	146.3	13.0	514.4	129.8	18.8	338.4	138.1	15.6	314.4	173.2	21.9
		3	0.2	608.7	118.8	13.1	599.1	126.8	15.1	372.0	111.4	12.6	367.4	129.8	14.4
			0.4	533.6	116.0	16.7	515.3	96.0	21.9	338.9	124.4	18.2	319.6	211.2	23.4
			0.6	544.3	126.8	15.3	505.7	113.7	20.9	330.3	165.6	18.1	312.3	185.8	24.5
			1	554.1	141.0	13.7	511.3	136.9	19.0	338.2	137.7	15.9	314.2	169.0	21.7
	Average			575.1	137.3	12.4	550.1	122.7	17.7	351.7	135.4	13.6	338.6	159.0	18.9
	2	1	0.2	646.9	265.5	9.6	651.7	165.5	15.2	386.2	163.0	8.7	389.0	158.6	12.9
			0.4	603.3	276.1	10.2	605.2	169.7	17.9	370.4	192.6	10.6	372.9	132.4	18.4
			0.6	589.2	339.4	9.2	573.6	194.0	16.1	361.5	186.7	10.0	358.6	168.4	17.3
			1	586.1	386.0	7.9	557.3	230.4	14.4	355.2	215.0	8.8	348.1	171.5	15.6
		2	0.2	597.5	138.1	16.3	591.9	121.3	18.4	370.6	131.6	16.2	364.6	117.3	18.5
			0.4	554.2	174.9	15.5	532.6	132.6	20.0	346.2	142.2	17.5	336.2	156.7	22.7
			0.6	557.0	227.0	13.7	525.0	163.5	19.5	341.6	166.8	16.0	331.8	123.3	21.8
			1	564.5	281.5	12.0	526.1	185.2	17.6	341.6	195.1	13.8	326.0	153.4	19.8
		3	0.2	586.0	132.7	16.6	579.1	123.1	18.6	362.6	132.1	16.5	363.8	120.4	18.8
			0.4	542.0	172.4	15.7	518.1	129.7	20.3	338.9	138.1	18.1	331.4	138.7	22.4
			0.6	549.2	215.7	14.2	514.0	158.0	19.7	338.8	168.4	16.2	323.8	138.1	21.9
			1	559.3	272.6	12.6	520.7	192.6	17.5	339.1	197.9	14.2	322.1	156.5	20.5
	Average			577.9	240.2	12.8	557.9	163.8	17.9	354.4	169.1	13.9	347.4	144.6	19.2

3 A Matheuristic for the Vehicle Routing Problem with Drones and its Variants

Abstract

In this work, we are interested in studying the *Vehicle Routing Problem with Drones* (VRPD). Given a fleet of trucks, where each truck carries a given number of drones, the objective consists in designing feasible routes and drone operations such that all customers are served and minimal makespan is achieved. We formulate the VRPD as a *Mixed-Integer Linear Program* (MILP), which can be solved by any standard MILP solver. Moreover, with the aim of improving the performance of solvers, we introduce several sets of *Valid Inequalities* (VIEQs). Due to limited performance of the solvers in addressing large instances, we propose a matheuristic approach that effectively exploits the problem structure of the VRPD. Integral to this approach, we propose the *Drone Assignment and Scheduling Problem* (DASP) that, given an existing routing of trucks, looks for an optimal assignment and schedule of drones such that the makespan is minimized. In this context, we propose two MILP formulations for the DASP. In order to evaluate the performance of a state-of-the-art solver in tackling the MILP formulation of the VRPD, the benefit of the proposed VIEQs, and the performance of the matheuristic, we carried out extensive computational experiments. According to the numerical results, the use of drones can significantly reduce the makespan and the proposed VIEQs as well as the matheuristic approach have a significant contribution in solving the VRPD effectively.

3.1 Introduction

In recent years, drones have started to play an increasing role in logistics systems in both, academic research and practical context. Several companies in the logistics industry, such as Amazon Inc., Deutsche Post AG, UPS Inc. and the DPDgroup are actively investigating the potentials of drones for parcel delivery (see (Murray and Chu, 2015), and references therein). Furthermore, drones have been successfully applied in many public and private sectors including energy, agriculture and forestry, environmental protection, and emergency response (see (Otto et al., 2018), and references therein).

Most recently, several problems have been proposed in the academic literature that integrate drones into last-mile delivery, in an effort to lower costs and reduce delivery times (see, e. g., (Agatz et al., 2018; Murray and Chu, 2015; X. Wang et al., 2017)).

Drones might generally move faster between two locations than trucks due to not being restricted to the road network or congestion. In addition, drones and their payload are far more lightweight than trucks, which causes drones to consume less energy for the movement between two points. However, a drone's capacity is inherently limited to just one or few parcels. Moreover, as drones rely on comparatively small batteries for powering their flight, their range of operation is quite restricted compared to a commercial delivery truck that is powered by a fossil fuel or a high-capacity battery. Consequently, due to the complementary nature of trucks and drones, the integration of the latter into existing architectures in last-mile delivery might yield synergistic benefits.

With the objective of taking into account the potential benefits of using drones, the VRPD was proposed by X. Wang et al. (2017). They propose the integration of drones into the classical Vehicle Routing Problem. More precisely, given a fleet of trucks that are equipped with sets of drones, the objective consists in designing feasible routes and drone operations such that all customers are served and minimal makespan is achieved.

In this paper, we address the VRPD. The contributions of our work are manyfold:

1. To the best of our knowledge, we are the first to provide a formal specification of the multi-drone VRPD through a MILP. In particular, as opposed to existing works that consider a multi-truck case (see, e. g., (Sacramento et al., 2019; Schermer et al., 2019a)), our formulation allows for *cyclic* (loop) drone operations. Furthermore, we show that our model is very flexible and easily adaptable when deviating assumptions apply.

2. In order to improve the performance of MILP solvers in solving VRPD instances, we derive several sets of VIEQs. We show that our proposed VIEQs have a significant impact on the solver's performance in finding optimal solutions of small instances.

3. Nevertheless, using the MILP formulation to obtain (optimal) solutions in a reasonable amount of time is only possible for small instances. Therefore, in order to address larger instances of the VRPD, we propose a matheuristic that effectively exploits the problem structure of the VRPD. As an integral component of our matheuristic, we propose the DASP as a MILP. Indeed, given an existing routing of trucks, the DASP looks for the optimal assignment and schedule of a set of drones, such that the makespan is minimized.

4. In any case, through our numerical study, we show that the application of drones can significantly reduce the makespan, thus making them a valuable prospect in last-mile delivery. Furthermore, we provide an in-depth sensitivity analysis on relevant drone parameters.

The remainder of this paper is organized as follows. We begin in Section 3.2 by providing a brief overview of the existing academic literature that is related to the application of drones in last-mile delivery. Afterwards, Section 3.3 introduces the notation, mathematical model, and the proposed VIEQs for the VRPD. In Section 3.4, we explain the basic structure of the matheuristic and show how it effectively exploits the problem structure of the VRPD. Within this section, we propose the DASP as a mathematical program through two MILP formulations. Detailed results on the numerical studies are presented in Section 3.5. Finally, concluding remarks on this work and future research implications are provided in Section 3.6.

3.2 Related Literature

In this section, we provide a short overview on drone-related optimization problems in last-mile delivery. While there are many works that specifically consider routing a single drone or a swarm of drones (see, e. g., (Dorling et al., 2017)), for the purpose of this work, we limit ourselves to problems that require intensive *synchronization* (Drexl, 2012b) between trucks and drones. A more general literature review on civil applications of drones is given by Otto et al. (2018).

Murray and Chu (2015) introduced two novel problems, that are visualized in Figure 3.2.1, where a truck is working in tandem with a drone. Both problems assume that a depot and a set of customers, each of whom must be served exactly once by either the truck or drone, are given:

- In the *Flying Sidekick Taveling Salesman Problem* (FSTSP), the drone travels alongside the truck. Both start from a common depot and must return to the same depot at the end of the tour. At any customer location (or at the depot), the drone may be launched from the truck, starting an *operation* (or sortie), which is described as follows: the drone begins its flight, performs a delivery to a customer, and will then be retrieved by the truck at a later customer location (or the depot).
- In the *Parallel Drone Scheduling Traveling Salesman Problem* (PDSTSP), it is assumed that the depot is in close proximity to most customers. In this case, it might be beneficial to let the drone make its deliveries independently of the truck. Whereas the truck serves several customers that are located further away from the depot, the drone will continuously return to the depot to pick up a new parcel after each delivery.

Both problems share the objective of minimizing the makespan. However, in the case of the FSTSP extended synchronization between the truck and the drone is required.

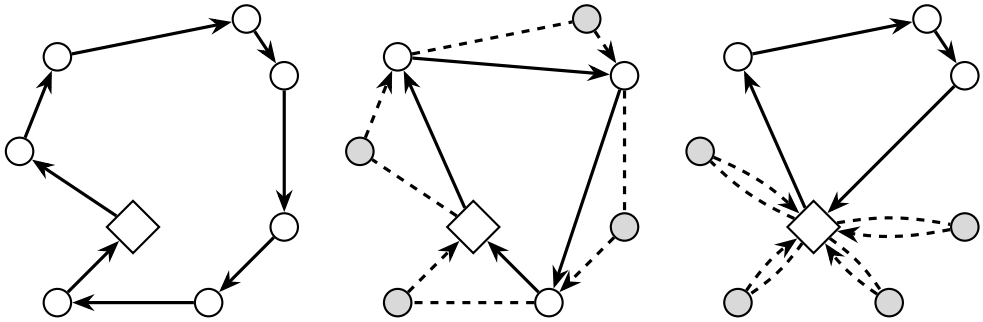

Figure 3.2.1: An illustration of TSP, FSTSP, and PDSTSP solutions where the truck's and drone's paths are indicated by solid and dashed lines, respectively.

Murray and Chu (2015) introduced MILP formulations as well as two heuristic methods for solving the FSTSP and PDSTSP. The authors conclude that, due to the $\mathcal{NP}$-hard nature of the proposed problems, only small-sized instances can be solved to optimality within a reasonable amount of time. Based on their experiments, they note that the FSTSP is significantly harder to solve due to the synchronization requirements. Finally, given a problem instance, Murray and Chu (2015) highlight that it is difficult to assess a priori whether treating a problem as the FSTSP or the PDSTSP will yield better results with regard to the objective value.

The *Traveling Salesman Problem with Drone* (TSPD) (Agatz et al., 2018) was proposed independently of the FSTSP but shares most of the common assumptions nonetheless. In this problem, a key difference to the FSTSP is that a drone can be retrieved at the same location from which it was launched by the truck and that drone operations are constrained by flight distance rather than time. The authors propose an *Integer Linear Programming* (ILP) formulation as well as heuristics for solving the TSPD. Agatz et al. performed a numerical study to assess the performance of their heuristics on various problem instances. The authors suggest extending their model in order to take into account multiple trucks (or drones) as well as to add different possibilities for recharging the drones.

Based on *Dynamic Programming* (DP), Bouman et al. (2018a) provide an exact solution method for the TSPD. The authors highlight that their method is able to find the optimal solution for problem instances with 10 vertices much faster compared to the ILP approach presented by (Agatz et al., 2018). In particular, Bouman et al. (2018a) also perform an in-depth analysis on the impact of (heuristically) adding additional constraints that restrict the number of drone operations. Based on their experiments, this can significantly reduce the time required to find the optimal solution, albeit, a non-optimal solution might be achieved after the imposition of these restrictions. They note that it might be beneficial to identify a procedure that detects unprofitable operations upfront. Furthermore, they

highlight that their method cannot be readily applied to cases where multiple drones are present.

Es Yurek and Ozmutlu (2018) provide an approach for solving the TSPD that is based on a decomposition of the problem into two components. First, all feasible tours to the *Traveling Salesman Problem* (TSP) are generated that visit only a subset of customers. Each tour provides a lower bound on a (feasible) TSPD solution that features the same TSP solution and serves the remaining customers by drones. Starting with the shortest tour, mathematical programming is used to find the optimal drone operations such that the customers that were not contained in the TSP tour are served by drone. Here, the objective is to minimize the waiting time of the truck such that the objective value after solving the mathematical model is as close as possible to the lower bound that is associated with the initial TSP tour. This procedure continues until the best remaining lower bound (of a tour that has not been examined) is worse than the incumbent TSPD solution. Es Yurek and Ozmutlu (2018) compare their results to (Agatz et al., 2018; Murray and Chu, 2015) and highlight that they are able to solve instances with 10 to 12 vertices in shorter computation time. However, the authors highlight that their approach might be limited to small instances due to the enumerative nature of the approach.

Carlsson and Song (2018) worked on a variant of the TSPD called the *Horsefly Routing Problem* w. r. t. a commercially available system with the same name. Compared to the TSPD, their variant features a single truck that can be equipped with one or more drones. The authors propose a heuristic method and use large instances (with up to 500 vertices) as well as practical data (with up to 100 vertices). One of their key findings is the hypothesis that the potential benefit of using a drone in tandem with a truck is proportional to the square root of the relative velocity between the truck and the drone.

X. Wang et al. (2017) introduced the VRPD as a generalization of the TSPD. In the VRPD a fleet of trucks, each truck equipped with a given number of drones, is tasked with delivering parcels to customers. The drones may be launched from the truck at the depot or at any customer location and may be retrieved by the truck at a different customer location (or at the depot, concluding its tour). When a drone is launched, it can serve exactly one customer before it needs to return to the truck such that it can be resupplied. In contrast to classic VRPs, the objective is to minimize the makespan, i. e., the time required to serve all customers such that the entire fleet has returned to the depot. Wang et al. studied the problem from a theoretical point of view. In particular, they introduced several upper bounds on the amount of time that can be saved by employing drones. The upper bounds are obtained by investigating the structure of optimal solutions and depend on the relative velocity of the drones compared to the trucks and the number of drones per truck. Poikonen et al. (2017) refined the work of X. Wang et al. (2017)

by considering the impact on the previously introduced bounds when integrating limited battery life, different distance metrics, and operational expenditures of deploying drones and trucks in the objective function. The authors suggest the possibility of launching and retrieving drones from arbitrary locations instead of being restricted to customer locations. Furthermore, as a future research direction, the authors suggest developing models, heuristics, and benchmark VRPD instances.

Di Puglia Pugliese and Guerriero (2017) introduced the *Vehicle Drone Routing Problem with Time Windows* (VDRPTW), where the objective consists in serving customers at minimum cost using a truck and drones, while respecting the time-window restrictions. The authors provide a mathematical model for the VDRPTW and use a commercial solver for solving the problem on randomly generated instances with 5 or 10 customers. The authors conclude that the benefit of drones for last-mile delivery strongly depends on the marginal cost per mile of both drones and trucks.

Boysen et al. (2018b) proposed the *Drone Scheduling Problem* (DSP), where a fixed sequence of customers that are visited by a single truck and a disjoint set of customers that must be served by drone(s) are given. In this problem, they look for the optimal schedule of drones, i. e., launch and retrieval locations, such that the makespan is minimized. The authors differentiate between a truck that is equipped with a single or multiple drones and different degrees of freedom regarding feasible launch and retrieval locations. Particularly, for special cases that restrict potential launch and retrieval locations, polynomial time algorithms can be derived. They propose two different MILPs for solving the DSP and conclude that the DSP might also be used for solving a more holistic problem such as the TSPD. In this case, a subsequence of customers visited by the truck might be generated and the drones schedule for the remaining customers derived through the DSP (similar to the work of Es Yurek and Ozmutlu (2018)).

Ha et al. (2018) considered a variant of the FSTSP where the objective is to minimize the operational costs. These costs include the variable cost of driving and flying and might also include costs for the time spent waiting by the truck and drone. In particular, the authors adapt the MILP model and heuristics proposed by Murray and Chu (2015) to their new problem. Furthermore, they propose a *Greedy Randomized Adaptive Search Procedure* (GRASP) as a solution method. The authors can show that their MILP formulation can be used for solving small-sized instances under the min-cost objective within a few minutes. Moreover, the proposed GRASP heuristic is able to achieve optimal results in all cases for 5 small instances with 10 vertices. Notably, Ha et al. (2018) also test their heuristic under the min-time objective. A detailed numerical study reveals the effectiveness of their approach. According to their experiments, under the min-cost objective, the costs are reduced by about 30% w. r. t. truck-only solutions (with an increase in the makespan by

typically 50%). Moreover, under the min-time objective, the makespan is reduced by about 10% (with a decrease in cost by typically 20%).

Sacramento et al. (2019) investigated a variant of the FSTSP, in which multiple trucks are present and call it the VRPD. The authors consider an objective where the operational costs are minimized under the restriction of a maximum duration for all routes. For this purpose, the MILP formulation of Murray and Chu (2015) is adapted and an *Adaptive Large Neighborhood Search* (ALNS) procedure proposed. As integral components of their ALNS algorithm, the authors suggest several problem-specific *destroy* and *repair* methods. Sacramento et al. (2019) show that their MILP formulation can be used to solve some small-sized instances with 6, 10, and 12 vertices to optimality. However, significant increases in runtime are observed with an increase in the instance size. On larger instances, they can show that the truck-drone fleet allows cost-savings of typically 20 to 30% compared to truck-only delivery.

The possibility of *en route* operations in the context of the VRPD has been studied in (Schermer et al., 2019a). En route operations were defined by Marinelli et al. (2018) as the possibility of constructing arc-based truck-drone operations. In (Schermer et al., 2019a), a formal specification of the VRPD with en route operations is given through a MILP. In this MILP, en route operations can be performed at some discrete points on each arc. Moreover, an algorithm that follows the framework of *Variable Neighborhood Search* (VNS) is proposed. Through a computational study, it is revealed that, in some cases, en route operations allow for additional savings, compared to the non en route case. Moreover, a strong dependence of these additional savings on drone parameters such as their velocity or endurance is shown. Albeit, it is revealed that the possibility of en route operations makes the problem significantly more difficult to solve.

Table 3.2.1 provides an overview of the problems that were introduced in this section. As we have already established, in contrast to classic VRPs (see, e. g., (Toth and Vigo, 2014) and references therein), the objective in these problems often consists in minimizing the makespan rather than cost. Furthermore, if multiple trucks are allowed, there is no fixed cost incurred when an additional truck is deployed. Rather, the number of available trucks and drones per truck (fleet) is considered a parameter of the problem itself as opposed to a decision-relevant component (see (Di Puglia Pugliese and Guerriero, 2017; Sacramento et al., 2019; X. Wang et al., 2017)). Finally, problem assumptions often differ with regard to *cyclic* (loop) drone operations, i. e., the possibility of drone operations that start and end at the same vertex (while the truck remains stationary). In fact, even though all problems allow acyclic operations, cyclic operations are often not permitted.

Table 3.2.1: Overview of drone-related optimization problems with extensive synchronization requirements presented in Section 3.2.

Reference	Trucks	Drones	Obj.	Cyclic Ops.	Contribution	Vertices
Murray and Chu (2015)	1	1	time	no	MILP, heuristic	10 ... 20
Poikonen et al. (2017) and X. Wang et al. (2017)	n	m	time	yes	theoretical insights	-
Di Puglia Pugliese and Guerriero (2017)	n	m	cost	no	MILP	5 ... 10
Agatz et al. (2018) and Bouman et al. (2018a)	1	1	time	yes	IP, DP, heuristic	10 ... 100
Carlsson and Song (2018)	1	m	time	no	heuristic	25 ... 500
Es Yurek and Ozmutlu (2018)	1	1	time	no	heuristic	10 ... 20
Boysen et al. (2018b)	1	m	time	yes	MILP, SA	5 ... 100
Ha et al. (2018)	1	1	cost/time	no	MILP, GRASP	10 ... 100
Sacramento et al. (2019)	n	1	cost	no	MILP, ALNS	6 ... 200
Schermer et al. (2019a)	n	m	time	no	MILP, VNS	10 ... 50
This work	n	m	time	yes	MILP, matheuristic	10 ... 100

3.3 Problem Definition

In this paper, we are interested in studying the VRPD as proposed by X. Wang et al. (2017). We recall that the VRPD asks for routing a fleet of trucks, each truck equipped with a set of drones, and seeks to minimize the makespan, i. e., the time required to serve all customers using the trucks and the drones such that, by the end of the mission, all trucks and drones have returned to the depot.

In Section 3.3.1, we formulate the VRPD as MILP model. Afterwards, in Section 3.3.2, we provide additional VIEQs. As a key difference to existing models (refer to Table 3.2.1), our model permits not only multiple trucks, but also multiple drones per truck and cyclic drone operations, i. e., operations that start and end at the same vertex.

3.3.1 Notation and Mathematical Model

Suppose that a set of customer locations, each with a uniform demand, and a fleet of homogeneous trucks, each carrying the same number of homogeneous drones, are given. In the VRPD, we look for minimizing the makespan required to serve all customers by using the given fleet such that, by the end of the mission, all trucks and drones must be at the depot (Poikonen et al., 2017; X. Wang et al., 2017). Furthermore, we make the following assumptions regarding the nature of drones (Agatz et al., 2018; Murray and Chu, 2015; X. Wang et al., 2017):

- When launched from the truck, a drone can serve exactly one customer before it needs to return to the same truck from which it was launched. Furthermore, we ask

that each truck has sufficient capacity.

- We assume that a drone has a limited endurance of $\mathcal{E}$ *distance units* per operation. After returning to the truck, the battery of the drone is recharged (or swapped) instantaneously. For a more detailed discussion on this assumption, we refer to Appendix 3.C.
- Associated with each launch and retrieval of a drone are two overhead times, $\bar{t}_l$ and $\bar{t}_r$. Furthermore, we assume that the truck is capable of launching and retrieving drones in parallel.
- Drones may only be dispatched and picked up from vertices, i. e., at the depot or any other customer location.

Inspired by the work of Di Puglia Pugliese and Guerriero (2017), we use the following notation in order to formulate the VRPD. Assume that a complete graph $\mathcal{G} = (V, E)$ is given, where V is the set of vertices (or nodes) and E is the set of edges. The set V contains n vertices associated with the customers, named $V_N = \{1, 2, \ldots, n-1, n\}$ and two extra vertices 0 and $n+1$ that (in order to simplify the notation and the mathematical formulation) both represent the same depot location at the start and end of each tour, respectively. Thus, $V = \{0, 1, \ldots, n, n+1\}$, where $0 \equiv n+1$. In addition, we call $V_D \subseteq V_N$ the *drone-serviceable* subset of customers.

We refer to $V_L = V \setminus \{n+1\}$ as the subset of vertices in V from which drones may be launched and retrieved after performing cyclic operations (that start and end at the same location). Furthermore, $V_R = V \setminus \{0\}$ denotes the subset of vertices where drones may be retrieved after performing an acyclic operation.

By the parameters d_{ij} and $\overline{d}_{ij}$ we define the distance required to travel from vertex i to vertex j by truck and drone, respectively. Additionally, as the drone may not be limited to the road network, we can assume that $\overline{d}_{ij} \leq d_{ij} \, \forall i, j \in V$. It is reasonable to assume that the triangle inequality applies to these distances.

We consider a limited set $\mathcal{K} = \{1, \ldots, K_n\}, K_n \in \mathbb{Z}_{>0}$ of $|\mathcal{K}| = K_n$ trucks. Furthermore, each truck $k \in \mathcal{K}$ holds an equal set of $\mathcal{D} = \{1, \ldots, D_n\}, D_n \in \mathbb{Z}_{>0}, |\mathcal{D}| = D_n$ drones. Each drone may travel a maximum range of $\mathcal{E}$ distance units per operation, where a *drone-operation* is characterized by a triple (i, w, j) as follows: the drone is launched from a truck at a vertex $i \in V_L$, delivers a parcel to $w \in V_N$, and is retrieved by the same truck from which it was launched at vertex $j \in V$ $(i \neq w, w \neq j)$. If $i = j$, (i. e., the operation is (i, w, i)) we call it a *cyclic* (loop) operation, otherwise, we call it an *acyclic* operation.

We use the decision variables in Table 3.3.1 for modeling the VRPD. The MILP formulation of the VRPD is given in (3.1)–(3.27) where (3.1) is the objective function and the

Table 3.3.1: Decision variables used in the VRPD formulation.

τ	Continuous variable that indicates the makespan.
x_{ij}^k $\forall i,j \in V, k \in \mathcal{K}, i \neq j$	Binary variables that state whether arc (i,j) is used by truck k.
y_{iwj}^{kd} $\forall i \in V_L, w \in V_D, j \in V, k \in \mathcal{K}, d \in \mathcal{D}, i \neq w, w \neq j$	Binary variables that specify whether a drone d associated with truck k performs the operation (i,w,j).
p_{ij}^k $\forall i,j \in V, k \in \mathcal{K}, i \neq j$	Binary variables that indicate whether vertex i is served before but not necessarily adjacent to vertex j in the route of truck k.
u_i^k $\forall i \in V, k \in \mathcal{K}$	Continuous variables with a lower bound of 0 that state the position of vertex i in the route of truck k.
z_{aiwje}^{kd} $\forall a,i,w,j,e \in V, k \in \mathcal{K}, d \in \mathcal{D}, \lvert\{a,i,w,j,e\}\rvert = 5$	Binary variables that specify whether the drone delivery y_{awe}^{kd} is performed circumjacent to x_{ij}^k, i. e., drone d is launched at vertex a before the truck k has reached i and retrieved at vertex e after the truck has reached j. The term $\lvert\{a,i,w,j,e\}\rvert = 5$ implies that all vertices are pairwise different.
a_i^k $\forall i \in V, k \in \mathcal{K}$	Continuous variables with a lower bound of 0 that indicate the arrival time of truck k at i.
s_i^k $\forall i \in V, k \in \mathcal{K}$	Continuous variables with a lower bound of 0 that indicate the earliest possible departure time of truck k at i.
r_i^{kd} $\forall i \in V, k \in \mathcal{K}, d \in \mathcal{D}$	Continuous variables with a lower bound of 0 that indicate the earliest possible release time of drone d that belongs to truck k at i.
s_i^{kd} $\forall i \in V, k \in \mathcal{K}, d \in \mathcal{D}$	Continuous variables with a lower bound of 0 that indicate the earliest time at which drone d that belongs to truck k is ready to depart from i.
y_i^{kd} $\forall i \in V, k \in \mathcal{K}, d \in \mathcal{D}$	Binary variables that take value 1, if drone d that belongs to truck k is available for an operation at i.

constraints are given by (3.2)–(3.27). Please note, due to the large number of variables involved in the model, for the sake of compactness, all constraints that restrict the domains of binary variables to the set $\{0,1\}$, integer variables to $\mathbb{Z}^+$, and continuous variables to take a value within their respective intervals $[0,\infty)$ are not listed explicitly.

$$\min \quad \tau, \tag{3.1}$$

$$\text{s. t.} \quad \tau \geq s_{n+1}^k \quad \forall k \in \mathcal{K}, \tag{3.2}$$

$$\sum_{j \in V_R} x_{0j}^k - \sum_{i \in V_L} x_{i,n+1}^k = 0 \quad \forall k \in \mathcal{K}, \tag{3.3}$$

$$\sum_{\substack{i \in V_L, \\ i \neq h}} x_{ih}^k - \sum_{\substack{j \in V_R, \\ h \neq j}} x_{hj}^k = 0 \quad \forall k \in \mathcal{K}, h \in V_N, \tag{3.4}$$

$$\sum_{j \in V_R} x_{0j}^k = 1 \quad \forall k \in \mathcal{K}, \tag{3.5}$$

$$\sum_{k \in \mathcal{K}} \sum_{\substack{i \in V_L, \\ i \neq w}} x_{iw}^k + \sum_{k \in \mathcal{K}} \sum_{d \in \mathcal{D}} \sum_{\substack{i \in V_L, \\ i \neq w}} \sum_{\substack{j \in V, \\ w \neq j}} y_{iwj}^{kd} = 1 \quad \forall w \in V_D, \tag{3.6}$$

$$\sum_{k \in \mathcal{K}} \sum_{\substack{i \in V_L, \\ i \neq w}} x_{iw}^k = 1 \quad \forall w \in V_N \setminus V_D, \tag{3.7}$$

$$y_{0wj}^{kd} \leq \sum_{\substack{h \in V_L, \\ h \neq j}} x_{hj}^k \quad \forall k \in \mathcal{K}, d \in D, w \in V_D, j \in V_R, w \neq j, \tag{3.8}$$

$$2y_{iwj}^{kd} \leq \sum_{\substack{h \in V_L, \\ h \neq i}} x_{hi}^k + \sum_{\substack{l \in V_N, \\ l \neq j}} x_{lj}^k \quad \begin{array}{l} \forall k \in \mathcal{K}, d \in D, i \in V_N, \\ w \in V_D, j \in V_R, |\{i, w, j\}| = 3, \end{array} \tag{3.9}$$

$$y_{iwi}^{kd} \leq \sum_{\substack{h \in V_L, \\ h \neq i}} x_{hi}^k \quad \forall k \in \mathcal{K}, d \in D, i \in V_N, w \in V_D, i \neq w, \tag{3.10}$$

$$u_i^k - u_j^k + 1 \leq (n+1)(1 - x_{ij}^k) \quad \forall k \in \mathcal{K}, i \in V_N, j \in V_R, i \neq j, \tag{3.11}$$

$$u_i^k - u_j^k + 1 \leq (n+1)(1 - y_{iwj}^{kd}) \quad \begin{array}{l} \forall k \in \mathcal{K}, d \in \mathcal{D}, i \in V_N, \\ w \in V_D, j \in V_R, |\{i, w, j\}| = 3, \end{array} \tag{3.12}$$

$$x_{ij}^k \leq p_{ij}^k \quad \forall k \in \mathcal{K}, i \in V_L, j \in V_R, i \neq j, \tag{3.13}$$

$$u_i^k - u_j^k \geq 1 - (n+1)p_{ij}^k \quad \forall k \in \mathcal{K}, i \in V_N, j \in V_R, i \neq j, \tag{3.14}$$

$$u_i^k - u_j^k \leq -1 + (n+1)(1 - p_{ij}^k) \quad \forall k \in \mathcal{K}, i \in V_N, j \in V_R, i \neq j, \tag{3.15}$$

$$3z_{aiwje}^{kd} \leq y_{awe}^{kd} + p_{ai}^k + p_{je}^k \quad \begin{array}{l} \forall k \in \mathcal{K}, d \in \mathcal{D}, a \in V_L, i, j \in V_N, \\ w \in V_D, e \in V_R, |\{a, i, w, j, e\}| = 5, \end{array} \tag{3.16}$$

$$2 + z_{aiwje}^{kd} \geq y_{awe}^{kd} + p_{ai}^k + p_{je}^k \quad \begin{array}{l} \forall k \in \mathcal{K}, d \in \mathcal{D}, a \in V_L, i, j \in V_N, \\ w \in V_D, e \in V_R, |\{a, i, w, j, e\}| = 5, \end{array} \tag{3.17}$$

$$M(x_{ij}^k - 1) + s_i^k + t_{ij} \leq a_j^k \quad \forall k \in \mathcal{K}, i \in V_L, j \in V_R, i \neq j, \tag{3.18}$$

$$a_i^k \leq s_i^k \quad \forall k \in \mathcal{K}, i \in V, \tag{3.19}$$

$$\begin{array}{r} M(y_{iwj}^{kd} - 1) + s_i^{kd} \\ +(\bar{t}_l + \bar{t}_{iw} + \bar{t}_{wj} + \bar{t}_r) \leq r_j^{kd} \end{array} \quad \begin{array}{l} \forall k \in \mathcal{K}, d \in \mathcal{D}, i \in V_L, w \in V_D, \\ j \in V_R, |\{i, w, j\}| = 3, \end{array} \tag{3.20}$$

$$r_i^{kd} + \sum_{\substack{w \in V_D, \\ i \neq w}} (\bar{t}_l + \bar{t}_{iw} + \bar{t}_{wi} + \bar{t}_r) y_{iwi}^{kd} \leq s_i^{kd} \quad \forall k \in \mathcal{K}, d \in \mathcal{D}, i \in V_L, \tag{3.21}$$

$$s_i^{kd} + \bar{t}_l \sum_{\substack{w \in V_D, \\ i \neq w}} \sum_{\substack{j \in V_R, \\ |\{i,w,j\}|=3}} y_{iwj}^{kd} \leq s_i^k \quad \forall k \in \mathcal{K}, d \in \mathcal{D}, i \in V, \tag{3.22}$$

$$a_j^k + \bar{t}_r \sum_{\substack{i \in V_L \\ i \neq j}} \sum_{\substack{w \in V_D, \\ |\{i,w,j\}|=3}} y_{iwj}^{kd} \leq r_j^{kd} \quad \forall k \in \mathcal{K}, d \in \mathcal{D}, j \in V, \tag{3.23}$$

$$(\bar{d}_{iw} + \bar{d}_{wj}) y_{iwj}^{kd} \leq \mathcal{E} \quad \begin{array}{l} \forall k \in \mathcal{K}, d \in \mathcal{D}, i \in V_L, w \in V_D, \\ j \in V, i \neq w, w \neq j, \end{array} \tag{3.24}$$

$$\begin{aligned} M(x_{ij}^k - 1) + y_i^{kd} + \sum_{\substack{a \in V_L, \\ a \neq i, \\ a \neq j}} \sum_{\substack{w \in V_D, \\ i \neq w, \\ |\{a,w,j\}|=3}} y_{awj}^{kd} \\ + \sum_{\substack{a \in V_L, \\ a \neq i, \\ a \neq j}} \sum_{\substack{w \in V_D, \\ |\{a,i,w,e\}|=4}} \sum_{\substack{e \in V_R, \\ |\{a,i,w,j,e\}|=5}} z_{aiwje}^{kd} \leq 1 \quad \forall k \in \mathcal{K}, d \in \mathcal{D}, i \in V_L, j \in V_R, i \neq j, \end{aligned} \tag{3.25}$$

$$M(x_{ij}^k - 1) + \sum_{\substack{w \in V_D, \\ i \neq w}} \sum_{\substack{e \in V_R, \\ |\{i,w,e\}|=3}} y_{iwe}^{kd} \leq y_i^{kd} \quad \forall k \in \mathcal{K}, d \in \mathcal{D}, i \in V_L, j \in V_R, i \neq j, \tag{3.26}$$

$$M(x_{ij}^k - 1) + \sum_{\substack{w \in V_D, \\ i \neq w}} y_{iwi}^{kd} \leq M y_i^{kd} \quad \forall k \in \mathcal{K}, d \in \mathcal{D}, i \in V_L, j \in V_R, i \neq j. \tag{3.27}$$

The objective function, represented by (3.1), minimizes the makespan through constraints (3.2). Constraints (3.3)–(3.5) form the preservation of flow for the trucks. More precisely, constraints (3.3) guarantee that each truck that departs from the depot must also return to the depot. Constraints (3.4) ensure that each truck that visits a customer $h \in V_N$ must also depart from the same customer. Due to constraints (3.5), each truck $k \in \mathcal{K}$ must start from the depot exactly once (note that a truck might remain stationary by following the arc $x_{0,n+1}^k$).

Constraints (3.6) ensure task synchronization, i. e., each customer $j \in V_D$ must be served exactly once by either a truck or a drone. Furthermore, constraints (3.7) guarantee that the non drone-eligible customers are served by a truck. Movement synchronization between routes of the trucks and its drones is handled by constraints (3.8)–(3.10). Constraints (3.8) handle the synchronization between the truck and drone in the case that a drone is launched from the depot. In the case of constraints (3.9), if a drone performs an acyclic operation (i, w, j), then truck $k \in \mathcal{K}$ must visit vertices $i \in V_n$ and $j \in V_R$ at some point to allow a launch and retrieval of the drone at vertices i and j, respectively. Constraints (3.10) ensure that cyclic operations are only possible, if the truck $k \in \mathcal{K}$ visits i.

Constraints (3.11)–(3.12) link the variables u_i^k, for truck $k \in \mathcal{K}$, to the decision variables x_{ij}^k and y_{iwj}^k. When i is adjacent to j, constraints (3.13) provide a strong coupling between x_{ij}^k and p_{ij}^k. For the remaining cases, constraints (3.14)–(3.15) associate the variables p_{ij}^k with u_i^k and u_j^k. We ask that $u_0 = 0$ and $p_{0j} = 1 \, \forall j \in V_R$. Constraints (3.16)–(3.17)

determine a value on the variables z^k_{aiwje} depending on the decision variables p^k_{ij} and y^k_{iwj}.

The set of constraints (3.18)–(3.23) guarantee temporal operational synchronization by providing lower bounds on the arrival and departure times for the trucks and drones. For each possible transition of a truck and acyclic operation of a drone, constraints (3.18) and (3.20) set lower bounds on the arrival times and release times for the trucks and drones, respectively. Furthermore, constraints (3.19) enforce the arrival time as a lower bound on the departure time of the truck. In case of cyclic operations, constraints (3.21) set lower bounds on the earliest departure time of the drone. More precisely, a drone can depart from a vertex as soon its cyclic operations have been completed. In contrast, the truck must wait for the last drone to complete its cyclic operation which is handled through constraints (3.22). Finally, constraints (3.23) guarantee that a drone can only be released after the truck has arrived. Constraints (3.22) (and (3.23)) also incorporate the overhead times $\bar{t}_l$ (and $\bar{t}_r$) in the case that a drone is launched for (retrieved after) an acyclic operation.

Constraints (3.24) impose a maximum distance (endurance) constraint on each drone operation (i, w, j). Finally, constraints (3.25)–(3.27) guarantee that during each transition of the truck from vertex i to vertex j a feasible number of drone operations are performed (see Figure 3.3.1). More precisely, constraints (3.25) force the variables y^d_i to take value 0, if a circumjacent operation is performed. Furthermore, the set of constraints (3.26) and (3.27) ensure that acyclic and cyclic operations can only be performed if the drone is available at i.

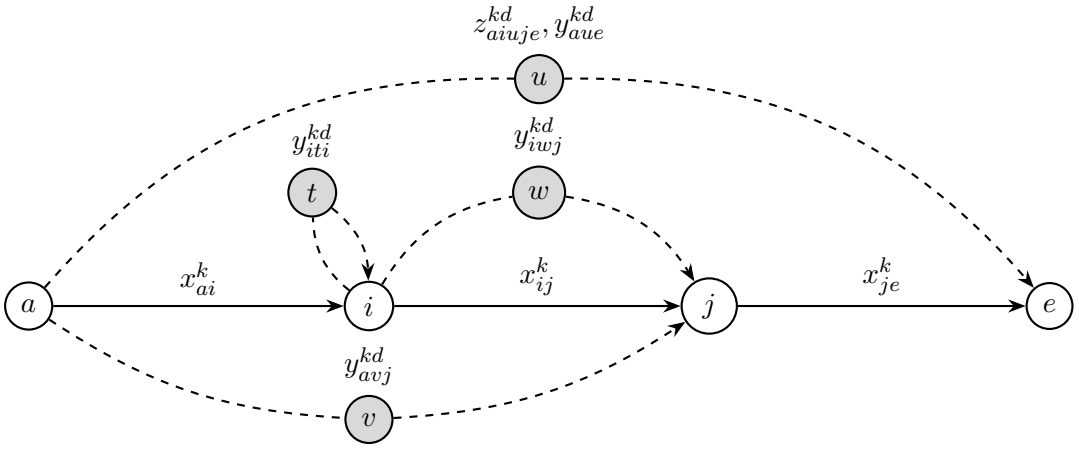

Figure 3.3.1: A visualization of some possible operations that might occur as a truck proceeds from i to j. In particular, if drone d performs a circumjacent operation such as, e. g., either y^{kd}_{avj} or y^{kd}_{aue}, no operation can be performed by drone d at i. Otherwise, the drone might perform a number of cyclic operations such as y^{kd}_{iti} followed by a single acyclic operation such as y^{kd}_{iwj}.

Any standard MILP solver might be used to solve MILP (3.1)–(3.27). However, a glance at MILP (3.1)–(3.27) reveals that size of the model grows significantly with an increase in the size of $\mathcal{G}$ as well as the sets $\mathcal{K}$ and $\mathcal{D}$. In particular, tracking circumjacent drone operations requires a significant modeling effort; and, compared to related models (Di Puglia Pugliese and Guerriero, 2017; Schermer et al., 2019a), cyclic drone operations lead to several additional variables and constraints. A detailed discussion on the flexibility of the model to deviating assumptions can be found in the Appendix 3.A of this paper.

3.3.2 Valid Inequalities

In order to enhance the performance of the MILP solvers in solving the VRPD, we derive several VIEQs as cuts in the solution space of the problem that exclude none of the optimal solutions. Hence, the addition of VIEQs might have a positive impact on the computational time that is required for solving the VRPD.

3.3.2.1 Makespan Bounds

In this section, we derive two sets of VIEQs that are used to place *Lower Bounds* (LBs) on the objective value τ. While the derivation of these VIEQ is relatively straightforward, they have a significant impact on the performance of the solver by providing an improved linear relaxation to the solver (refer to Section 3.5).

Proposition 3.1. In the MILP (3.1)–(3.27), τ is bounded by the time spent traveling by trucks.

Proof. For each truck $k \in \mathcal{K}$, the time spent traveling can be calculated as follows:

$$\sum_{i \in V_L} \sum_{\substack{j \in V_R, \\ i \neq j}} t_{ij} x_{ij}^k \quad \forall k \in \mathcal{K}. \tag{3.28}$$

Furthermore, the time spent waiting at vertices can be accounted as follows:

$$\sum_{i \in V} (s_i^k - a_i^k) \quad \forall k \in \mathcal{K}. \tag{3.29}$$

The variable τ marks the time at which all trucks and all drones have arrived at vertex $n+1$. In a feasible VRPD solution, a truck spends time traveling on arcs and might spend further time waiting on vertices for a drone to be retrieved. Therefore, the time spent traveling and waiting by each truck qualifies as a valid lower bound on the objective value

as follows:

$$\sum_{i \in V} (s_i^k - a_i^k) + \sum_{i \in V_L} \sum_{\substack{j \in V_R, \\ i \neq j}} t_{ij} x_{ij}^k \leq \tau \quad \forall k \in \mathcal{K}. \tag{3.30}$$

This completes the proof of Proposition 3.1. ■

Proposition 3.2. In the MILP (3.1)–(3.27), τ is bounded by the time spent traveling by each drone.

Proof. For each drone $d \in \mathcal{D}$ that belongs to a truck $k \in \mathcal{K}$, the total time spent traveling can be accounted as follows:

$$\sum_{i \in V_L} \sum_{\substack{w \in V_D, \\ i \neq w}} (\bar{t}_l + \bar{t}_{iw} + \bar{t}_{wi} + \bar{t}_r) y_{iwi}^{kd} + \sum_{i \in V_L} \sum_{w \in V_D,} \sum_{\substack{j \in V_R, \\ i \neq j, \\ w \neq j}} (\bar{t}_l + \bar{t}_{iw} + \bar{t}_{wi} + \bar{t}_r) y_{iwj}^{kd} \quad \forall k \in \mathcal{K}, d \in \mathcal{D}. \tag{3.31}$$

The variable τ marks the time at which all trucks and all drones have arrived at vertex $n+1$. In a feasible VRPD solution, a drone spends time traveling on arcs and might spend further time hovering on vertices after performing an acyclic operation for the truck to arrive, such that it can be retrieved. Thus, the time spent traveling by each drone qualifies as a valid lower bound on the objective value as follows:

$$\sum_{i \in V_L} \sum_{\substack{w \in V_D, \\ i \neq w}} (\bar{t}_l + \bar{t}_{iw} + \bar{t}_{wi} + \bar{t}_r) y_{iwi}^{kd} + \sum_{i \in V_L} \sum_{w \in V_D,} \sum_{\substack{j \in V_R, \\ i \neq j, \\ w \neq j}} (\bar{t}_l + \bar{t}_{iw} + \bar{t}_{wi} + \bar{t}_r) y_{iwj}^{kd} \leq \tau \quad \forall k \in \mathcal{K}, d \in \mathcal{D}. \tag{3.32}$$

Therefore, we have proven Proposition 3.2. ■

3.3.2.2 Symmetry in the VRPD

In this section, we show that in the case of a symmetric graph $\mathcal{G}$, the VRPD is a symmetric problem, hence, has symmetric solutions. We use this property in order to derive a set of *symmetry-breaking cuts*, and for this purpose, we introduce the following proposition.

Proposition 3.3. Assume that we have a solution for the MILP (3.1)–(3.27) with components x_{ij}^k and y_{iwj}^k. Furthermore, in the given solution, let π^k defines the sequence of customers visited by the truck k, for each truck $k \in \mathcal{K}$. Then, the following statements are true:

- (i) The VRPD is a symmetric problem, i. e., corresponding to π^k, there is another solution, having the same objective value, consisting of reverse order of visiting customers as it is in π^k.
- (ii) The following inequality breaks the symmetry in the VRPD

$$\sum_{i \in V_N} i \cdot x^k_{0,i} \leq \sum_{j \in V_N} j \cdot x^k_{j,n+1} \quad \forall k \in \mathcal{K}. \tag{3.33}$$

Proof. (i) In the arc-based formulation MILP (3.1)–(3.27), for each truck $k \in \mathcal{K}$, a feasible selection of decision variables x^k_{ij} and y^k_{iwj} imply makespan s^k_{n+1} and a directed graph $\mathcal{G}(V, A)$. We call $\mathcal{G}^T(V, A^T)$ the transpose graph of $\mathcal{G}$. We prove that the same makespan s^k_{n+1} is associated with $\mathcal{G}$ and the transpose graph $\mathcal{G}^T$. We consider the following three possible cases:

1. If no drones are present, $\mathcal{G}$ is a simple path. In this case, the longest path through $\mathcal{G}$ (which corresponds to s^k_{n+1}) is equal to the inverse longest path through the transpose graph $\mathcal{G}^T$.
2. Next, assume that acyclic drone operations occur. In this case, $\mathcal{G}$ is no longer a simple path but remains a directed acyclic graph nonetheless. Thus, the longest path through $\mathcal{G}$ (which corresponds to s^k_{n+1}) equals the longest path through $\mathcal{G}^T$.
3. Finally, consider the general cases where acyclic and cyclic drone operations occur. In this case, based on the structure of the problem, any directed graph G with cyclic operations can be transformed into directed acyclic graph through the introduction of a dummy vertex for each cyclic operation (see Figure 3.3.2). Therefore, this case can be reduced to previous case.

(ii) For each truck $k \in \mathcal{K}$, the decision variables x^k_{ij} directly correspond to a sequence of customers π^k, as visited by the truck k. Based on the symmetry of the problem, we might introduce a constraint that guarantees that a tour $\pi^k = \{0, i \ldots, j, n+1\}$ (here, $x^k_{0,i} = x^k_{j,n+1} = 1$) and its partially inverted sequence $\pi^{Tk} = \{0, j \ldots, i, n+1\}$ are not both evaluated. To this end, we introduce the following set of *static symmetry breaking inequalities* (Jünger et al., 2010) that guarantee that for any sequence $\pi^k = \{0, i, \ldots, j, n+1\}, i \leq j$, the partially inverted sequence $\pi^{Tk} = \{0, j, \ldots, i, n+1\}$ must not be evaluated.

$$\sum_{i \in V_N} i \cdot x^k_{0,i} \leq \sum_{j \in V_N} j \cdot x^k_{j,n+1} \quad \forall k \in \mathcal{K}. \tag{3.34}$$

This completes the proof of Proposition 3.3. ■

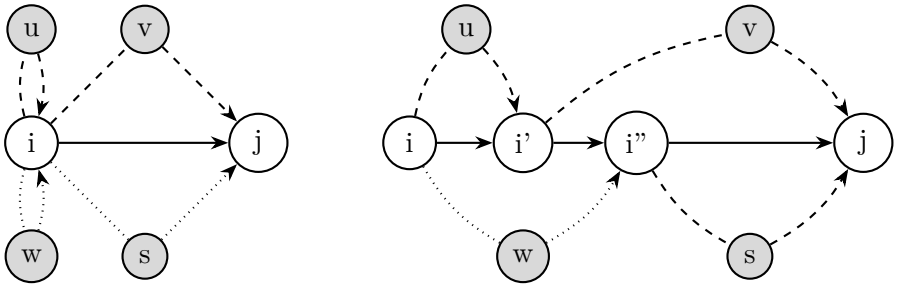

Figure 3.3.2: A segment of a VRPD route as a directed graph with two drones indicated by dashed and dotted lines, respectively, and the corresponding directed acyclic graph. In the corresponding graph, we assume that the weight of solid edges that connect dummy vertices is equal to 0. In this case, the order of the dummy vertices implies that the drone that serves u returns before the drone that serves w.

3.3.2.3 Knapsack Inequalities

In this section, we provide two sets of knapsack inequalities.

Proposition 3.4. Assume that $i \in V_L$, $j \in V_R$, and $w \in V_D$, where $|\{i, w, j\}| = 3$; then, the following inequalities are valid for the MILP (3.1)–(3.27):

$$x_{iw}^k + \sum_{d \in \mathcal{D}} \left(y_{iwi}^{kd} + \sum_{\substack{j \in V_R, \\ i \neq j, \\ w \neq j}} y_{iwj}^{kd} \right) \leq 1 \quad \forall k \in \mathcal{K}, i \in V_L, w \in V_D, i \neq w, \tag{3.35}$$

$$x_{iw}^k + x_{wj}^k + x_{ij}^k + \sum_{d \in \mathcal{D}} (y_{iwj}^{kd} + y_{iwi}^{kd} + y_{jwj}^{kd}) \leq 2 \quad \forall k \in \mathcal{K}, i \in V_L, w \in V_D, j \in V_R, |\{i, w, j\}| = 3. \tag{3.36}$$

Proof. Consider a pair of two vertices $i \in V_L$ and $w \in V_D$ where $i \neq w$. Constraints (3.6) guarantee that the indegree of performed deliveries is at most one. Therefore, w can be served from i at most once by either a truck, a drone that performs a cyclic delivery, or a drone that performs an acyclic delivery. Mathematically, this can be expressed through the following inequalities:

$$x_{iw}^k + \sum_{d \in \mathcal{D}} \left(y_{iwi}^{kd} + \sum_{\substack{j \in V_R, \\ i \neq j, \\ w \neq j}} y_{iwj}^{kd} \right) \leq 1 \quad \forall k \in \mathcal{K}, i \in V_L, w \in V_D, i \neq w. \tag{3.37}$$

Next, consider a triple of vertices $i \in V_L$, $w \in V_N$, and $j \in V_R$, where all vertices are pairwise different ($|\{i, w, j\}| = 3$). In this triple, with the truck (and possibly drones) located at i, there can be at most two deliveries such that all demands are fulfilled.

Starting at i, either the truck first serves w and then j, or the truck directly moves from i to j. In the latter case, a drone is permitted to either perform a cyclic or acyclic operation. Therefore, we can introduce the following inequalities:

$$x^k_{iw}+x^k_{wj}+x^k_{ij}+\sum_{d\in\mathcal{D}}(y^{kd}_{iwj}+y^{kd}_{iwi}+y^{kd}_{jwj}) \leq 2 \quad \forall k \in \mathcal{K}, i \in V_L, w \in V_D, j \in V_R, |\{i,w,j\}| = 3. \tag{3.38}$$

The inequalities (3.37) and (3.38) complete the proof of Proposition (3.4). ■

3.4 Matheuristic

As we will see in Section 3.5.1, a state-of-the-art MILP solver can solve only small VRPD instances within reasonable runtime. As an alternative, we might use a heuristic solution method which must be able to provide high-quality solutions within short computation time. In this context, a heuristic might approach the VRPD by solving the following hierarchical subproblems in succession:

1. *Allocation Problem*: Which customers should be served by which tandem (where each tandem consists of a truck and its drone fleet)?
2. *Sequencing Problem*: In which order should each allocation be processed?
3. *Assignment Problem*: In each sequence, which customers should be assigned to the truck and drones?
4. *Scheduling Problem*: Given the assignment, which feasible operations should the drones perform, i. e., what are the concrete locations where drones are launched and retrieved?

Methods that embed mathematical programming inside a heuristic framework are called *matheuristics* and have been applied to several routing problems (see, e. g., (Archetti and Speranza, 2014) and references therein). We propose a matheuristic in which we decompose the problem into two natural components as follows:

1. *Allocation & Sequencing*, that can be solved simultaneously through established heuristics for VRPs (see, e. g., (Toth and Vigo, 2014) and references therein). In the following, we refer to this subproblem as the *routing* subproblem of the VRPD.
2. *Drone assignment & Scheduling*, that are solved simultaneously through the use of mathematical programming techniques. In the following, we refer to this subproblem as the DASP. In contrast to existing works (e. g., (Boysen et al., 2018b; Es Yurek

and Ozmutlu, 2018)), the assignment of drones to customers is not fixed in advance or determined during routing, i. e., we solve the assignment and scheduling problem simultaneously. Furthermore, our method is suitable for cases where multiple drones are present that might have different relative velocities or endurances.

Algorithm 3.1 shows the basic structure of our proposed matheuristic which will be explained in more detail in the following sections. Initially, the incumbent objective value $f(x^*)$ is set to a sufficiently large value M. Then, our algorithm iterates until a termination criterion (e. g., a runtime limit) is met. In this algorithm, lines 3 to 8 refer to the allocation and sequencing problem which will be addressed in Section 3.4.1. Afterwards, in Section 3.4.2, we dedicate ourselves to lines 10 to 17 that are concerned with the drone assignment and scheduling problem.

Algorithm 3.1 An overview of the matheuristic framework.

```
1:  f(x*) ← M
2:  while termination critera is not met do
3:      Routes ← RandomizedSavingsHeuristic
4:      Routes ← LocalSearch(Routes)
5:      Routes ← sort(Routes)                                  } Allocation and
6:      if tabuList.contains(π : Routes) then                  } Sequencing (Section 3.4.1)
7:          continue while
8:      end if
9:      RUB ← 0
10:     for (π : Routes) do                                    }
11:         x_π ← DASP(π, BestObjStop(RUB), Cutoff(f(x*)))     }
12:         RUB ← (f(x_π) > RUB) ? f(x_π) : RUB                }
13:         if RUB ≥ f(x*) then                                } DASP (Section 3.4.2)
14:             tabuList.add(π)                                }
15:             continue while                                 }
16:         end if                                             }
17:     end for                                                }
18:     f(x*) ← RUB
19:     tabuList.add(π that determined RUB)
20: end while
```

3.4.1 Allocation and Sequencing

In each iteration, a set of routes is generated and sorted according to their length. As we use an exact procedure to determine the optimal drone assignment and schedule for a given route (refer to Section 3.4.2), a tabu list is employed. The tabu list stores examined

routes and inhibits the matheuristic from repeatedly solving the same DASP for routes that are guaranteed not to improve the incumbent solution. The criterion, under that a route is added to the list, will be described in more detail in Section 3.4.2.

In principle, any algorithm that generates a feasible routing (a set of routes) might be used as an initialization procedure within our heuristic. Moreover, it is conceivable to embed the DASP as a local search method within a more sophisticated metaheuristic approach. However, to show the strength of the DASP as a component, we choose a more simplistic procedure for solving the routing problem. More precisely, we adapt the parallel Clarke-Wright-Savings heuristic as initialization procedure (Clarke and Wright, 1964). For customizing this heuristic for the VRPD, we make two minor adjustments:

- Since the trucks do not possess a limited capacity in the VRPD, we limit each route to contain at most $\lceil |\mathcal{G}|/|\mathcal{K}| \rceil$ customers.
- In order to generate different starting solutions, within the heuristic the savings are weighted with a *random* value drawn at uniform from an interval $[r_{\text{lb}}, 1]$, where $0 < r_{\text{lb}} \leq 1$.

Afterwards, the initial routing is improved through some local search operations. Here, for a limited number of iterations, we attempt to improve the routing by minimizing the length of the longest remaining route. As local search method, we use the *2-opt* as single-route improvement heuristic (Lin and Kernighan, 1973). Furthermore, we rely on *String Relocation* (SR) and *String Exchange* (SE) with *best insertion*, respectively, as multi-route improvement heuristics (Toth and Vigo, 2014). All operators work by choosing *random* vertices within the routes. In the case of 2-opt, two vertices from the same route are selected and the sequence is inverted. If this inversion leads to an improved route length, the move is accepted, otherwise it is reverted. This move can be efficiently evaluated in $\mathcal{O}(1)$ per iteration. In the case of SR and SE, first, two random routes are selected from the set of routes. Clearly, these moves are only applied when the number of routes (trucks) is larger than one. In the case of SR, a random vertex is selected from the longer route and inserted into the shorter route at the best location in the sequence, i. e., the one that will add the minimal length to the route after insertion. In the case of SE, a random vertex is selected from each route and inserted into the respective other route at the best location in the sequence. Both moves, SR and SE, can be efficiently evaluated in $\mathcal{O}(n)$.

3.4.2 Drone Assignment and Scheduling

Once a feasible routing (a set of routes) has been generated, the next steps involve solving the DASP for each route. More precisely, we use a MILP solver to determine the optimal

assignment and schedule of drones for each truck. The proposed MILP formulations will be discussed in Sections 3.4.2.1 and 3.4.2.2. Let the *Route Upper Bound* (RUB) be the supremum on the makespan based on all routes that have been evaluated thus far in the current iteration. The RUB is re-initialized to zero in every iteration. Then, for each route π in the list of routes, we solve the DASP as a MILP, yielding a solution x_π, by supplying the solver with the following parameters (line 11 in Algorithm 3.1):

- *BestObjStop*(RUB): When the solver detects a feasible solution x_π with an objective value $f(x_\pi) \leq$ RUB, we can terminate the solver's internal branch-and-cut process. In this case, the objective value associated with the DASP for the current route π is at least as good as the RUB. Therefore, there is no need to solve the current DASP to optimality as the RUB remains unchanged. In particular, in presence of multiple trucks, once an initial value on the RUB $\neq 0$ has been established, by solving the DASP for the first truck's route, subsequent DASPs can often be evaluated more quickly through this process.
- *Cutoff*($f(x^*)$): When the solver detects that the best possible solution value is greater than or equal to the incumbent objective value $f(x^*)$, we can cut off (terminate) the solver's branch-and-cut process. In this case, the objective value $f(x_\pi)$ would generate a new RUB that is guaranteed to be greater than or equal to the current incumbent objective value $f(x^*)$. Therefore, the makespan associated with the current set of routes is guaranteed not to improve on the incumbent solution.

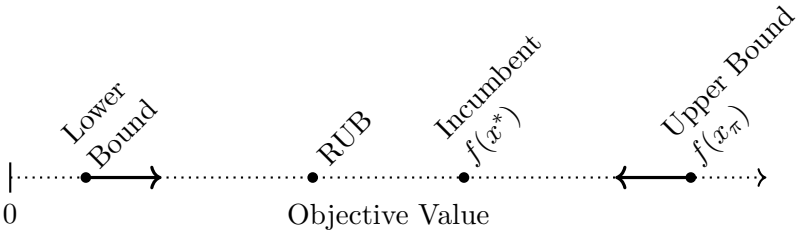

Figure 3.4.1: For each route π, internally, the MILP solver attempts to find an optimal solution x_π to the DASP by closing the gap between the lower and upper bound. If the lower bound is greater than or equal to the incumbent solution $f(x^*)$, a cutoff can be applied. In this case, the next RUB is guaranteed to be greater than or equal to $f(x^*)$ and, thus, the incumbent solution will not change. Similarly, if the upper bound $f(x_\pi) \leq$ RUB, we can stop and do not need to solve the DASP to optimality. In this case, as the RUB determines the makespan of the VRPD solution, it is sufficient to prove that a solution x_π exists for the current route π such that $f(x_\pi) \leq$ RUB.

Figure 3.4.1 visualizes how this procedure can significantly speed up the process, requiring not all *Mixed-Integer Programming* (MIP) gaps to be tight when solving the DASP.

Whenever the MILP solver returns a solution x_π with an objective $f(x_\pi) >$ RUB, we set the RUB to the value of $f(x_\pi)$ (line 12 in Algorithm 3.1). Furthermore, if the current RUB is greater than or equal to $f(x^*)$, the current iteration can terminate, i. e., no further routes have to be evaluated as no improvement on the incumbent solution can be found with the current set of routes. In this case, the route π that determined the RUB (and, in case of a symmetric graph, the reverse route as a result from Section 3.3.2.2) are stored on the tabu list and we continue with the next iteration by generating a completely new set of routes (lines 13 to 16 in Algorithm 3.1). Finally, if the algorithm evaluated the last route and if RUB $< f(x^*)$ still holds, then we have a new and improved incumbent solution x (that consists of all x_π for each route π) with an objective value $f(x) =$ RUB. This solution is then stored as the incumbent solution and the route that determined the RUB is also added to the tabu list (refer to Section 3.4.1).

At this point, we require a MILP that, given a feasible routing, returns the optimal assignment of customers to either the truck or a drone including the schedule of drones for each route π such that the makespan is minimized. In principle, MILP (3.1)–(3.27) can be readily used for this, by fixing the variables x_{ij}^k based on each route π. However, in terms of columns and rows, the resulting MILP is still very large and, given a fixed allocation and sequencing decision, contains many redundant constraints.

In what follows, we will show that a much more compact MILP formulation for the DASP can be derived. To this end, assume that a graph $\mathcal{G}(V, E)$ and a route (permutation) $\pi = (0, \ldots, n+1), \pi \subseteq V$ is given as an ordered list of vertices. Let $V_L = \pi \setminus \{n+1\}$, $V_R = \pi \setminus \{0\}$, and $V_N = V_L \cap V_R$. Furthermore, we call $V_D \subseteq V_N$ the *drone-serviceable* subset of customers. As in Section 3.3, we assume that t_{ij} (d_{ij}) and $\overline{t}_{ij}$ ($\overline{d}_{ij}$) are the times (distances) required to travel from $i \in V_L$ to $j \in \pi$ by truck and drone, respectively. We use the notation $i \prec j$ ($i \succ j$) to indicate that the index associated with i in the ordered list π is smaller (larger) than the index associated with j. Furthermore, $i \preceq j$ ($i \succeq j$) implies that the index of i is smaller (larger) than or equal to the index of j.

For the DASP, we will now provide two MILP formulations. Section 3.4.2.1 provides a formulation, where a drone operation (i, w, j) is directly modelled through three indices, similar to the decision variables that were already used in Section 3.3. Alternatively, Section 3.4.2.2 models each drone operation (i, w, j) through two disjoint arcs (i, w) and (w, j). The latter formulation reduces the number of variables required for modeling the problem at the cost of additional constraints.

3.4.2.1 Three-Index-Operation Formulation

In this section, we model a drone operation (i, w, j) directly through three indices. Compared to the two-index formulation presented in Section 3.4.2.2, this makes the number of variables larger but the model itself compact. For modeling the problem, we introduce the decision variables in Table 3.4.1.

Table 3.4.1: Decision variables used in the three-index DASP formulation.

y_w $\forall w \in V_N$	Binary variables that determine if $w \in V_N$ is served by any drone.
o_i^d $\forall d \in \mathcal{D}, i \in V_L$	Binary variables that indicate if drone $d \in \mathcal{D}$ is available at $i \in V_L$.
y_{iwj}^d $\forall d \in \mathcal{D}, i \in V_L, w \in V_D, j \in V_R,$ $i \prec w \prec j$	Binary variables that specify if a drone $d \in \mathcal{D}$ performs an acyclic operation (i, w, j).
y_{iwi}^d $\forall d \in \mathcal{D}, i \in V_L, w \in V_D, i \prec w$	Binary variables that determine if a drone $d \in \mathcal{D}$ performs a cyclic operation (i, w, i).
a_i $\forall i \in \pi$	Continuous variables with a lower bound of 0 that specify the earliest possible arrival time of the truck at i.
s_i $\forall i \in \pi$	Continuous variables with a lower bound of 0 that specify the earliest possible departure time of the truck from i.
r_i^d $\forall d \in \mathcal{D}, i \in \pi$	Continuous variables with a lower bound of 0 that specify the earliest possible release time of drone d at i.
s_i^d $\forall d \in \mathcal{D}, i \in \pi$	Continuous variables with a lower bound of 0 that specify the earliest possible time at which drone d is available for a departure from i.

Given $\mathcal{G}(V, E)$ and a fixed sequence of vertices π, MILP (3.39)–(3.54) is used to find the best possible assignment and schedule of drones such that the makespan at vertex $n+1$ is minimized.

$$\min \quad s_{n+1}, \tag{3.39}$$

$$\text{s.t.} \quad y_w = 0 \quad \forall w \in V_N \setminus V_D, \tag{3.40}$$

$$\sum_{d \in \mathcal{D}} \sum_{\substack{i \in V_L, \\ i \prec w}} (y_{iwi}^d + \sum_{\substack{j \in V_R, \\ w \prec j}} y_{iwj}^d) = y_w \quad \forall w \in V_D, \tag{3.41}$$

$$\sum_{d \in \mathcal{D}} \sum_{\substack{w \in V_D, \\ v \prec w}} (y_{vwv}^d + \sum_{\substack{j \in V_R, \\ w \prec j}} y_{vwj}^d) \leq M(1 - y_v) \quad \forall v \in V_N, \tag{3.42}$$

$$\sum_{d \in \mathcal{D}} \sum_{\substack{w \in V_D, \\ w \prec v}} \sum_{\substack{i \in V_L, \\ i \prec w}} y_{iwv}^d \leq M(1 - y_v) \quad \forall v \in V_N, \tag{3.43}$$

$$-M(y_i + y_j) + s_i + t_{ij} \leq a_j \quad i \in V_L, j \in V_R, i \prec j, \tag{3.44}$$

$$a_i \leq s_i \quad \forall i \in V, \tag{3.45}$$

$$-M(1 - y^d_{iwj}) + s^d_i + (\bar{t}_l + \bar{t}_{iw} + \bar{t}_{wj} + \bar{t}_r) \leq r^d_j \quad \forall d \in \mathcal{D}, i \in V_L, w \in V_D, j \in V_R, i \prec w \prec j, \tag{3.46}$$

$$r^d_i + \sum_{\substack{w \in V_D, \\ i \prec w}} (\bar{t}_l + \bar{t}_{iw} + \bar{t}_{wj} + \bar{t}_r) y^d_{iwi} \leq s^d_i \quad \forall d \in \mathcal{D}, i \in V_L, \tag{3.47}$$

$$s^d_i + \bar{t}_l \sum_{\substack{w \in V_D, \\ i \prec w}} \sum_{\substack{j \in V_R, \\ w \prec j}} y^d_{iwj} \leq s_i \quad \forall d \in \mathcal{D}, i \in V, \tag{3.48}$$

$$a_j + \bar{t}_r \sum_{\substack{i \in V_L, \\ i \prec w}} \sum_{\substack{w \in V_D, \\ w \prec j}} y^d_{iwj} \leq r^d_j \quad \forall d \in \mathcal{D}, j \in V, \tag{3.49}$$

$$(\bar{d}_{iw} + \bar{d}_{wj}) y^d_{iwj} \leq \mathcal{E} \quad \forall d \in \mathcal{D}, i \in V_L, w \in V_D, j \in V, i \prec w \prec j, \tag{3.50}$$

$$(\bar{d}_{iw} + \bar{d}_{wi}) y^d_{iwi} \leq \mathcal{E} \quad \forall d \in \mathcal{D}, i \in V_L, w \in V_D, i \prec w, \tag{3.51}$$

$$o^d_h + \sum_{\substack{i \in V_L, \\ i \prec h}} \sum_{\substack{w \in V_D, \\ i \prec w}} \sum_{\substack{j \in V_R, \\ j \succ h}} y^d_{iwj} = 1 \quad \forall d \in \mathcal{D}, h \in V_N, \tag{3.52}$$

$$\sum_{\substack{w \in V_D, \\ i \prec w}} \sum_{\substack{j \in V_R, \\ w \prec j}} y^d_{iwj} \leq o^d_i \quad \forall d \in \mathcal{D}, i \in V_L, \tag{3.53}$$

$$\sum_{\substack{w \in V_D, \\ i \prec w}} y^d_{iwi} \leq M o^d_i \quad \forall d \in \mathcal{D}, i \in V_L. \tag{3.54}$$

The objective function (3.39) minimizes the time required such that all customers are served and the truck and drones have returned to the depot. The set of constraints (3.40)–(3.41) force the variables y_w to 1 if a customer w is served by any drone and to value 0 for any customer that cannot be served by drone. Furthermore, they guarantee that each customer is served at most once by a drone. Constraints (3.42) and (3.43) impose that no drone sortie can start or end at a customer v that is served by a drone. Lower bounds on the arrival time of the truck are specified through the set of constraints (3.44). Moreover, due to constraints (3.45) the departure time of the truck is bound by its arrival time at each vertex. For each drone operation (i, w, j), constraints (3.46) provide valid lower bounds on the release times. Constraints (3.47) ensure that a drone can only depart from a vertex as soon as all cyclic operations have been completed. The earliest possible departure time of each drone qualifies as a lower bound on the earliest possible departure time of the truck (constraints (3.48)). Furthermore, constraints (3.49) enforce that a drone can only be released after the truck has arrived. Constraints (3.50) and (3.51) ensure that

no drone operation violates the maximum endurance. In practice, this can be efficiently handled during a preprocessing step such that the associated variables y^d_{iwj} are bound to 0 if the operation violates the endurance. Constraints (3.52) determine the values on the variables o^d_h, that indicate if a drone is available for an operation at each vertex h. If there is an operation (i, w, j) that occurs circumjacent to h, then the drone is unavailable at h. Finally, constraints (3.53) and (3.54) ensure that at each vertex, a drone can only perform an acyclic and cyclic operation, respectively, if the drone is available.

For completeness, in principle, it is also possible to introduce some additional auxiliary binary variables x_{ij} where $i \in V_L, j \in V_R, i \prec j$, that explicitly determine the path of the truck. This might be done by including the following inequalities (3.55)–(3.57) as constraints in the model and by replacing (3.44) with (3.58).

$$x_{ij} \leq (1 - y_i) \quad \forall i \in V_L, j \in V_R, i \prec j, \tag{3.55}$$

$$x_{ij} \leq (1 - y_j) \quad \forall i \in V_L, j \in V_R, i \prec j, \tag{3.56}$$

$$(1 - y_i) + (1 - y_j) - 1 - \sum_{\substack{w \in V_N, \\ i \prec w, \\ w \prec j}} (1 - y_w) \leq x_{ij} \quad \forall i \in V_L, j \in V_R, i \prec j, \tag{3.57}$$

$$s_i + t_{ij} x_{ij} \leq a_j \quad \forall d \in D, i \in V_L, j \in V, i \prec j. \tag{3.58}$$

Here, constraints (3.55) and (3.56) ensure that the truck departs or arrives at a customer that is served by drone. Furthermore, constraints (3.57) guarantees that x_{ij} is equal to 1 if neither i nor j are served by drone and all remaining customers w in between i and j are not served by the truck. Adding these constraints would remove the M in the set of constraints (3.44) and allow for the inclusion of VIEQs (3.59) and (3.60). These VIEQs are analogous to the previously introduced VIEQs (3.30) and (3.32) and have proven themselves to be very beneficial in solving MILP (3.1)–(3.27) more efficiently.

$$\sum_{i \in V} (s_i - a_i) + \sum_{i \in V_L} \sum_{\substack{j \in V_R, \\ i \prec j}} t_{ij} x_{ij} \leq s_{n+1}, \tag{3.59}$$

$$\sum_{i \in V_L} \sum_{\substack{w \in V_D, \\ i \prec w}} ((\bar{t}_{iw} + \bar{t}_{wi}) y^d_{iwi} + \sum_{\substack{j \in V_R, \\ w \prec j}} (\bar{t}_{iw} + \bar{t}_{wj}) y^d_{iwj}) \leq s_{n+1} \quad \forall d \in \mathcal{D}. \tag{3.60}$$

However, we do not use these additional variables and constraints when solving the DASP.

3.4.2.2 Two-Index-Operation Formulation

Another possibility is to represent an operation (i, w, j) through two disjoint arcs (i, w) and (w, j), instead. This allows for a significantly reduced number of variables at the cost of several additional constraints that guarantee valid drone operations. For this formulation, consistent with the three-index formulation, we use the variables $y_w, o_i^d, a_i, s_i, r_i^d$, and s_i^d (refer to Section 3.4.2.1). Furthermore, we introduce the decision variables in Table 3.4.2.

Table 3.4.2: Decision variables used in the two-index DASP formulation.

y_{iw}^d $\forall d\in\mathcal{D}, i\in V_L, w\in V_D, i\prec w$	Binary variables that specify if a drone $d \in \mathcal{D}$ performs an acyclic *delivery* to $w \in V_D$ which starts at $i \in V_L$.
$\tilde{y}_{wj}^d$ $\forall d\in\mathcal{D}, w\in V_D, j\in V_R, w\prec j$	Binary variables that specify if a drone $d \in \mathcal{D}$ is *retrieved* at $j \in V_R$ after an acyclic delivery to $w \in V_D$.
z_{iw}^d $\forall d\in\mathcal{D}, i\in V_L, w\in V_D, i\prec w$	Binary variables that determine if a drone $d \in \mathcal{D}$ performs a cyclic delivery (i, w, i).

The model is then given as follows, where (3.61) marks the objective and the constraints are given by (3.62)–(3.78).

$$\min \quad s_{n+1} \tag{3.61}$$

$$\text{s.t.} \quad y_w = 0 \quad w \in V_N \setminus V_D, \tag{3.62}$$

$$\sum_{k\in D} \sum_{\substack{i\in V_L \\ i\prec w}} (y_{iw}^d + z_{iw}^d) = y_w \quad w \in V_D, \tag{3.63}$$

$$\sum_{d\in\mathcal{D}} \sum_{\substack{v\in V_D, \\ w\prec v}} (y_{wv}^d + z_{wv}^d) \leq M(1 - y_w) \quad \forall w \in V_N, \tag{3.64}$$

$$\sum_{d\in\mathcal{D}} \sum_{\substack{v\in V_D, \\ v\prec w}} (\tilde{y}_{vw}^d + z_{vw}^d) \leq M(1 - y_w) \quad \forall w \in V_N, \tag{3.65}$$

$$\sum_{\substack{i\in V_L, \\ i\prec w}} y_{iw}^d = \sum_{\substack{j\in V_R, \\ w\prec j}} \tilde{y}_{wj}^d \quad \forall d \in \mathcal{D}, w \in V_D, \tag{3.66}$$

$$-M(y_i + y_j) + s_i + t_{ij} \leq a_j \quad \forall i \in V_L, j \in V, i \prec j, \tag{3.67}$$

$$a_i \leq s_i \quad \forall i \in V, \tag{3.68}$$

$$-M(1 - y_{iw}^d) + s_i^d + \bar{t}_l + \bar{t}_{iw} \leq r_w^d \quad \forall d \in \mathcal{D}, i \in V_L, w \in V_D, i \prec w, \tag{3.69}$$

$$-M(1 - \tilde{y}_{wj}^d) + s_w^d + \bar{t}_{wj} + \bar{t}_r \leq r_j^d \quad \forall d \in \mathcal{D}, w \in V_D, j \in V_R, i \prec w, \tag{3.70}$$

$$r_i^d + \sum_{\substack{w \in V_D, \\ i \prec w}} (\bar{t}_l + \bar{t}_{iw} + \bar{t}_{wi} + \bar{t}_r) z_{iw}^d \leq s_i^d \quad \forall d \in \mathcal{D}, i \in V_L, \tag{3.71}$$

$$s_i^d + \bar{t}_l \sum_{\substack{w \in V_D, \\ i \prec w}} y_{iw}^d \leq s_i \quad \forall d \in \mathcal{D}, i \in V, \tag{3.72}$$

$$a_j + \bar{t}_r \sum_{\substack{w \in V_D, \\ w \prec j}} \tilde{y}_{wj}^d \leq r_j^d \quad \forall d \in \mathcal{D}, j \in V, \tag{3.73}$$

$$\sum_{\substack{i \in V_L, \\ i \prec w}} \bar{d}_{iw} y_{iw}^d + \sum_{\substack{j \in V_R, \\ w \prec j}} \bar{d}_{wj} \tilde{y}_{wj}^d \leq \mathcal{E} \quad \forall d \in \mathcal{D}, w \in V_D, \tag{3.74}$$

$$\sum_{\substack{i \in V_L, \\ i \prec w}} (\bar{d}_{iw} + \bar{d}_{wi}) z_{iw}^k \leq \mathcal{E} \quad \forall d \in \mathcal{D}, w \in V_D, \tag{3.75}$$

$$o_h^d + \sum_{\substack{i \in V_L, \\ i \prec h}} \sum_{\substack{w \in V_D, \\ w \succ i}} y_{iw}^d - \sum_{\substack{j \in V_R, \\ j \preceq h}} \sum_{\substack{w \in V_D, \\ w \prec j}} \tilde{y}_{wj}^d = 1 \quad \forall d \in \mathcal{D}, h \in V_N, \tag{3.76}$$

$$\sum_{w \in V_D} y_{iw}^d \leq o_w^d \quad \forall d \in \mathcal{D}, i \in V_L, \tag{3.77}$$

$$\sum_{w \in V_D} z_{iw}^d \leq M o_w^d \quad \forall d \in \mathcal{D}, i \in V_L. \tag{3.78}$$

This model closely follows the one presented in Section 3.4.2.1. The objective function is given by (3.61) and minimizes the arrival time at the depot. Constraints (3.62) force the variables y_w to value 0 for any customer that cannot be served by drone. Furthermore, the variables y_w are forced to value 1 by constraints (3.63) if a drone-eligible customer w is served by any drone. These constraints also guarantee that there can be at most one delivery by drone to each customer. Constraints (3.64) and (3.65) ensure that no drone starts or ends a delivery at a customer w that is served by any drone. For each delivery that does not start and end at the same location, there must be exactly one incoming and one outgoing arc, which are synchronized through constraints (3.66).

The set of constraints (3.67) provides lower bounds on the arrival time of the truck. For acyclic drone operations, constraints (3.69) and (3.70) account for the release times of drones during departure and return flight, respectively. Furthermore, the set of constraints (3.71) provide a lower bound on the departure time of the drone at each vertex in case of cyclic operations. The departure time of the truck is bound by its arrival time through constraints (3.68) and the departure time of drones (constraints (3.72)). Furthermore, constraints (3.73) guarantee that at each vertex where a drone is retrieved, the drone can only be released after its respective truck has arrived. The endurance check is implemented through the set of constraints (3.74) and (3.75). Constraints (3.76) determine the value

on the variables o_h^d, that indicate if a drone k is available for an operation at vertex h. In particular, if a drone k performs a circumjacent operation at h, then o_h^d is forced to 0. Finally, constraints (3.77) and (3.78) guarantee that a drone can only perform acyclic and cyclic operations, respectively, if it is available.

3.4.3 Discussion and Scalability

As outlined in Section 3.4, because of the min-max objective of the VRPD, a significant speed up can be achieved in our matheuristic framework through a cutoff and the subsequent prevention of unnecessary evaluations of either MILP. This cutoff can be applied, whenever the MILP yields a lower bound that is greater than or equal to the incumbent solution $f(x^*)$ and by an early termination of the solver whenever the upper bound moves beneath the current RUB (see Figure 3.4.1).

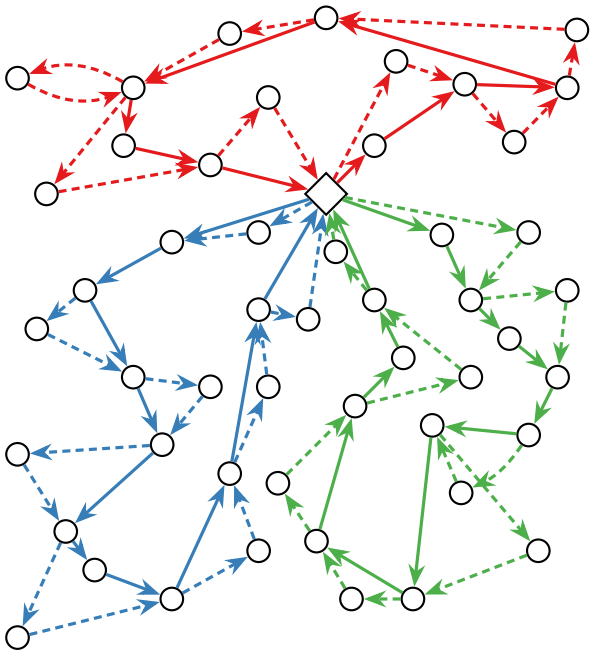

Figure 3.4.2: A sample solution for an instance with 50 customers found through the matheuristic with $|\mathcal{K}| = 3, |\mathcal{D}| = 1$.

Both, MILP (3.39)–(3.54) and MILP (3.61)–(3.78), can be solved relatively quickly for reasonably sized routes. Figure 3.4.2 shows a sample solution found through the matheuristic (Algorithm 3.1) within just a few seconds. As we have outlined, the number of variables required for modeling the DASP is $\mathcal{O}(|\pi|^3)$ and $\mathcal{O}(|\pi|^2)$ for the three- and two-index-Operation formulation, respectively. For routes that contain a small amount of customers, the performance difference is negligible. Nevertheless, the reduced number of

constraints involved in the formulation presented in Section 3.4.2.1 and the large number of variables that can be removed during preprocessing often make it the preferred formulation for the instances considered in this work. Overall, the runtime is heavily influenced by the set of binary variables that determine the drone operations. In particular, the value of $\mathcal{E}$ can severely limit the number of feasible drone operations and therefore significantly reduce the model size. This can be effectively exploited by the three-index formulation during preprocessing.

When solving the DASP for larger routes with up to 100 customers, it might take the solver several seconds to resolve the MILP formulation. At the cost of an optimal drone schedule, an improved runtime behavior can be achieved in several ways:

- It is easily possible to restrict the set of feasible drone operations further, e. g., by limiting the maximum number of vertices that can be visited by the truck between the launch and recovery of a drone (similar approaches were used by Bouman et al. (2018a) and Boysen et al. (2018b)).
- Furthermore, at the expense of potentially suboptimal drone placements, a time limit or relative MIP gap might be imposed on the MILP solver when solving the DASP.

As an alternative approach, within the matheuristic, we suggest splitting the route (permutation) π into m overlapping partitions $\pi_1|\ldots|\pi_m$. As an example, consider the route $\pi = (0, \ldots, n/2, \ldots, n+1)$ where n is an even number in $\mathbb{Z}^+$. If n is very large, instead of solving the DASP on π, we might consider two (or m) overlapping partitions $\pi_1 = (0, \ldots, n/2)$ and $\pi_2 = (n/2, \ldots, n+1)$ such that the makespan $s_{n/2}$ determined by solving the DASP on π_1 marks the starting time of the DASP in π_2. This might be achieved by replacing line 11 in Algorithm 3.1 with Algorithm 3.2. In this algorithm, the function Partition(π, m) generates m overlapping partitions of π, $f(x_\pi)$ is the accumulated cost associated with the route after evaluating each partition, and $f(x_{\pi_i})$ is the cost of solving the DASP on each partition π_i. Preliminary experiments have shown that this approach is well-suited for large routes where the runtime behavior is improved significantly with little or no loss on the solution quality. Figure 3.4.3 shows a sample application of the DASP involving many partitions.

3.5 Computational Experiments and their Numerical Results

We carried out two main classes of computational experiments, on small- and large-scale instances and we present the numerical results in this section. More precisely, in Section 3.5.1 we discuss the performance of Gurobi Optimizer in solving MILP (3.1)–(3.27).

Algorithm 3.2 DASP partition routine.

$f(x_\pi) = 0$
for $\pi_i \in \text{Partition}(\pi, m)$ **do**
  if $i \neq m$ **then**
    $x_{\pi_i} \leftarrow \text{DASP}(\pi_i, \text{BestObjStop}(0), \text{Cutoff}(f(x^*) - f(\pi)))$
  else
    $x_{\pi_i} \leftarrow \text{DASP}(\pi_i, \text{BestObjStop}(\text{RUB} - f(\pi)), \text{Cutoff}(f(x^*) - f(\pi)))$
  end if
  $f(x_\pi) \leftarrow f(x_\pi) + f(x_{\pi_i})$
end for

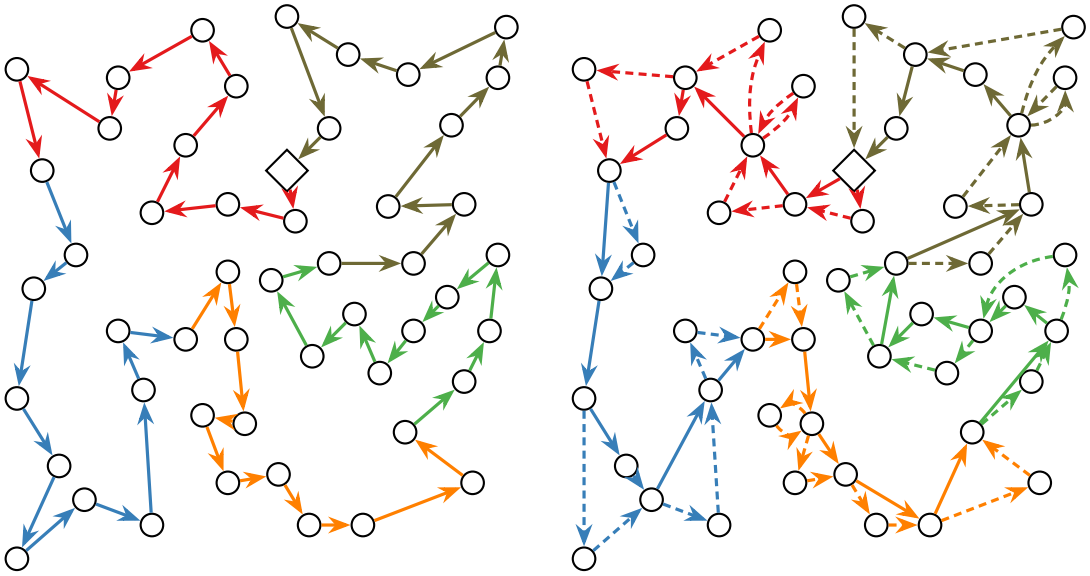

Figure 3.4.3: A sample solution that was found through the Clarke-Wright savings heuristic and improved through local search for a single truck (left-hand side) and the respective solution after solving the DASP ($|\mathcal{D}| = 1$) on each partition, as indicated by the different colors (right-hand side).

Moreover, we highlight the benefits of adding the VIEQs with bounds on the makespan (3.30) and (3.32), the symmetry breaking constraints (3.34), and (knapsack) inequalities (3.37) and (3.38) to the model. Finally, we compare the performance of the matheuristic in solving the same instances through the three-index formulation of the DASP (see Section 3.4.2.1) and two-index formulation of the DASP (see Section 3.4.2.2). Afterwards, Section 3.5.2 shows that the proposed matheuristic can also be used to solve large-scale instances effectively.

For the matheuristic, the set of constraints (3.46) for the three-index formulation and (3.69)–(3.70) for the two-index formulation were treated as lazy constraints. All experiments were performed on Intel® Xeon® Gold 6126 CPUs and 16 GB of RAM. We imple-

mented our algorithms in Java SE 8 and for solving MILPs we used Gurobi Optimizer 8.1.0 (Gurobi Optimization, LLC, 2023).

3.5.1 Small Instances

In this section, we describe the computational experiments on small instances and provide their numerical results. Since there is no commonly accepted benchmark for the VRPD, we perform our experiments on the benchmark instances that were used by Agatz et al. (2018) for the TSPD and Murray and Chu (2015) for the FSTSP. In Section 3.5.1.1 we test the MILP and matheuristic on the instances used by Agatz et al. (2018). Furthermore, in Section 3.5.1.2, we evaluate our model and heuristic on the instances used by Murray and Chu (2015). On the latter instances, to respect the problem description of Murray and Chu (2015) and to produce comparable results, several minor adjustments to the VRPD and DASP MILPs are necessary that we explain in detail in Appendix 3.B.

3.5.1.1 Small Instances used by Agatz et al. (2018)

In this section, we provide the results that we obtained on the instances used by Agatz et al. (2018). These instances are specified by a symmetric graph $\mathcal{G}(V, E)$ and assume that the truck and drone follow the Euclidean distance metric. Furthermore, the instances specify two parameters: the relative velocity α and the endurance $\mathcal{E}$. In particular, through a relative endurance parameter $\mathcal{E}_r \in \mathbb{R}^+$, the endurance can be calculated as follows, where $\max(E)$ refers to the edge with maximum weight in the graph $\mathcal{G}$:

$$\mathcal{E} \coloneqq \mathcal{E}_r \cdot \max(E) \quad \mathcal{E}_r \in [0, 2]. \tag{3.79}$$

In this case, $\mathcal{E}_r = 0$ guarantees that no drone operation is possible while $\mathcal{E}_r = 2$ marks any operation as feasible. The velocity of the drones $\overline{v}$ is assumed to be $\alpha \in \mathbb{R}^+$ times the velocity of the truck. Furthermore, the times required to launch and retrieve drones, i. e., $\overline{t}_l$ and $\overline{t}_r$, are assumed to be negligible. Agatz et al. (2018) provide 10 instances (with 10 vertices each), where each instance is associated with a value $\alpha \in \{1, 2, 3\}$ and a value $\mathcal{E}_r \in \{0.2, 0.4, 0.6, 1.0, 1.5, 2.0\}$ yielding a total of 180 different instances. By definition of Poikonen et al. (2017) and X. Wang et al. (2017), the number of trucks and drones per truck are a parameter in the VRPD. Thus, as a meaningful selection on for small instances, we limit ourselves to $|\mathcal{K}| \in \{1, 2\}$ and $|\mathcal{D}| \in \{1, 2\}$, i. e., we either have one or two trucks that are equipped with either one or two drones each. Therefore, we consider a total of $2 \cdot 2 \cdot 180 = 720$ instances for our experiments. We solve these 720 instances in five different ways as follows:

1. We solve the MILP (3.1)–(3.27) without VIEQs.
2. We further include the VIEQs (3.30)–(3.32) that provide bounds on the makespan to the model.
3. Additionally, we include the symmetry breaking constraints (3.34) and (knapsack) inequalities (3.37)–(3.38) to previous model and VIEQs.
4. Finally, we consider the matheuristic with either the three-index formulation MILP (3.39)–(3.54) or the two-index formulation MILP (3.61)–(3.78) for solving the DASP.

When solving the VRPD as MILPs, we limit the runtime of the solver to 15 minutes. For solving the VRPD through the matheuristic, we limit the runtime of the heuristic to 10 minutes. Furthermore, within this time limit, if the incumbent value remains unchanged for more than 1 minute, the heuristic terminates. The parameter r_{lb} that randomizes the savings during initialization was set to 0.7 and the number of iterations spent on local search were set to $\lceil |V|^{1.5} \rceil$, where $|V|$ marks the number of vertices in the instance. Finally, we perform five runs of the matheuristic per instance.

In total, this yields $3 \cdot 720 = 2160$ unique experiments using the MILP and the VIEQ and $5 \cdot 720 = 3600$ unique experiments using the matheuristic with each MILP for solving the DASP. Therefore, we have done $2160 + 2 \cdot 3600 = 9360$ experiments in total.

For the analysis of our results, we introduce the following metric. More precisely, given a fixed number of trucks, Δ indicates the relative change in makespan through the introduction of drones and is calculated as follows:

$$\Delta := 100\% - \frac{\text{objective value}}{\text{optimal objective value with } \mathcal{D} = \{\}} \tag{3.80}$$

Tables 3.D.1–3.D.3 (see Appendix 3.D) show the average solutions achieved by solving the instances through steps 1.–3. (see above). Each table shows the average value of Δ, the average MIP gap returned by the solver, the average runtime t, the average number of acyclic (#a) and cyclic operations ($\#c$) in a solution, and the number of runs where an optimal solution was found within the runtime limit.

Furthermore, Tables 3.D.4 and 3.D.5 (see Appendix 3.D) show the average solution achieved through the matheuristic using the three- and two-index DASP formulation, respectively. In this case, we report the average value of Δ, the average runtime t_{best} after which the incumbent solution was found, the average number of changes in the incumbent solution until convergence n, the average number of acyclic and cyclic operations, respectively, and the share of solutions where the heuristic found a solution of equal or better quality than the one returned by the solver.

Overall, we can make the following observations. Regarding the MILP in Tables 3.D.1–3.D.3, on average, the introduction of the VIEQ significantly improved the performance of the MILP solver. In particular, we can observe a larger share of instances that are solved to optimality and significantly reduced runtimes (and MIP gaps) while doing so. The average time to optimality is significantly influenced by the number of drones and the relative endurance $\mathcal{E}_r$ which by implication have a large influence on the number of binary variables y_{iwj}^{kd} that might be removed during preprocessing. Furthermore, $\mathcal{E}_r$ has a strong influence on the quality of the root relaxation. This is highlighted in Table 3.5.1, which shows the average objective, iterations, and time (in seconds) after solving the root relaxation. In this table, the column labeled $\%_R$ is calculated as follows:

$$\%_R \coloneqq \frac{\text{root relaxation objective}}{\text{objective value of the feasible solution after runtime}} \tag{3.81}$$

Table 3.5.1: Results obtained after solving the root relaxation depending on the relative endurance $\mathcal{E}_r$ (averaged over all instances).

	MILP (3.1)–(3.27) without VIEQ				MILP + makespan bounds				MILP + all VIEQ			
$\mathcal{E}_r$	Objective	Iterations	Time	$\%_R$	Obj.	Iter.	Time	$\%_R$	Obj.	Iter.	Time	$\%_R$
0.2	0	281.1	0.0	0.0%	163.0	399.7	0.0	58.0%	163.0	388.1	0.0	58.0%
0.4	0	334.5	0.0	0.0%	131.8	579.2	0.1	47.5%	131.8	536.6	0.1	47.5%
0.6	0	438.4	0.2	0.0%	82.5	892.5	0.3	31.0%	82.5	825.0	0.3	31.0%
1.0	0	502.2	0.8	0.0%	65.4	1,017.0	1.5	28.1%	65.4	1,016.1	1.3	28.3%
1.5	0	588.4	1.4	0.0%	62.3	1,081.1	2.8	30.6%	62.3	1,167.1	2.3	30.7%
2.0	0	227.5	1.2	0.0%	62.3	1,348.1	3.6	31.7%	62.3	1,454.8	3.4	32.0%
Average	0	395.3	0.6	0.0%	94.6	886.3	1.4	37.8%	94.6	898.0	1.2	37.9%

Regarding the matheuristic in Tables 3.D.4 and 3.D.5, on average, optimal or near-optimal solutions are found in almost all cases. On small instances, there is no significant difference in solving the DASP through either the three- or two-index formulation with regard to the time required to solve the MILP. In particular, the matheuristic converges very fast and does not require many changes to the incumbent solution to find optimal solutions. Solving the DASP through mathematical programming provides an intensive local search mechanism that is also very robust. In some cases, the optimal VRPD solution can be constructed by solving the DASP on structurally similar but not necessarily equal routes (i. e., distinct routes that share a large number of the same arcs).

In almost all cases, the matheuristic is able to provide a solution that is at least as good (or better) than the one provided by the MILP solver. In order to highlight this behavior, Tables 3.D.4 and 3.D.5 contain a column that shows the number of times where a solution of equal or better quality was found by the matheuristic.

In what follows, we will provide a more detailed analysis of solutions based on Table 3.D.3 (MILP with all VIEQs active) which performed the best on average. To this end, Figure 3.5.1 shows the average relative improvement Δ as a function of the relative endurance $\mathcal{E}_r$, the relative velocity α, the number of trucks $|\mathcal{K}|$, and the number of drones per truck $|\mathcal{D}|$. Overall, we can observe logarithmic-shaped curves in all cases, indicating a diminishing return with regard to an increase in the relative endurance. Recall that a value of $\mathcal{E}_r = 0$ indicates that no drone operation is possible while a value of $\mathcal{E}_r = 2$ indicates that, in principle, all operations are feasible. In all cases, even small increases in the drone's endurance can lead to significant savings (e. g., moving $\mathcal{E}_r$ from 0 to 1) whereas further increases often lead to smaller returns (e. g., moving $\mathcal{E}_r$ from 1 to 2). In general, α and $\mathcal{E}_r$ seem strongly correlated. If the drone is relatively slow, performing operations that require a large endurance does not seem to be beneficial. In contrast, if the drone is relatively fast, it can be beneficial to perform operations that require a high endurance and visit distant locations.

For a more detailed analysis on the performed operations, see Figure 3.5.2. This figure shows the average number of acyclic and cyclic operations performed by drones as a function of the relative endurance $\mathcal{E}_r$, the relative velocity α, the number of trucks $|\mathcal{K}|$, and the number of drones per truck $|\mathcal{D}|$. Overall, the share of acyclic operations performed is much larger than the share of cyclic operations. Moreover, the number of acyclic operations performed as a function of the relative endurance also appears logarithmic-shaped. This might be an indication that after reaching a certain threshold of relative endurance $\mathcal{E}_r$, depending on the relative velocity α, the operations performed by drones will not change structurally. These results are in line with Figure 3.5.1. However, we also have an indication that an improvement in Δ is not necessarily linked to additional drone operations. To this end, consider, e. g., the cases for $\alpha \in \{2, 3\}, |\mathcal{K}| = 1, |\mathcal{D}| = 1$ in Figures 3.5.1 and 3.5.2. Even though the number of acyclic and cyclic operations is almost identical, the relative improvements Δ are very much different. Therefore, in some cases, it can be beneficial to perform the same number of operations but at an increased relative velocity. Clearly, once the drone is fast enough, the velocity of the truck becomes the bottleneck such that an increase in the relative velocity yields no further improvement.

Finally, Figure 3.5.3 shows the relative improvement as box plots based on the 10 different base instances (specifying the customers and depot) that we used in our experiments. On this figure, we observe that even though the median relative improvement is typically large, the minimum and maximum values can differ noticeably. In particular for the cases of $\alpha = 3$, the difference between the maximum and minimum relative improvement when comparing different problem instances can be large.

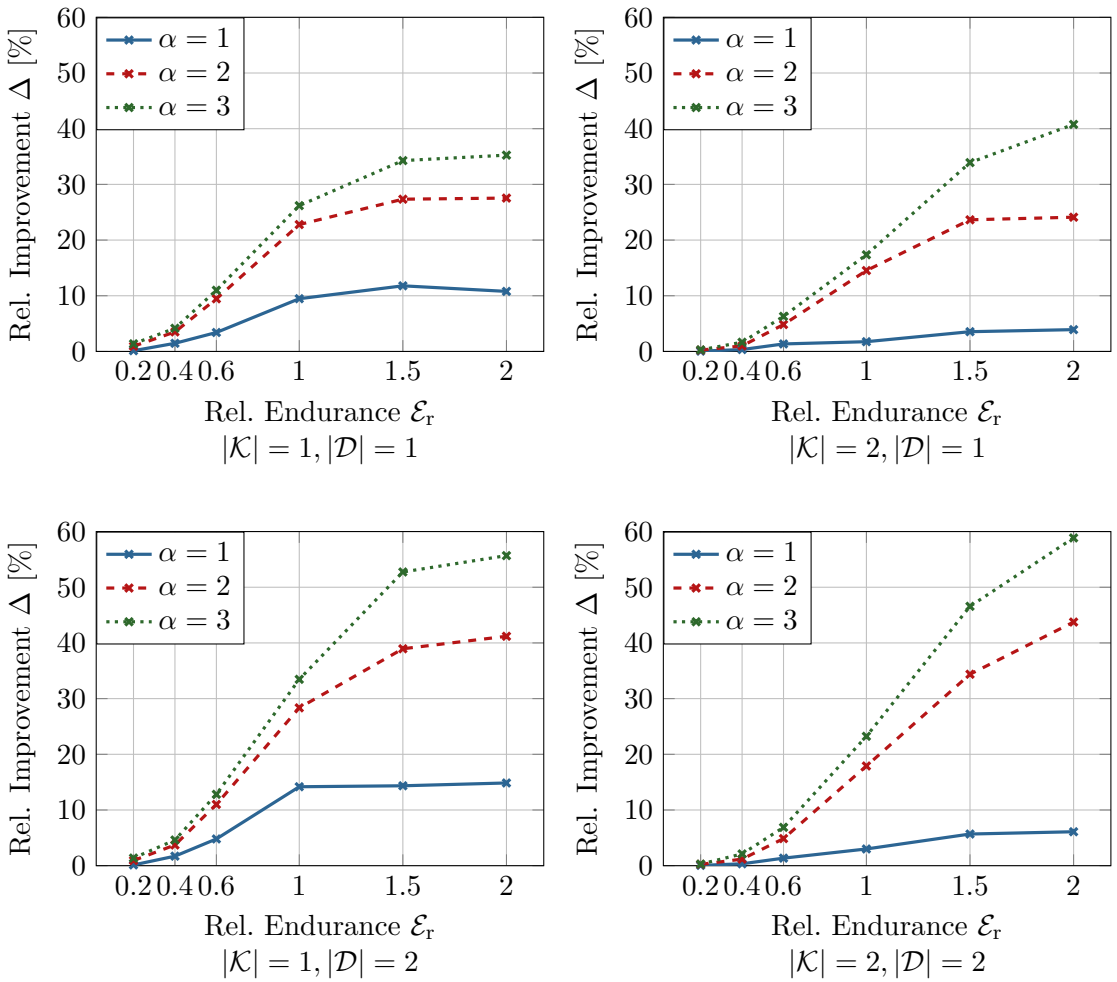

Figure 3.5.1: The relative improvement Δ depending on the number of trucks $|\mathcal{K}|$, the number of drones per truck $|\mathcal{D}|$, the relative endurance $\mathcal{E}_r$, and the relative velocity α (based on 10 instances).

3.5.1.2 Small Instances used by Murray and Chu (2015)

To show that our model and heuristic are also well-suited to the assumptions and instances of Murray and Chu (2015), we provide some computational evidence in this section. In these instances, there is a single truck which is equipped with a single drone that follow a Manhattan and Euclidean distance matrix at 25 miles per hour (mph) and 15, 25, or 35 mph, respectively. For the sake of compactness, in this section, we respect these assumptions and do not consider a multi-truck or multi-drone case. Murray and Chu (2015) provide 72 problems that contain 10 customers distributed across an 8-mile square region (80% to 90% of which are drone serviceable) and 1 depot location i. e., 11 vertices in total (one more vertex compared to the previous section). Among these 72 instances,

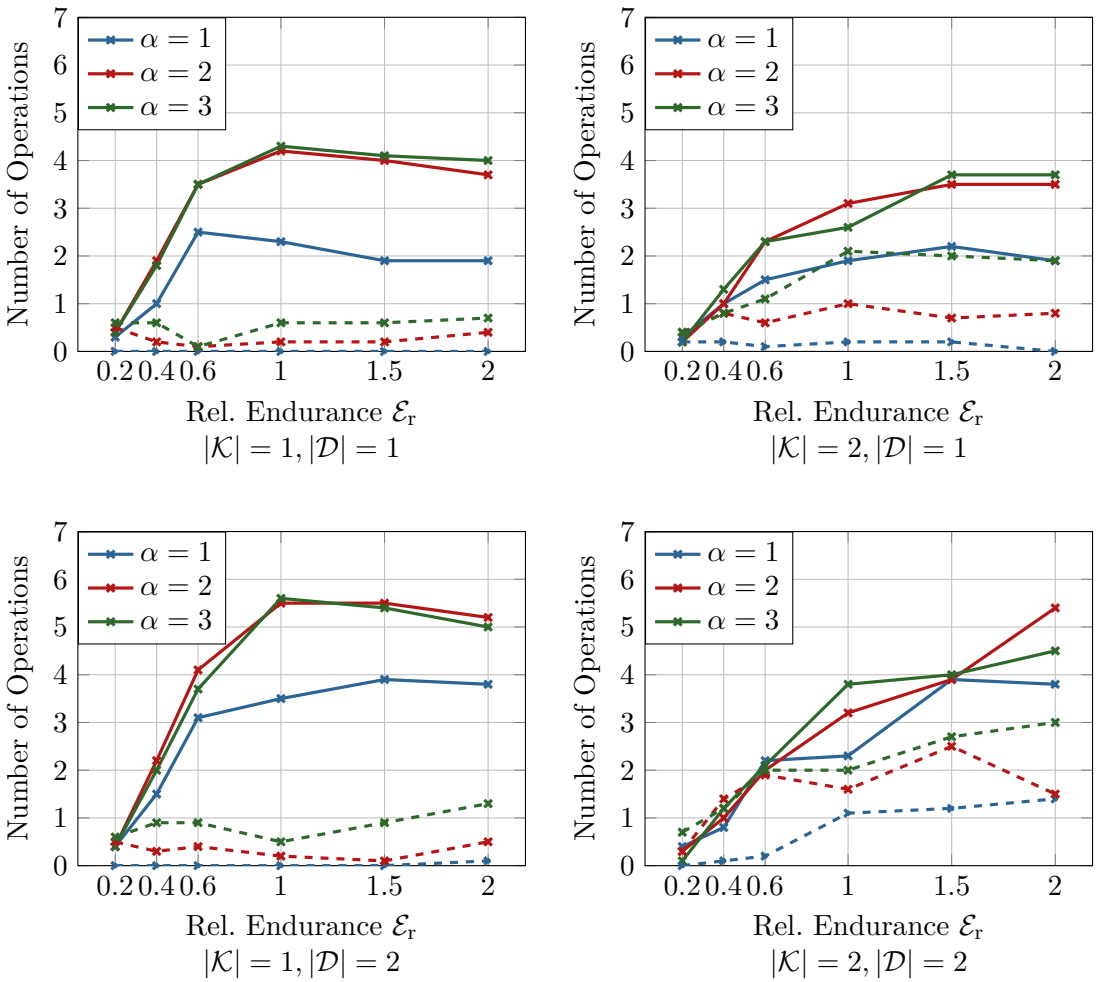

Figure 3.5.2: The average share of acyclic operations (solid lines) and cyclic operations (dashed lines) depending on the number of trucks $|\mathcal{K}|$, the number of drones per truck $|\mathcal{D}|$, the relative endurance $\mathcal{E}_r$, and the relative velocity α (based on 10 instances).

the endurance of the drone is either 20 or 40 minutes.

Several minor modifications to the VRPD and DASP MILPs are necessary to match the problem formulation of Murray and Chu (2015). A detailed discussion is provided in Appendix 3.B. Due to the effectiveness of the VIEQs (refer to Section 3.5.1.1), we solved the modified MILP through Gurobi Optimizer in the presence of all VIEQs that are proposed in Section 3.3.2. For comparative purposes, we also solved the same set of instances through the matheuristic with the three-index formulation using the same set of parameters that was described in the previous section. The detailed results are available in Table 3.D.6 (refer to Appendix 3.D).

Noteworthy, Murray and Chu (2015) report that they were unable to solve any of these

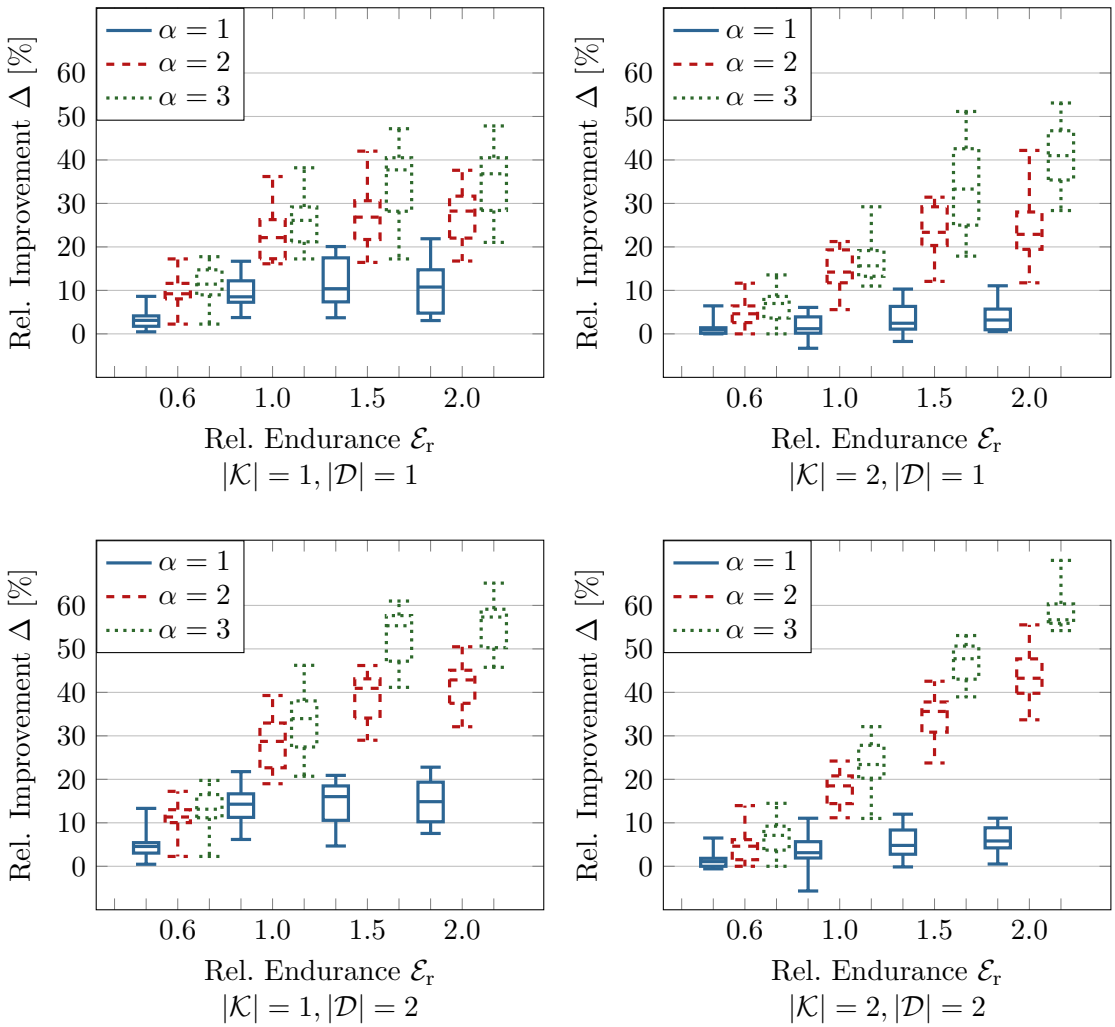

Figure 3.5.3: The relative improvement Δ as a box plot depending on the number of trucks $|\mathcal{K}|$, the number of drones per truck $|\mathcal{D}|$, the relative endurance $\mathcal{E}_r$, and the relative velocity α. Values taken from Table 3.D.3 (averaged over 10 instances).

problems to proven optimality within 30 minutes of runtime. Compared to this, within 15 minutes of runtime, we can solve 26 out of 72 instances to optimality. The average MIP gap over all instances remained 16.69% after runtime. As we observed in Section 3.5.1.1, an increased endurance (i. e., flight time $\mathcal{E}_t$) generally decreases the performance of the MILP solver. This is also evident by the average runtime of 535.0 seconds and MIP gap of 11.4% (781.9 seconds and 22.0%) after runtime when the maximum flight time is set to 20 minutes (40 minutes).

Table 3.5.2 provides some additional information based on the solutions returned by the MILP solver. More precisely, this table contains the average savings Δ (refer to

Formula (3.80)) and the average number of drone operations present in the solution vector. Moreover, information on the root relaxation solution is shown (similar to Table 3.5.1). This table highlights that slightly smaller savings of a similar magnitude are possible on these instances compared to the ones observed in Section 3.5.1.1. Furthermore, the dependence of the savings on the drone velocity and endurance structurally matches the one observed in Figure 3.5.1.

Table 3.5.2: The savings Δ, the MIP gap, and the number of drone operations (Ops.) on the instances of Murray and Chu (2015) as well as information on the root relaxation solution (averaged over 12 instances).

Drone Speed (mph)	$\mathcal{E}_t$=20 minutes						$\mathcal{E}_t$=40 minutes					
	Solution			Root			Solution			Root		
	Δ	Gap	Ops.	Iterations	Time	$\%_R$	Δ	Gap	Ops.	Iterations	Time	$\%_R$
15	2.7%	0.0%	0.7	1,028.5	0.4	58.0%	10.2%	22.0%	1.0	1,123.8	1.4	54.6%
25	15.7%	17.2%	2.1	1,174.8	1.1	52.8%	17.6%	23.9%	2.2	1,298.6	1.7	53.9%
35	23.0%	16.9%	2.9	1,400.7	1.7	54.0%	22.8%	20.3%	3.0	1,417.6	1.8	53.8%
Average	13.8%	11.4%	1.9	1,201.3	1.1	54.9%	16.8%	22.0%	2.1	1,280.0	1.6	54.1%

With respect to the *Best Known Solutions* (BKSs) provided by (Ha et al., 2018; Murray and Chu, 2015), both our MILP and matheuristic showed favorable performance. More precisely, both approaches were able to generate solutions that were on average within a 0.5% margin to the BKS. Moreover, to the best of our knowledge, we believe that we were able to identify several new and slightly improved BKSs through the MILP solver and matheuristic on 13 instances (refer to Table 3.D.6).

3.5.2 Large Instances

Regarding the large instances, we also used the instances provided by Agatz et al. (2018) with 20, 50, and 100 vertices, respectively. In accordance with the explanations in Section 3.5.1.1, for each size, there are ten instances that are associated with a relative velocity $\alpha \in \{1, 2, 3\}$ and a relative endurance $\mathcal{E}_r \in \{0.2, 0.4, 0.6, 1.0, 1.5, 2.0\}$. For our experiments, we limit ourselves to $\mathcal{E}_r \leq 1$ as no significant differences can be observed at larger endurances (see the results in Section 3.5.1.1). Furthermore, we test with $|\mathcal{K}| \in \{1, 2, 3\}$ and $|\mathcal{D}| \in \{1, 2\}$, i. e., we have one, two, or three trucks that are equipped with either one or two drones each. This yields a total of $3 \cdot 10 \cdot 3 \cdot 4 \cdot 3 \cdot 2 = 2160$ instances. On each instance, we run the matheuristic 5 times using either the three- or two-index formulation of the DASP, yielding a total of 21600 unique experiments.

In order to achieve an improved runtime behavior, in particular when solving instances with 100 customers while a single truck is present, we use the DASP partition (see Al-

gorithm 3.2). To this end, a route containing more than 20 vertices is split into multiple (equally large) partitions such that no partition contains more than 20 vertices.

As we have no optimal objective values for these problems, we introduce the following updated metric Δ for studying the benefit of using drones. Given a fixed number of trucks, the updated metric gives insights into the change in makespan through the utilization of drones:

$$\Delta \coloneqq 100\% - \frac{\text{incumbent objective value}}{\text{incumbent objective value before solving the DASP}} \tag{3.82}$$

Tables 3.D.7–3.D.12 show the results from solving the instances with 20, 50, and 100 vertices by using the matheuristic with either the three- or two-index formulation for solving the DASP. Overall, on larger instances the matheuristic also converges very fast. Whereas preliminary results have shown that the three-index formulation typically performs better in solving larger DASPs, when using the DASP partition, there is no significant performance difference in using the three- or two-index formulation for solving the MILPs within the heuristic. Furthermore, similar observations regarding the share of acyclic and cyclic operations can be made, i. e., when the relative velocity is small ($\alpha \in \{1, 2\}$), then the share of cyclic operations also tends to be very small irrespective of the number of drones. Apart from this observation, there is no significant increase in the number of acyclic operations performed when moving from $\alpha = 2$ to $\alpha = 3$.

Figure 3.5.4 visualizes the relative improvement Δ (according to Formula (3.82)) depending on the number of trucks, drones per truck, instance size, relative velocity, and endurance, respectively, based on the results achieved through the three-index formulation. The basic shape of these curves is very similar to the one that we observed in Figure 3.5.1. That is, on the one hand, increasing the relative endurance $\mathcal{E}_r$ quickly leads to a threshold where a further increase does not yield any changes to the relative improvement. In particular, based on the definition of the relative endurance in Formula (3.79), for a fixed relative endurance, the larger the instance, the larger the number of possible operations. Therefore, we can observe a lower threshold for larger (or more dense) instances. Furthermore, on the other hand, an increase in the relative velocity leads to a greater relative improvement in all cases. We can also observe that the smaller the number of trucks (or, the larger the number of customers served by each truck) the larger the relative improvement gained by increasing α. These improvements heavily depend on the number of drones. In most cases, the improvement gained by increasing α from 2 to 3 is significantly larger in the presence of two drones than in the presence of a single drone per truck. Finally, we can also observe that substitutability is possible. Therefore, in some cases, a single fast moving drone (e. g., $\alpha = 3$) allows for a similar relative improvement as two slower drones (e. g., $\alpha = 2$ with possibly reduced endurance).

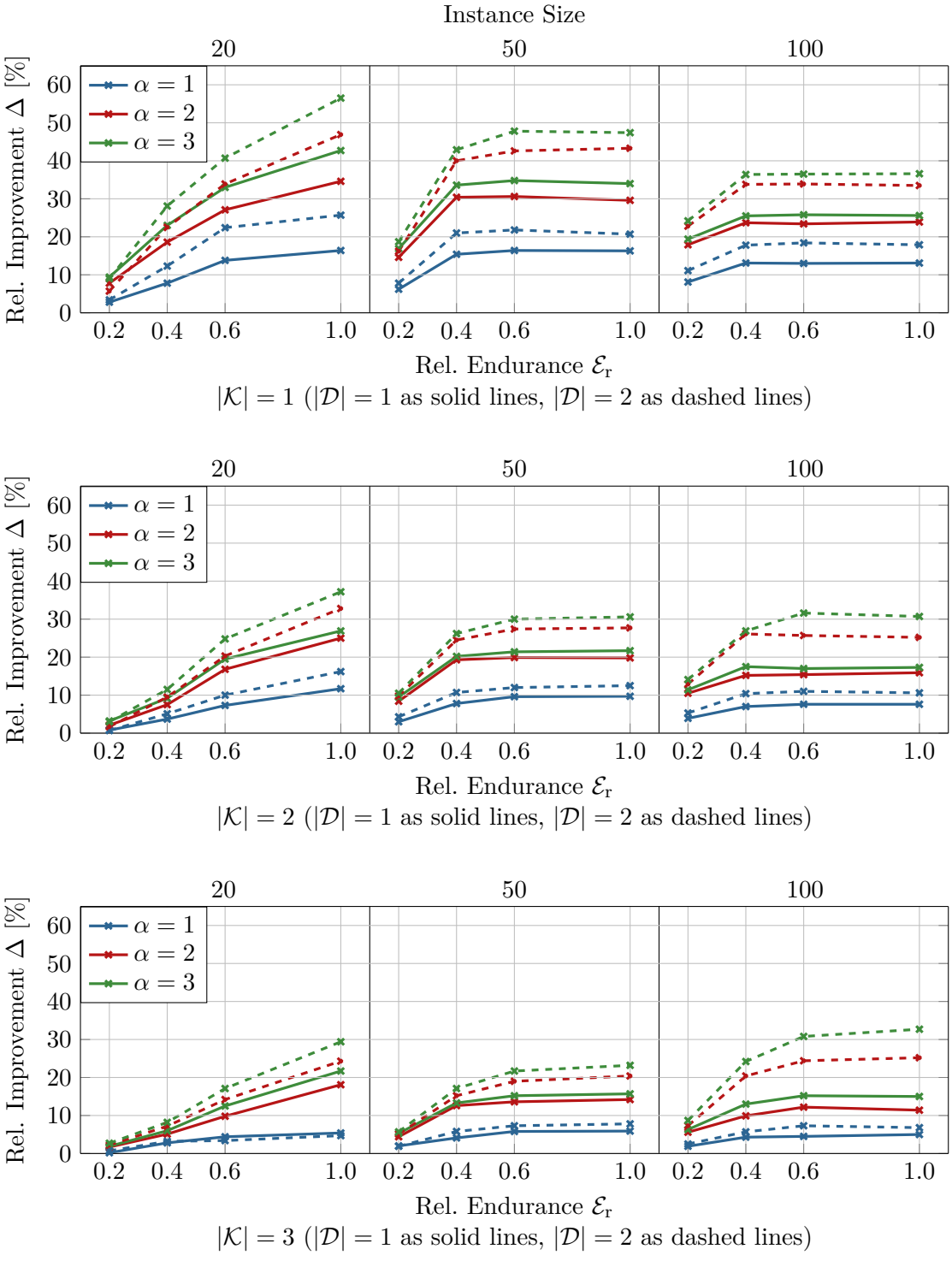

Figure 3.5.4: The relative improvement Δ depending on the instance size, number of trucks $|\mathcal{K}|$, number of drones per truck $|\mathcal{D}|$, relative endurance $\mathcal{E}_r$, and relative velocity α based on Tables 3.D.7, 3.D.9, and 3.D.11.

3.5.3 Comparison to Theoretical Bounds

As discussed in Section 3.2, X. Wang et al. (2017) provide several theoretical insights on the VRPD. In this section, we are interested in investigating these theoretical bounds and comparing them to the numerical results of our study. In particular, we are interested in a bound that concerns the relative completion ratio Z_{r}. Let $Z(\mathrm{VRPD}_{|\mathcal{K}|,\alpha,|\mathcal{D}|})$ be the optimal objective value of the VRPD for a fleet consisting of $|\mathcal{K}|$ trucks equipped with $|\mathcal{D}|$ drones each. Each drone is assumed to move at a relative velocity α (with unlimited endurance, i. e., $\mathcal{E}_r = 2$). In addition, we denote the optimal objective value by $Z(\mathrm{VRP}^*)$, if we use the same fleet, but with no drones. Wang et al. have introduced the following bound:

$$Z_{\mathrm{r,max}} = \frac{Z(\mathrm{VRP}^*)}{Z(\mathrm{VRPD}_{|\mathcal{K}|,\alpha,|\mathcal{D}|})} \leq \alpha\,|\mathcal{D}| + 1. \tag{3.83}$$

Table 3.5.3 highlights the investigated cases and the corresponding results. Depending on the number of drones and their relative velocity, the table shows the bound on the relative completion ratio ($Z_{\mathrm{r,max}}$), the analogous maximum possible savings $\Delta_{\max}$ (see Equation (3.80)), and the average savings observed for one and two trucks, respectively, based on Table 3.D.3 for $\mathcal{E}_r = 2$.

Table 3.5.3: Comparison to theoretical bounds for the cases investigated in this work based on (X. Wang et al., 2017).

$\lvert\mathcal{D}\rvert$	-	1			2		
α	0	1	2	3	1	2	3
$Z_{\mathrm{r,max}} \leq$	1	2	3	4	3	5	7
$\Delta_{\max} = 100\% - \frac{1}{Z_{\mathrm{r}}} \leq$	0.0%	50.0%	66.66%	75.0%	66.66%	80.0%	85.7%
$\Delta_{\mathrm{avg},\lvert\mathcal{K}=1\rvert}$	0.0%	10.8%	27.5%	35.3%	14.9%	41.2%	55.7%
$\Delta_{\mathrm{avg},\lvert\mathcal{K}=2\rvert}$	0.0%	3.9%	24.1%	40.8%	6.1%	43.7%	58.9%

Overall, the savings (or completion ratios) that we observed in our numerical study are significantly smaller than the maximum possible theoretical savings. These observations apply to both, the small instances (see Section 3.5.1) and heuristic solutions to large instances (see Section 3.5.2).

3.6 Conclusion

In this paper, we investigated the VRPD. To be more precise, we began in Section 3.2 with a brief overview of related problems in last-mile delivery that feature trucks and drones which have intensive synchronization requirements. Afterwards, in Section 3.3 we provided a MILP formulation of the VRPD and derived several VIEQs. In Section 3.4, for solving

the VRPD, we introduced a matheuristic that can effectively leverage any MILP solver by exploiting the structure of the VRPD. As part of this matheuristic, we introduced the DASP that, given a feasible routing, finds an optimal assignment and schedule of drones such that the makespan is minimized. In this context, we provided two different MILP formulations for the DASP. Finally, in Section 3.5, we presented the results and observations of our extensive computational experiments. The numerical results indicate that the use of drones can significantly reduce the makespan. Furthermore, we have shown that similar makespan reductions can be achieved with different drone designs. According to the numerical results, our matheuristic is very fast, converges quickly, and is able to consistently provide optimal or near optimal solutions.

Throughout this work, we have hinted at several potential opportunities for future research. In particular, we have shown that different recharge policies or a mixed drone fleet might be of future research interest (see Appendix 3.A). Furthermore, even though our matheuristic performed very well overall, we believe that it might be interesting to embed it as a component within a more sophisticated metaheuristic. Finally, it might be worthwhile to investigate the existence of more compact MILP formulations.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments and suggestions that helped us in improving the quality of this paper.

3.7 References

Agatz, Niels, Paul Bouman, and Marie Schmidt (2018). "Optimization Approaches for the Traveling Salesman Problem with Drone". In: *Transportation Science* 52.4, pp. 965–981.

Archetti, Claudia and M. Grazia Speranza (2014). "A survey on matheuristics for routing problems". In: *EURO Journal on Computational Optimization* 2.4, pp. 223–246.

Bouman, Paul, Niels Agatz, and Marie Schmidt (2018a). "Dynamic programming approaches for the traveling salesman problem with drone". In: *Networks* 72.4, pp. 528–542.

Boysen, Nils et al. (2018b). "Drone delivery from trucks: Drone scheduling for given truck routes". In: *Networks* 72.4, pp. 506–527.

Carlsson, John Gunnar and Siyuan Song (2018). "Coordinated Logistics with a Truck and a Drone". In: *Management Science* 64.9, pp. 3971–4470.

Clarke, Geoff and John W. Wright (1964). "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points". In: *Operations Research* 12.4, pp. 568–581.

Di Puglia Pugliese, Luigi and Francesca Guerriero (2017). "Last-Mile Deliveries by Using Drones and Classical Vehicles". In: *Optimization and Decision Science: Methodologies and Applications*. Springer Proceedings in Mathematics & Statistics 217. Cham: Springer, pp. 557–565.

Dorling, Kevin et al. (2017). "Vehicle Routing Problems for Drone Delivery". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 47.1, pp. 70–85.

Drexl, Michael (2012b). "Synchronization in Vehicle Routing—A Survey of VRPs with Multiple Synchronization Constraints". In: *Transportation Science* 46.3, pp. 297–316.

Es Yurek, Emine and H. Cenk Ozmutlu (2018). "A decomposition-based iterative optimization algorithm for traveling salesman problem with drone". In: *Transportation Research Part C: Emerging Technologies* 91, pp. 249–262.

Gurobi Optimization, LLC (2023). *Gurobi Optimizer Reference Manual*.

Ha, Quang Minh et al. (2018). "On the min-cost Traveling Salesman Problem with Drone". In: *Transportation Research Part C: Emerging Technologies* 86, pp. 597–621.

Jünger, Michael et al., eds. (2010). *50 Years of Integer Programming 1958-2008*. Berlin, Heidelberg: Springer.

Lin, S. and B. W. Kernighan (1973). "An Effective Heuristic Algorithm for the Traveling-Salesman Problem". In: *Operations Research* 21.2, pp. 498–516.

Marinelli, Mario et al. (2018). "En route truck–drone parcel delivery for optimal vehicle routing strategies". In: *IET Intelligent Transport Systems* 12.4, pp. 253–261.

Murray, Chase C. and Amanda G. Chu (2015). "The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery". In: *Transportation Research Part C: Emerging Technologies* 54, pp. 86–109.

Otto, Alena et al. (2018). "Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey". In: *Networks* 72.4, pp. 411–458.

Poikonen, Stefan, Xingyin Wang, and Bruce Golden (2017). "The vehicle routing problem with drones: Extended models and connections". In: *Networks* 70.1, pp. 34–43.

Sacramento, David, David Pisinger, and Stefan Ropke (2019). "An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones". In: *Transportation Research Part C: Emerging Technologies* 102, pp. 289–315.

Schermer, Daniel, Mahdi Moeini, and Oliver Wendt (2019a). "A hybrid VNS/Tabu search algorithm for solving the vehicle routing problem with drones and en route operations". In: *Computers & Operations Research* 109, pp. 134–158.

Toth, Paolo and Daniele Vigo, eds. (2014). *Vehicle routing: problems, methods, and applications*. 2nd ed. MOS-SIAM series on optimization. Philadelphia: SIAM.

Wang, Xingyin, Stefan Poikonen, and Bruce Golden (2017). "The vehicle routing problem with drones: several worst-case results". In: *Optimization Letters* 11.4, pp. 679–697.

Appendix

3.A Possible Variations

As we have seen in Section 3.2, the literature on optimization problems with drones in the context of last-mile delivery is vast and fast-growing. Moreover, even though the basic assumptions of these problems are commonly accepted by most researchers, there is no general consensus on more detailed aspects of the problem. These aspects concern questions such as whether cyclic (loop) operations are allowed and if trucks as well as drones follow the same distance metric. In particular, while waiting for a truck to arrive, a relevant question is whether drones can safely land and remain stationary on the ground or if they must continue hovering. This has different implications for distance and time constrained endurance (refer to Appendix 3.C). With the purpose of illustrating the flexibility of the proposed model, as we will see in 3.A.1–3.A.6, our model can be easily adapted to cases where different assumptions might apply.

3.A.1 Limited Flight Distance without Recharge

In this section, we show that it is possible to consider the case where a drone has a limited range of operation and cannot be recharged. In the MILP (3.1)–(3.27), this can be done by replacing constraints (3.24) with the following constraints that account for the total distance covered during all operations for each drone:

$$\sum_{i \in V_L} \sum_{\substack{w \in V_D, \\ i \neq w}} \sum_{\substack{j \in V, \\ w \neq j}} (\overline{d}_{iw} + \overline{d}_{wj})\, y_{iwj}^{kd} \leq \mathcal{E} \quad \forall k \in \mathcal{K}, d \in \mathcal{D}. \tag{3.84}$$

3.A.2 Limited Flight Time with Instantaneous Recharge

If a drone cannot safely remain stationary on its own while waiting for the arrival of a truck, we might consider a maximum flight time including hovering time. To this end, the endurance (in time units) might be limited by a maximum flight time $\mathcal{E}_t$, rather than distance. If we assume that the endurance is recharged after each operation (e. g., through a swap of the battery), we can introduce the following sets of inequalities as a replacement for constraints (3.24) that consider the maximum flight time $\mathcal{E}_t$ instead:

$$M(y_{iwj}^{kd} - 1) + (r_j^{kd} - s_i^{kd}) \leq \mathcal{E}_t \quad \forall k \in \mathcal{K}, d \in \mathcal{D}, i \in V_L, w \in V_D, j \in V_R, |\{i, w, j\}| = 3, \tag{3.85}$$

$$(\overline{t}_l + \overline{t}_{iw} + \overline{t}_{wi} + \overline{t}_r)\, y_{iwi}^{kd} \leq \mathcal{E}_t \quad \forall k \in \mathcal{K}, d \in \mathcal{D}, i \in V_L, w \in V_D, i \neq w. \tag{3.86}$$

In this case, the set of inequalities in (3.85) and (3.86) account for the time spent performing acyclic and cyclic operations, respectively. In particular, in the case of acyclic operations, the hovering time is also considered if the drone arrives before the truck. Note that constraints (3.85) imply that a drone must *immediately* depart from the customer. If a drone can safely remain stationary at the place of delivery for some time, the case discussed in Appendix 3.C applies.

3.A.3 Limited Flight Time without Recharge

Analogously, we might consider a case where the drone has a limited flight time and an endurance that cannot be recharged. In this case, the following (non-linear) set of constraints might be introduced instead of constraints (3.24):

$$\sum_{i \in V_L} \sum_{\substack{w \in V_D, \\ i \neq w}} (\bar{t}_l + \bar{t}_{iw} + \bar{t}_{wi} + \bar{t}_r)\, y_{iwi}^{kd} + \sum_{i \in V_L} \sum_{\substack{w \in V_D, \\ i \neq w}} \sum_{\substack{j \in V_R, \\ i \neq j, \\ w \neq j}} (r_j^{kd} - s_i^{kd})\, y_{iwj}^{kd} \leq \mathcal{E}_t \quad \forall k \in \mathcal{K}, d \in \mathcal{D}. \tag{3.87}$$

These constraints guarantee that the total time spent performing operations (including hovering time) must be less than or equal to the endurance $\mathcal{E}_t$. As an alternative, to provide a linear model, we might introduce additional continuous decision variables q_i^{kd} and $\overline{q}_i^{kd}$ where $k \in \mathcal{K}, d \in \mathcal{D}, i \in V$ that can take a value in $[0, \mathcal{E}_t]$. The variables q_i^{kd} and $\overline{q}_i^{kd}$ indicate the remaining flight time of drone d (that belongs to truck k) when initially retrieved and after departing from vertex i, respectively (we assume that $q_0^{kd} = \mathcal{E}_t$). With these additional variables, we might use the following sets of inequalities instead of constraints (3.87) to provide a linear model:

$$M(1 - x_{ij}^{kd}) + \overline{q}_i^{kd} \geq q_j^{kd} \quad \forall k \in \mathcal{K}, d \in \mathcal{D}, i \in V_L, j \in V_R, i \neq j, \tag{3.88}$$

$$q_i^{kd} - \sum_{\substack{w \in V_D, \\ i \neq w}} (\bar{t}_l + \bar{t}_{iw} + \bar{t}_{wi} + \bar{t}_r)\, y_{iwi}^{kd} \geq \overline{q}_i^{kd} \quad \forall k \in \mathcal{K}, d \in \mathcal{D}, i \in V_L, \tag{3.89}$$

$$M(1 - y_{iwj}^{kd}) + \overline{q}_i^{kd} - (r_j^{kd} - s_i^{kd}) \geq q_j^{kd} \quad \forall k \in \mathcal{K}, d \in \mathcal{D}, i \in V_L, w \in V_D, j \in V_R, |\{i, w, j\}| = 3. \tag{3.90}$$

Constraints (3.88) guarantee that for any transition x_{ij}^k that is performed by a truck, the remaining endurance of drone d upon arrival at j must be less than or equal to the remaining endurance upon departure at i. In between arrival and departure, further charge might be used to perform cyclic operations which are accounted for by the set of constraints (3.89). Finally, the inequalities in (3.90) account for a depletion of the drone's

battery for performing acyclic operations.

3.A.4 Limited Flight Time with Non-instantaneous Recharge

Similar to the previous section, we might consider a scenario where a drone's battery can be recharged non-instantaneously. In this case, we assume that the battery of the drone might be recharged while the truck is powered (in motion). To this end, we might use the decision variables q_i^{kd} and $\overline{q}_i^{kd}$ that were introduced in the previous section and add the set of constraints (3.89) and (3.90) as well as the following additional constraint:

$$M(1-x_{ij}^{kd})+\overline{q}_i^{kd}+\frac{t_{ij}}{\kappa}(y_i^{kd}-\sum_{\substack{w\in V_D,\\ i\neq w}}\sum_{\substack{h\in V_R,\\ i\neq h,\\ w\neq h}} y_{iwh}^{kd})\geq q_i^{kd} \quad \forall k\in\mathcal{K}, d\in\mathcal{D}, i\in V_L, j\in V_R, i\neq j. \tag{3.91}$$

This constraint guarantees that for any transition x_{ij}^k performed by truck k where the drone d is available at i (i. e., $y_i^{kd}=1$) and performs no acyclic operation (i. e., the drone remains stationary on the truck during the transition to j), the endurance might be recharged with t_{ij}/κ, where κ is a parameter that determines the charging rate for each unit of time that the truck is in motion.

If we assume that a drone can also be recharged while a truck remains stationary, it is possible to introduce further continuous decision variables h_i^{kd} with lower bound 0 that denote the additional waiting time of drone d that remains stationary on truck k at vertex i. Then, we might replace inequalities (3.21) with the following set of inequalities which guarantee that the drone departure time s_i^{kd} is bound by the release time r_i^{kd}, the time spent performing operations at i, and the waiting time h_i^{kd}.

$$r_i^{kd}+\sum_{\substack{w\in V_D,\\ i\neq w}}(\overline{t}_l+\overline{t}_{iw}+\overline{t}_{wi}+\overline{t}_r)\,y_{iwi}^{kd}+h_i^{kd}\leq s_i^{kd} \quad \forall k\in\mathcal{K}, d\in\mathcal{D}, i\in V. \tag{3.92}$$

With these variables h_i^{kd}, we might replace constraints (3.89) with the following set of constraints that allow a drone to be recharged at a rate of κ' per unit of time that it remains on a stationary truck:

$$q_i^{kd}-\sum_{\substack{w\in V_D,\\ i\neq w}}(\overline{t}_l+\overline{t}_{iw}+\overline{t}_{wi}+\overline{t}_r)\,y_{iwi}^{kd}+\frac{h_i^{kd}}{\kappa'}\geq \overline{q}_i^{kd} \quad \forall k\in\mathcal{K}, d\in\mathcal{D}, i\in V_L. \tag{3.93}$$

Note that similar considerations are easily possible if the endurance is limited by distance (i. e., excluding hovering time) rather than time.

3.A.5 Heterogeneous Drone Fleet

Moreover, since each drone is referenced by a specific index, it is easily possible to consider a mixed drone fleet where each truck is equipped with a different number of drones which might have different velocities and endurances. This can be done by introducing specific drone travel times $\bar{t}_{ij}^{kd}$ and endurances $\mathcal{E}^{kd}$.

3.A.6 Incorporating these Variations into the DASP

As a concluding remark, note that the possible variations on MILP (3.1)–(3.27), that were implied in Sections 3.A.1–3.A.5, might be easily incorporated into either model for the DASP. All variations to the MILP require minor adjustments to constraints (3.21) and (3.24). Note that there is a direct correspondence between these constraints and constraints (3.47), (3.50), and (3.51) for the three-index formulation (refer to Section 3.4.2.1) and constraints (3.71), 3.74, and (3.75) for the two-index formulation (refer to Section 3.4.2.2). Moreover, most variations require the variables r_i^{kd} and s_i^{kd} from MILP (3.1)–(3.27). The equivalent variables are already present in both proposed DASP models as r_i^d and s_i^d. Any additional variables or parameters that were introduced in Sections 3.A.1–3.A.5 might be added to the DASP models analogously.

Appendix 3.B provides a guideline on how all MILPs must be adjusted to respect the assumptions of Murray and Chu (2015). This appendix also provides an illustrative example on how, e. g., the endurance constraints must be adjusted to respect the maximum flight time.

3.B Matching the Problem Formulation of Murray and Chu

In this section, we describe how the MILP formulations of the VRPD and DASP might be adjusted to respect the problem formulation of Murray and Chu (2015). As outlined in Section 3.2, several slightly deviating assumptions apply for this problem. More precisely, cyclic drone-operations are prohibited and the overhead times and associated temporal constraints are treated in a slightly different way. In particular, the endurance is measured in units of time rather than units of distance. For a formulation that follows the assumptions of Murray and Chu (2015), we propose MILP (3.1)–(3.19), (3.94)–(3.99), (3.25)–(3.27).

$$M(y_{iwj}^{kd} - 1) + s_i^{kd} + \bar{t}_{iw} \leq r_w^{kd} \quad \begin{array}{l} \forall k \in \mathcal{K}, d \in \mathcal{D}, i \in V_L, w \in V_D, \\ j \in V_R, |\{i, w, j\}| = 3, \end{array} \tag{3.94}$$

$$M(y^{kd}_{iwj} - 1) + s^{kd}_w + \bar{t}_{wj} + \bar{t}_r \leq r^{kd}_j \quad \forall k \in \mathcal{K}, d \in \mathcal{D}, i \in V_L, w \in V_D, j \in V_R, |\{i, w, j\}| = 3, \tag{3.95}$$

$$r^{kd}_i \leq s^{kd}_i \quad \forall k \in \mathcal{K}, d \in \mathcal{D}, i \in V, \tag{3.96}$$

$$s^{kd}_i \leq s^{k}_i \quad \forall k \in \mathcal{K}, d \in \mathcal{D}, i \in V, \tag{3.97}$$

$$a^k_j + \bar{t}_l \sum_{\substack{w \in V_D, \\ w \neq j}} \sum_{\substack{h \in V_r \\ j \neq h, \\ w \neq h}} y^{kd}_{jwh} + \bar{t}_r \sum_{\substack{i \in V_L \\ i \neq j}} \sum_{\substack{w \in V_D, \\ i \neq w, \\ w \neq j}} y^{kd}_{iwj} \leq r^{kd}_j \quad \forall k \in \mathcal{K}, d \in \mathcal{D}, j \in V_R, \tag{3.98}$$

$$M(y^{kd}_{iwj} - 1) + (r^{kd}_j - s^{kd}_w) + \bar{t}_{iw} \leq \mathcal{E}_t \quad \forall k \in \mathcal{K}, d \in \mathcal{D}, i \in V_L, w \in V_D, j \in V_R, |\{i, w, j\}| = 3. \tag{3.99}$$

In this model, we ask that all variables y_{iwi} where $i \in V_L, w \in V_D$ are bound to 0 to prevent any cyclic operation from occurring. The updated temporal constraints (3.94)–(3.98) are consistent with the model of Murray and Chu (2015). In particular, the overhead times are treated in a slightly different way through constraints (3.94), (3.95), and (3.98). Finally, constraints (3.99) provide an endurance based on flight time that is coherent with the one proposed by Murray and Chu (2015). Similar adjustments can be made for the DASP models, where it is also necessary to bind all variables y_{iwi} and z_{iw} where $i \in V_L, w \in V_D$ to values 0. For the three-index formulation, we propose MILP (3.39)–(3.44), (3.100)–(3.105), (3.52)–(3.54).

$$-M(1 - y^{d}_{iwj}) + s^{d}_i + \bar{t}_{iw} \leq r^{d}_w \quad \forall d \in \mathcal{D}, i \in V_L, w \in V_D, j \in V_R, i \prec w \prec j, \tag{3.100}$$

$$-M(1 - y^{d}_{iwj}) + s^{d}_w + \bar{t}_{wj} + \bar{t}_r \leq r^{d}_j \quad \forall d \in \mathcal{D}, i \in V_L, w \in V_D, j \in V_R, i \prec w \prec j, \tag{3.101}$$

$$r^{d}_i \leq s^{d}_i \quad \forall d \in \mathcal{D}, i \in V, \tag{3.102}$$

$$s^{d}_i \leq s_i \quad \forall d \in \mathcal{D}, i \in V, \tag{3.103}$$

$$a_j + \bar{t}_l \sum_{\substack{w \in V_D, \\ j \prec w}} \sum_{\substack{g \in V_R, \\ w \prec g}} y^{d}_{jwg} + \bar{t}_r \sum_{\substack{i \in V_L, \\ i \prec w}} \sum_{\substack{w \in V_D, \\ w \prec j}} y^{d}_{iwj} \leq r^{d}_j \quad \forall d \in \mathcal{D}, j \in V_R, \tag{3.104}$$

$$M(y^{d}_{iwj} - 1) + (r^{d}_j - s^{d}_w) + \bar{t}_{iw} \leq \mathcal{E}_t \quad \forall k \in \mathcal{K}, d \in \mathcal{D}, i \in V_L, w \in V_D, j \in V_R, i \prec w \prec j. \tag{3.105}$$

Analogously, we propose MILP (3.61)–(3.68), (3.106)–(3.111), (3.76)–(3.78) for the two-index formulation.

$$-M(1 - y^{d}_{iw}) + s^{d}_i + \bar{t}_{iw} \leq r^{d}_w \quad \forall d \in \mathcal{D}, i \in V_L, w \in V_D, i \prec w, \tag{3.106}$$

$$-M(1-\tilde{y}^d_{wj}) + s^d_w + \bar{t}_{wj} + \bar{t}_r \leq r^d_j \quad \forall d \in \mathcal{D}, w \in V_D, j \in V_R, i \prec w, \tag{3.107}$$

$$r^d_i \leq s^d_i \quad \forall d \in \mathcal{D}, i \in V, \tag{3.108}$$

$$s^d_i \leq s_i \quad \forall d \in \mathcal{D}, i \in V, \tag{3.109}$$

$$a_j + \bar{t}_l \sum_{\substack{w \in V_D, \\ j \prec w}} y^d_{jw} + \bar{t}_r \sum_{\substack{w \in V_D, \\ w \prec j}} \tilde{y}^d_{wj} \leq r^d_j \quad \forall d \in \mathcal{D}, j \in V_R, \tag{3.110}$$

$$M(\tilde{y}^d_{wj} - 1) + (r^d_j - s^d_w) + \sum_{\substack{w \in V_D, \\ i \prec w}} (\bar{t}_{iw} y^d_{iw}) \leq \mathcal{E}_t \quad \forall k \in \mathcal{K}, d \in \mathcal{D}, i \in V_L, w \in V_D, \; j \in V_R, i \prec w \prec j. \tag{3.111}$$

3.C Endurance: Limited Flight Distance vs. Limited Flight Time

Within the research community, there is no consensus on the operational constraints that should apply to drones. In particular, this concerns the possibility of *cyclic* drone operations (refer to Table 3.2.1) and the way the *endurance* of drones is modelled. Some suggest the application of range-constrained drones (see, e. g., (Agatz et al., 2018; Di Puglia Pugliese and Guerriero, 2017)), while others propose time-constrained drones instead (see, e. g., (Ha et al., 2018; Murray and Chu, 2015)). In order to bridge this gap, we introduce the following proposition, which states that there is no problematic difference between the two mentioned cases, if the drone is permitted to delay its departure after serving a customer. Before attending to the proposition, we introduce the following constraints, which can be used if a drone must not immediately depart after serving a customer:

$$M(y^{kd}_{iwj} - 1) + s^{kd}_i + (\bar{t}_l + \bar{t}_{iw}) \leq r^{kd}_w \quad \forall k \in \mathcal{K}, d \in \mathcal{D}, i \in V_L, w \in V_D, \; j \in V_R, |\{i,w,j\}| = 3, \tag{3.112}$$

$$M(y^{kd}_{iwj} - 1) + s^{kd}_w + (\bar{t}_{wj} + \bar{t}_r) \leq r^{kd}_j \quad \forall k \in \mathcal{K}, d \in \mathcal{D}, i \in V_L, w \in V_D, \; j \in V_R, |\{i,w,j\}| = 3, \tag{3.113}$$

$$M(y^{kd}_{iwj} - 1) + \bar{t}_l + \bar{t}_{iw} + (r^{kd}_j - s^{kd}_w) \leq \mathcal{E}_t \quad \forall k \in \mathcal{K}, d \in \mathcal{D}, i \in V_L, w \in V_D, \; j \in V_R, |\{i,w,j\}| = 3, \tag{3.114}$$

$$(\bar{t}_l + \bar{t}_{iw} + \bar{t}_{wi} + \bar{t}_r)\, y^{kd}_{iwi} \leq \mathcal{E}_t \quad \forall k \in \mathcal{K}, d \in \mathcal{D}, i \in V_L, w \in V_D, i \neq w. \tag{3.115}$$

More precisely, constraints (3.112)–(3.113) might serve as a replacement for constraints (3.20). Through these constraints, we explicitly account for the arrival time r^{kd}_w of a drone d at $w \in V_D$ and its earliest possible release time r^{kd}_j after its return flight towards $j \in V_R$. Given a maximum flight time $\mathcal{E}_t$, the endurance constraints are enforced by (3.114) and (3.115) for acyclic and cyclic operations, respectively, and might serve as a replacement for

constraints (3.24). Note that similar constraints were also used by Murray and Chu (2015) (refer to Appendix 3.B). Due to constraints (3.21), in principle, the departure time from w, i. e., s_w^{kd}, can be arbitrarily larger than r_w^{kd}. In the following proposition, we address the relation between distance- and time-constrained endurance assumptions.

Proposition 3.5. If a drone can remain stationary at a customer location for a sufficient amount of time, then distance- and time-constrained endurance assumptions are equivalent.

Proof. Assume that a drone d is retrieved at some vertex j by truck k after an operation (i, w, j). We distinguish two cases: (i) either the drone arrives at the same time as (or after) the truck, (ii) or the drone arrives before the truck.

First, we assume that the drone arrives at the same time as or after the truck, i. e., $s_w^{kd} + \overline{t}_{wj} + \overline{t}_r \geq a_j^k + \overline{t}_r$. This case is not problematic. The reason is the fact that, as the truck is already at vertex j, the drone does not need to hover. More precisely, to perform the operation (i, w, j), the drone is in flight for exactly the following amount of time units:

$$(\overline{t}_l + \overline{t}_{iw}) + (r_j^{kd} - s_w^{kd}) = (\overline{t}_l + \overline{t}_{iw} + \overline{t}_{wj} + \overline{t}_r). \tag{3.116}$$

Given its average velocity $\overline{v}$, the drone covers $(\overline{t}_l + \overline{t}_{iw} + \overline{t}_{wj} + \overline{t}_r) \cdot \overline{v} = \overline{d}_{iw} + \overline{d}_{wj}$ distance units during the operation.

Next, we assume that the drone arrives before the truck, i. e., $s_w^{kd} + \overline{t}_{wj} + \overline{t}_r < a_j^k + \overline{t}_r$. Here, if a drone cannot safely land, it would need to continue hovering for a time $\delta > 0$ that satisfies the following condition:

$$s_w^{kd} + \overline{t}_{wj} + \overline{t}_r + \delta = a_j^k + \overline{t}_r. \tag{3.117}$$

Hovering for δ time units would require a continuous power supply approximately equal to that during straight flight (Dorling et al., 2017). However, the return flight from w can simply be delayed such that the expected arrival time of the drone and truck match. More precisely, it is always feasible to delay the departure time s_w^{kd} from w by the amount δ such that the arrival time of the truck is matched and hovering becomes unnecessary, i. e., $s_w^{kd} + \overline{t}_{wj} + \overline{t}_r = a_j^k + \overline{t}_r$. Note that the makespan remains unaffected. Consequently, analogously to the first case, the drone is in flight for exactly $(\overline{t}_l + \overline{t}_{iw} + \overline{t}_{wj} + \overline{t}_r)$ time units and covers $(\overline{t}_l + \overline{t}_{iw} + \overline{t}_{wj} + \overline{t}_r) \cdot \overline{v} = \overline{d}_{iw} + \overline{d}_{wj}$ distance units during flight. As a result, let us replace $\mathcal{E}_t \cdot \overline{v} \leftarrow \mathcal{E}$ as well as:

$$(\overline{t}_l + \overline{t}_{iw}) \cdot \overline{v} \leftarrow \overline{d}_{iw} \text{ and } (r_j^{kd} - s_w^{kd}) \cdot \overline{v} = (\overline{t}_{wj} + \overline{t}_r) \cdot \overline{v} \leftarrow \overline{d}_{wj}.$$

After integrating these transformations into (3.114)–(3.115), we obtain the following inequalities:

$$M(y_{iwj}^{kd} - 1) + \overline{d}_{iw} + \overline{d}_{wj} \leq \mathcal{E} \quad \forall k \in \mathcal{K}, d \in \mathcal{D}, i \in V_L, w \in V_D, j \in V_R, |\{i, w, j\}| = 3, \tag{3.118}$$

$$(\overline{d}_{iw} + \overline{d}_{wi})\, y_{iwi}^{kd} \leq \mathcal{E} \quad \forall k \in \mathcal{K}, d \in \mathcal{D}, i \in V_L, w \in V_D, i \neq w. \tag{3.119}$$

Note that these inequalities are functionally equivalent to (3.24), which is the desired conclusion of Proposition 3.5. ■

As a concluding remark, notably, the concept of drones that may wait at customer locations is already implicitly present in some works that consider time-constrained drones (see, e. g., (Murray and Chu, 2015)).

3.D Detailed Numerical Results

Table 3.D.1 contains the results from solving small instances with 10 customers through Gurobi Optimizer 8.1.0 (Gurobi Optimization, LLC, 2023) by using MILP (3.1)–(3.27). Moreover, Tables 3.D.2 shows the results when adding VIEQs (3.30) and (3.32) to the model. Finally, in Table 3.D.3, the symmetry breaking (3.34) and knapsack inequalities (3.37) and (3.38) were added to the formulation. Each table shows the relative savings Δ (see, Equation (3.80)), the MIP gap, the runtime t, the number of acyclic (#a) and cyclic (#c) operations, respectively, as average values as well as the number of cases in which an optimal solution was found within the runtime limit.

Next, Tables 3.D.4 and 3.D.5 show the results from solving the small instances through the matheuristic (see Algorithm 3.1) using the three-index formulation (MILP (3.39)–(3.54)) and two-index formulation (MILP (3.61)–(3.78)) for solving the mathematical program, respectively. In these tables, the relative savings Δ, the time until the final solution is found (t_{best}), the number of changes to the incumbent solution n, and the number of operations are shown as average values. Furthermore, the column labeled "$\leq$" highlights how often the matheuristic found a solution with a better or equal objective value to the feasible (in some cases optimal) solutions returned by the MILP solver after runtime (refer to Table 3.D.3). Recall that the matheuristic ran five times for each problem instance; therefore, for each optimal solution in Table 3.D.3, the matheuristic might reach this solution up to five times.

The results to the instances of Murray and Chu (2015) are shown in Table 3.D.6. In this table, the BKSs were taken from (Ha et al., 2018; Murray and Chu, 2015). The

objective value of the MILP and the best solution value after five runs of the matheuristic are then shown relative to the BKS. These results were obtained by adjusting all MILP formulations according to Appendix 3.B.

Finally, Tables 3.D.7–3.D.12 show the results achieved by solving instances with 20, 50, and 100 customers through matheuristic by solving the MILP through the three- and two-index formulation, respectively. In these tables, as we have no information on any optimal values, the Δ columns are calculated according to Equation (3.82). Furthermore, the share of acyclic and cyclic operations are shown as percentages, rather than absolute values.

Table 3.D.1: MILP solver results on instances with 10 vertices.

$\lvert\mathcal{D}\rvert$	α	$\mathcal{E}_\mathrm{r}$	$\lvert\mathcal{K}\rvert = 1$						$\lvert\mathcal{K}\rvert = 2$					
			Δ	Gap	t	#a	#c	Opt.	Δ	Gap	t	#a	#c	Opt.
1	1	0.2	0.1%	10.0%	315.7	0.2	0.0	9	0.1%	49.7%	687.1	0.2	0.3	5
		0.4	1.5%	70.0%	779.5	0.8	0.0	3	0.3%	90.0%	881.6	1.1	0.0	1
		0.6	2.6%	100.0%	900.4	2.1	0.0	0	1.2%	100.0%	900.3	1.7	0.0	0
		1	3.1%	100.0%	900.4	2.2	0.0	0	-1.3%	100.0%	901.3	1.1	0.2	0
		1.5	-1.2%	100.0%	900.3	1.1	0.0	0	-0.3%	100.0%	911.2	1.9	0.1	0
		2	-2.4%	100.0%	901.1	1.2	0.1	0	-0.7%	100.0%	903.7	1.5	0.1	0
	2	0.2	0.9%	0.0%	337.7	0.4	0.5	10	0.2%	40.0%	635.9	0.1	0.5	6
		0.4	3.5%	57.1%	730.1	1.9	0.2	4	1.0%	100.0%	900.8	1.4	0.5	0
		0.6	9.5%	96.2%	900.3	3.5	0.1	0	4.6%	100.0%	900.6	1.8	1.1	0
		1	14.1%	100.0%	901.0	4.0	0.6	0	7.2%	100.0%	900.3	2.4	2.5	0
		1.5	15.8%	100.0%	902.4	3.2	0.7	0	20.6%	100.0%	904.4	3.4	1.8	0
		2	14.9%	100.0%	900.2	3.0	1.1	0	20.0%	100.0%	911.7	3.6	1.5	0
	3	0.2	1.4%	0.0%	291.6	0.4	0.6	10	0.3%	41.7%	676.1	0.1	0.5	5
		0.4	4.2%	80.0%	808.4	1.8	0.6	2	1.6%	90.3%	900.7	1.0	0.7	0
		0.6	10.8%	100.0%	900.4	3.4	0.1	0	6.3%	100.0%	900.4	2.0	2.1	0
		1	19.6%	100.0%	900.3	3.7	1.5	0	15.4%	100.0%	901.5	2.6	2.7	0
		1.5	24.8%	100.0%	903.4	3.5	1.8	0	34.3%	100.0%	912.2	3.2	2.8	0
		2	25.1%	100.0%	900.3	3.7	1.1	0	35.8%	100.0%	909.4	3.3	2.7	0
Average			8.2%	78.5%	781.9	2.2	0.5	38/180	8.1%	89.5%	863.3	1.8	1.1	17/180
1	1	0.2	0.1%	20.0%	371.9	0.4	0.0	8	0.0%	60.0%	752.2	0.2	0.2	4
		0.4	1.7%	80.0%	831.6	1.4	0.0	2	0.3%	100.0%	901.3	1.2	0.1	0
		0.6	3.3%	100.0%	900.4	3.1	0.0	0	0.3%	100.0%	901.6	1.6	0.6	0
		1	4.3%	100.0%	901.2	3.9	0.7	0	1.6%	100.0%	906.5	2.7	1.5	0
		1.5	6.7%	100.0%	900.3	4.1	0.3	0	4.3%	100.0%	900.9	3.5	2.5	0
		2	2.6%	100.0%	904.1	3.9	0.8	0	4.9%	100.0%	900.8	3.5	2.3	0
	2	0.2	0.9%	18.6%	391.1	0.4	0.5	8	0.2%	38.7%	620.7	0.1	0.5	6
		0.4	3.7%	76.4%	824.7	2.3	0.2	2	1.1%	86.1%	873.9	1.4	1.0	1
		0.6	10.1%	100.0%	900.4	4.1	0.5	0	4.3%	100.0%	900.5	2.2	1.7	0
		1	20.4%	100.0%	901.4	4.8	1.0	0	15.3%	100.0%	905.6	2.7	3.2	0
		1.5	30.8%	100.0%	900.3	4.6	2.2	0	31.5%	100.0%	901.1	3.8	3.1	0
		2	34.2%	100.0%	900.4	4.2	1.9	0	41.0%	100.0%	901.1	4.7	2.6	0
	3	0.2	1.4%	10.0%	358.3	0.4	0.6	9	0.3%	48.0%	720.3	0.3	0.6	5
		0.4	4.6%	90.0%	896.6	2.0	0.9	1	2.1%	91.9%	901.1	1.0	1.5	0
		0.6	12.2%	100.0%	900.4	3.9	0.7	0	6.3%	100.0%	902.4	1.4	3.3	0
		1	26.2%	100.0%	902.7	4.5	2.0	0	20.5%	100.0%	904.2	2.5	3.6	0
		1.5	46.1%	100.0%	902.2	4.4	2.5	0	44.7%	100.0%	901.3	3.5	3.4	0
		2	48.2%	100.0%	900.4	4.2	3.0	0	56.2%	100.0%	901.0	3.6	4.5	0
Average			14.3%	83.1%	804.9	3.1	1.0	30/180	13.1%	90.3%	866.5	2.2	2.0	16/180

Table 3.D.2: MILP with VIEQ (makespan bounds) on instances with 10 vertices.

$\lvert\mathcal{D}\rvert$	α	$\mathcal{E}_r$	$\lvert\mathcal{K}\rvert = 1$						$\lvert\mathcal{K}\rvert = 2$					
			Δ	Gap	t	#a	#c	Opt.	Δ	Gap	t	#a	#c	Opt.
1	2	0.2	0.9%	0.0%	1.6	0.4	0.5	10	0.2%	0.0%	29.3	0.2	0.6	10
		0.4	3.5%	0.0%	7.5	1.9	0.2	10	1.0%	0.0%	113.8	1.0	0.4	10
		0.6	9.5%	0.0%	41.5	3.5	0.1	10	4.8%	2.0%	445.4	2.6	0.7	9
		1	22.8%	3.5%	377.5	4.3	0.2	9	14.1%	58.0%	902.6	3.4	0.7	0
		1.5	26.2%	41.0%	901.7	3.8	0.2	0	23.8%	65.4%	903.4	4.3	0.7	0
		2	26.1%	39.4%	902.3	4.0	0.0	0	24.8%	63.3%	907.8	3.7	0.7	0
	3	0.2	1.4%	0.0%	2.4	0.4	0.6	10	0.3%	0.0%	35.4	0.3	0.4	10
		0.4	4.2%	0.0%	6.8	1.8	0.6	10	1.7%	0.0%	125.5	1.3	0.9	10
		0.6	11.0%	0.0%	34.1	3.5	0.1	10	6.3%	6.2%	554.4	2.3	1.5	7
		1	26.5%	0.0%	247.4	4.5	0.4	10	18.5%	56.8%	901.7	2.7	2.0	0
		1.5	35.0%	32.9%	875.3	4.1	0.5	1	35.2%	61.1%	841.0	3.8	1.7	1
		2	35.0%	35.5%	871.1	4.3	0.5	1	39.3%	62.3%	917.3	4.0	1.7	0
Average			13.1%	14.4%	394.8	2.6	0.2	111/180	10.1%	31.5%	562.6	2.2	0.8	85/180
1	2	0.2	0.1%	0.0%	3.4	0.2	0.0	10	0.1%	0.0%	59.1	0.4	0.1	10
		0.4	1.7%	0.0%	49.5	1.7	0.0	10	0.3%	0.0%	299.2	1.1	0.0	10
		0.6	4.8%	1.1%	412.7	3.0	0.0	8	1.5%	30.6%	810.5	2.5	0.2	2
		1	11.9%	46.4%	903.6	3.7	0.0	0	3.3%	65.5%	911.6	2.4	1.2	0
		1.5	16.2%	51.0%	902.1	3.9	0.0	0	4.2%	74.0%	900.6	3.7	1.1	0
		2	13.4%	53.3%	900.3	3.9	0.0	0	5.7%	74.6%	900.5	4.5	1.0	0
	3	0.2	0.9%	0.0%	2.9	0.4	0.5	10	0.2%	0.0%	58.1	0.4	0.3	10
		0.4	3.7%	0.0%	13.4	2.2	0.3	10	1.2%	0.9%	274.7	1.6	0.8	9
		0.6	11.0%	0.0%	86.4	4.0	0.5	10	5.2%	23.2%	802.5	2.7	1.6	2
		1	27.8%	16.9%	778.3	5.4	0.5	3	15.3%	60.8%	901.0	3.5	1.2	0
		1.5	38.8%	50.0%	900.4	5.6	0.3	0	33.3%	73.8%	900.6	4.8	2.1	0
		2	40.5%	51.0%	900.4	5.5	0.4	0	42.1%	71.3%	900.7	4.8	2.3	0
	3	0.2	1.4%	0.0%	2.8	0.4	0.6	10	0.3%	0.0%	50.4	0.4	0.5	10
		0.4	4.6%	0.0%	14.6	2.1	0.8	10	2.1%	0.9%	238.3	1.4	1.3	9
		0.6	12.8%	0.0%	68.6	3.8	0.8	10	7.0%	26.2%	776.9	2.1	2.4	2
		1	33.5%	0.0%	486.5	5.6	0.5	10	22.0%	69.7%	859.4	3.7	1.8	1
		1.5	51.7%	42.9%	870.1	5.6	0.8	1	45.5%	74.1%	900.6	3.8	3.4	0
		2	53.9%	44.1%	884.5	5.0	1.8	1	59.9%	57.7%	900.5	5.1	2.7	0
Average			18.3%	19.8%	454.5	3.4	0.4	103/180	13.8%	39.1%	635.8	2.7	1.3	65/180

Table 3.D.3: MILP with VIEQ (makespan bounds, symmetry, knapsack) on instances with 10 vertices.

$\lvert\mathcal{D}\rvert$	α	$\mathcal{E}_\mathrm{r}$	$\lvert\mathcal{K}\rvert = 1$						$\lvert\mathcal{K}\rvert = 2$					
			Δ	Gap	t	#a	#c	Opt.	Δ	Gap	t	#a	#c	Opt.
1	1	0.2	0.1%	0.0%	1.3	0.3	0.0	10	0.1%	0.0%	35.5	0.3	0.2	10
		0.4	1.5%	0.0%	4.0	1.0	0.0	10	0.3%	0.0%	125.9	1.0	0.2	10
		0.6	3.4%	0.0%	39.1	2.5	0.0	10	1.3%	7.1%	714.9	1.5	0.1	8
		1	9.5%	0.5%	625.7	2.3	0.0	9	1.8%	60.8%	900.3	1.9	0.2	0
		1.5	11.8%	33.5%	900.4	1.9	0.0	0	3.5%	63.9%	907.6	2.2	0.2	0
		2	10.8%	34.7%	900.2	1.9	0.0	0	3.9%	62.5%	926.3	1.9	0.0	0
	2	0.2	0.9%	0.0%	1.4	0.4	0.5	10	0.2%	0.0%	31.4	0.2	0.4	10
		0.4	3.5%	0.0%	3.2	1.9	0.2	10	1.0%	0.0%	138.3	1.0	0.8	10
		0.6	9.5%	0.0%	15.2	3.5	0.1	10	4.8%	10.3%	560.2	2.3	0.6	6
		1	22.8%	0.0%	210.1	4.2	0.2	10	14.5%	56.4%	870.8	3.1	1.0	1
		1.5	27.3%	27.4%	792.6	4.0	0.2	2	23.6%	64.1%	902.0	3.5	0.7	0
		2	27.5%	33.6%	900.5	3.7	0.4	0	24.1%	66.1%	916.8	3.5	0.8	0
	3	0.2	1.4%	0.0%	1.7	0.4	0.6	10	0.3%	0.0%	42.9	0.2	0.4	10
		0.4	4.2%	0.0%	3.3	1.8	0.6	10	1.7%	0.0%	139.8	1.3	0.8	10
		0.6	11.0%	0.0%	13.5	3.5	0.1	10	6.3%	6.9%	528.5	2.3	1.1	8
		1	26.2%	4.8%	168.3	4.3	0.6	9	17.4%	58.6%	857.6	2.6	2.1	1
		1.5	34.3%	33.7%	869.2	4.1	0.6	2	33.9%	67.4%	900.7	3.7	2.0	0
		2	35.3%	28.7%	790.7	4.0	0.7	2	40.8%	62.8%	915.4	3.7	1.9	0
Average			13.4%	10.9%	346.7	2.5	0.3	124/180	10.0%	32.6%	578.6	2.0	0.8	84/180
1	1	0.2	0.1%	0.0%	1.9	0.4	0.0	10	0.1%	0.0%	66.5	0.4	0.0	10
		0.4	1.7%	0.0%	22.6	1.5	0.0	10	0.3%	0.0%	361.6	0.8	0.1	10
		0.6	4.8%	0.0%	154.3	3.1	0.0	10	1.4%	35.6%	900.8	2.2	0.2	0
		1	14.2%	39.9%	892.1	3.5	0.0	1	3.0%	66.7%	913.1	2.3	1.1	0
		1.5	14.3%	52.5%	900.6	3.9	0.0	0	5.7%	74.4%	900.7	3.9	1.2	0
		2	14.9%	53.3%	900.2	3.8	0.1	0	6.1%	74.6%	900.6	3.8	1.4	0
	2	0.2	0.9%	0.0%	1.7	0.4	0.5	10	0.2%	0.0%	56.3	0.3	0.3	10
		0.4	3.7%	0.0%	5.5	2.2	0.3	10	1.2%	1.2%	319.5	1.0	1.4	9
		0.6	11.0%	0.0%	24.4	4.1	0.4	10	4.9%	36.9%	797.1	2.0	1.9	2
		1	28.3%	25.2%	670.0	5.5	0.2	3	17.9%	59.6%	909.6	3.2	1.6	1
		1.5	39.0%	41.7%	900.3	5.5	0.1	0	34.4%	74.6%	900.6	3.9	2.5	0
		2	41.2%	46.1%	900.8	5.2	0.5	0	43.7%	70.2%	900.6	5.4	1.5	0
	3	0.2	1.4%	0.0%	1.7	0.4	0.6	10	0.3%	0.0%	69.5	0.1	0.7	10
		0.4	4.6%	0.0%	6.1	2.0	0.9	10	2.1%	0.0%	349.1	1.2	1.2	10
		0.6	12.8%	0.0%	30.6	3.7	0.9	10	6.9%	25.3%	730.1	2.1	2.0	4
		1	33.5%	4.3%	413.7	5.6	0.5	7	23.2%	73.6%	848.8	3.8	2.0	1
		1.5	52.7%	23.4%	713.7	5.4	0.9	4	46.6%	74.1%	900.6	4.0	2.7	0
		2	55.7%	20.2%	723.9	5.0	1.3	4	58.9%	65.3%	900.5	4.5	3.0	0
Average			18.6%	17.0%	403.6	3.4	0.4	109/180	14.3%	40.7%	651.4	2.5	1.4	67/180

Table 3.D.4: Matheuristic with the three-index formulation on instances with 10 vertices.

$\lvert\mathcal{D}\rvert$	α	$\mathcal{E}_{\text{rel}}$	$\lvert\mathcal{K}\rvert = 1$						$\lvert\mathcal{K}\rvert = 2$					
			Δ	t_{best}	n	#a	#c	$\leq$	Δ	t_{best}	n	#a	#c	$\leq$
1	1	0.2	0.1%	3.2	4.3	0.2	0.0	50	0.1%	0.4	2.6	0.3	0.0	50
		0.4	1.5%	11.2	5.2	0.9	0.0	50	0.3%	3.3	3.1	1.1	0.0	50
		0.6	3.4%	17.2	5.5	2.1	0.0	49	1.3%	7.9	3.9	1.8	0.0	50
		1	9.5%	25.4	5.8	2.1	0.0	50	3.5%	13.5	4.4	2.3	0.0	45
		1.5	12.7%	34.4	5.9	2.0	0.0	50	5.8%	7.1	3.8	2.4	0.0	50
		2	12.7%	27.1	5.0	2.0	0.0	49	7.2%	17.9	4.1	2.2	0.0	50
	2	0.2	0.9%	6.1	5.3	0.4	0.5	50	0.2%	1.0	3.0	0.4	0.3	50
		0.4	3.5%	8.5	5.6	1.9	0.2	50	1.0%	5.0	3.1	1.4	0.4	50
		0.6	9.5%	10.2	5.6	3.5	0.1	50	4.6%	10.9	5.1	2.8	0.6	45
		1	22.3%	24.6	5.7	4.2	0.1	42	16.5%	28.9	6.1	3.7	0.4	50
		1.5	27.0%	36.4	6.5	4.0	0.2	43	23.4%	27.4	5.2	4.0	1.0	30
		2	27.4%	36.1	6.5	4.0	0.2	38	23.4%	20.1	5.1	3.7	1.1	27
	3	0.2	1.4%	9.8	5.6	0.4	0.6	50	0.3%	1.3	3.0	0.4	0.4	50
		0.4	4.2%	14.0	6.2	1.8	0.6	50	1.6%	3.0	3.8	1.4	0.6	45
		0.6	10.9%	16.1	6.0	3.5	0.1	50	6.1%	10.0	5.0	2.6	1.0	45
		1	25.9%	40.5	7.8	4.3	0.8	43	19.2%	23.8	6.0	3.5	1.4	41
		1.5	32.2%	33.1	6.4	3.9	1.6	22	31.6%	20.4	5.9	3.3	2.7	18
		2	32.0%	41.7	7.2	3.7	1.6	14	32.8%	22.3	6.3	3.1	2.9	0
Average			13.2%	22.0	5.9	2.5	0.4	800/900	9.9%	12.5	4.4	2.3	0.7	746/900
2	1	0.2	0.1%	3.0	4.4	0.2	0.0	50	0.1%	0.6	2.1	0.4	0.0	50
		0.4	1.7%	7.0	5.3	1.5	0.0	50	0.3%	2.1	3.1	1.2	0.0	50
		0.6	4.7%	24.3	6.1	2.8	0.0	49	1.6%	14.2	3.9	2.7	0.0	45
		1	14.5%	34.8	5.9	3.7	0.0	48	3.9%	17.3	4.5	3.7	0.1	45
		1.5	19.5%	39.3	5.0	3.4	0.0	50	6.3%	23.4	3.7	4.3	0.3	50
		2	19.8%	33.9	4.2	3.3	0.1	50	7.3%	20.7	3.4	4.4	0.0	50
	2	0.2	0.9%	7.6	4.9	0.4	0.5	50	0.2%	2.4	3.0	0.4	0.2	50
		0.4	3.7%	11.1	5.3	2.2	0.3	50	1.1%	11.1	3.3	1.6	0.6	50
		0.6	11.0%	26.2	6.7	4.0	0.5	50	5.2%	9.1	4.4	3.1	1.1	50
		1	28.6%	29.5	6.1	5.7	0.3	49	21.0%	32.9	6.6	4.3	1.0	45
		1.5	39.5%	41.8	6.4	5.5	0.6	33	33.7%	23.9	5.6	4.8	1.6	24
		2	40.2%	36.4	5.9	5.4	0.7	19	38.1%	30.3	5.3	5.2	0.8	10
	3	0.2	1.4%	11.1	5.1	0.4	0.6	50	0.3%	3.4	3.0	0.4	0.3	50
		0.4	4.6%	12.9	6.1	2.0	0.9	50	1.8%	5.4	3.5	1.6	0.7	45
		0.6	12.8%	24.9	6.8	3.8	0.8	50	6.9%	13.7	4.7	3.0	1.4	50
		1	33.1%	37.5	7.1	5.5	0.6	46	23.1%	23.0	5.9	4.2	1.5	45
		1.5	49.2%	45.9	7.4	4.2	2.8	6	47.2%	22.5	5.3	4.0	2.5	40
		2	51.5%	47.6	6.7	4.2	2.6	5	51.8%	32.7	5.7	4.1	2.5	5
Average			18.7%	26.4	5.9	3.2	0.6	755/900	13.9%	16.0	4.3	3.0	0.8	754/900

Table 3.D.5: Matheuristic with the two-index formulation on instances with 10 vertices.

$\lvert\mathcal{D}\rvert$	α	$\mathcal{E}_{\text{rel}}$	$\lvert\mathcal{K}\rvert = 1$						$\lvert\mathcal{K}\rvert = 2$					
			Δ	t_{best}	n	#a	#c	$\leq$	Δ	t_{best}	n	#a	#c	$\leq$
1	1	0.2	0.1%	5.3	4.9	0.3	0.0	50	0.1%	0.6	2.3	0.4	0.1	50
		0.4	1.5%	11.1	5.1	1.4	0.0	50	0.3%	1.6	3.0	1.4	0.0	50
		0.6	3.4%	14.4	5.4	2.2	0.0	49	1.3%	7.6	3.8	2.3	0.0	47
		1	9.5%	27.7	5.7	2.1	0.0	50	4.8%	21.9	5.0	2.5	0.0	50
		1.5	12.6%	37.0	5.5	1.9	0.0	49	5.8%	17.8	4.3	2.4	0.0	50
		2	12.7%	38.0	5.2	2.1	0.0	47	7.2%	22.0	5.1	2.1	0.0	50
	2	0.2	0.9%	8.5	5.1	0.4	0.5	49	0.2%	1.2	3.1	0.4	0.3	50
		0.4	3.5%	7.9	5.9	1.9	0.2	50	1.0%	1.1	3.7	1.4	0.4	45
		0.6	9.5%	17.9	6.9	3.5	0.1	50	4.6%	11.0	4.8	2.7	0.7	43
		1	22.2%	32.8	6.1	4.2	0.1	36	16.5%	20.6	6.3	4.1	0.3	49
		1.5	27.0%	42.6	6.5	3.9	0.3	30	23.5%	27.0	5.3	3.8	1.2	28
		2	27.4%	45.8	7.0	4.0	0.2	28	23.6%	21.2	5.3	3.7	1.2	26
	3	0.2	1.4%	15.8	6.0	0.4	0.6	50	0.3%	2.2	3.1	0.4	0.5	50
		0.4	4.2%	17.3	6.2	1.8	0.6	50	1.6%	2.7	3.9	1.4	0.7	45
		0.6	11.0%	14.6	6.0	3.4	0.2	46	6.0%	4.7	5.4	2.6	0.9	40
		1	25.5%	39.1	7.4	4.3	0.6	34	18.9%	22.0	6.2	3.7	1.0	37
		1.5	31.2%	41.4	6.8	4.0	0.9	15	30.6%	13.0	6.2	3.7	1.8	20
		2	31.1%	46.8	6.9	4.2	0.8	3	31.1%	18.1	5.7	3.3	1.8	0
Average			13.1%	25.8	6.0	2.6	0.3	736/900	9.9%	12.0	4.6	2.3	0.6	730/900
2	1	0.2	0.1%	5.8	5.1	0.2	0.0	50	0.1%	0.9	2.3	0.4	0.2	50
		0.4	1.7%	16.2	5.0	1.7	0.0	50	0.3%	3.4	3.1	1.6	0.0	50
		0.6	4.7%	35.1	6.4	3.0	0.0	48	1.6%	12.8	4.4	3.2	0.1	42
		1	14.5%	38.5	5.9	3.7	0.0	47	4.6%	24.0	5.0	4.1	0.1	48
		1.5	19.5%	45.3	4.8	3.4	0.0	50	6.4%	23.9	5.0	4.5	0.3	45
		2	19.8%	42.1	4.9	3.2	0.0	50	7.3%	39.8	4.8	4.3	0.2	50
	2	0.2	0.9%	6.0	5.3	0.4	0.5	50	0.2%	0.9	2.9	0.4	0.4	50
		0.4	3.7%	14.9	5.6	2.3	0.2	50	1.1%	3.8	3.7	1.4	0.8	45
		0.6	11.0%	21.1	6.5	4.1	0.4	50	5.2%	5.0	4.8	3.1	1.2	45
		1	28.5%	43.9	6.6	5.7	0.3	47	21.0%	30.6	6.1	4.4	0.9	45
		1.5	39.6%	52.3	6.2	5.3	0.6	34	33.7%	32.2	5.9	4.8	1.6	21
		2	40.4%	42.0	6.0	5.6	0.5	19	38.3%	25.0	4.9	5.1	0.9	10
	3	0.2	1.4%	7.4	5.2	0.4	0.6	50	0.3%	1.5	2.9	0.4	0.5	50
		0.4	4.6%	15.7	5.9	2.0	0.9	50	1.8%	3.9	3.9	1.4	1.0	45
		0.6	12.8%	27.0	7.0	3.8	0.8	50	6.7%	10.3	4.9	3.0	1.4	45
		1	33.0%	38.3	7.1	5.5	0.6	45	23.2%	23.2	5.8	4.2	1.5	40
		1.5	48.0%	52.2	6.8	4.9	1.7	3	47.2%	24.4	6.2	3.8	2.6	40
		2	50.4%	52.7	6.9	4.6	1.6	0	51.9%	28.0	5.2	4.3	2.3	4
Average			18.6%	30.9	6.0	3.3	0.5	743/900	13.9%	16.3	4.5	3.0	0.9	725/900

Table 3.D.6: Detailed results on the FSTSP instances of Murray and Chu (2015) with 11 vertices.

Drone Speed (mph)	Instance	$\mathcal{E}_t$ = 20 minutes						$\mathcal{E}_t$ = 40 minutes					
		BKS	MILP			Matheuristic		BKS	MILP			Matheuristic	
			BKS_{rel}	Gap	t	BKS_{rel}	t_{best}		BKS_{rel}	Gap	t	BKS_{rel}	t_{best}
15	37v1	56.5	100.00%	0.0%	116.0	100.00%	12.9	50.6	100.00%	17.0%	900.0	100.00%	27.6
	37v2	53.2	100.00%	0.0%	62.0	100.00%	36.3	47.3	100.00%	0.0%	196.1	100.00%	64.8
	37v3	53.7	100.00%	0.0%	120.2	100.00%	16.7	53.7	100.00%	30.4%	900.0	100.00%	98.2
	37v4	67.5	100.00%	0.0%	510.5	100.00%	4.7	66.5	100.83%	39.4%	900.0	100.00%	83.3
	40v1	49.4	100.00%	0.0%	296.4	100.00%	12.9	46.9	100.00%	9.8%	900.0	100.00%	22.9
	40v2	50.7	100.00%	0.0%	69.1	100.00%	41.5	46.4	102.41%	11.0%	900.0	100.00%	43.2
	40v3	56.1	100.00%	0.0%	325.7	100.00%	50.3	53.9	104.02%	24.3%	900.0	100.00%	11.7
	40v4	69.9	100.00%	0.0%	891.1	100.00%	81.4	67.9	100.71%	37.1%	900.0	100.71%	8.1
	43v1	69.6	100.00%	0.0%	55.3	100.00%	63.7	55.5	100.00%	11.6%	900.0	100.00%	36.0
	43v2	72.1	100.00%	0.0%	54.8	100.00%	56.4	58.1	100.00%	15.5%	900.0	100.00%	37.9
	43v3	77.3	100.00%	0.0%	45.1	100.00%	43.4	69.2	102.43%	29.9%	900.0	100.05%	47.5
	43v4	90.1	100.00%	0.0%	521.1	100.00%	14.6	82.7	101.15%	37.5%	900.0	100.00%	89.5
25	37v5	47.5	108.29%	30.1%	900.0	106.52%	12.6	45.8	104.16%	23.3%	900.0	110.29%	50.9
	37v6	45.1	104.80%	27.9%	900.0	104.80%	39.0	44.6	106.07%	33.7%	900.0	**97.76%**	106.2
	37v7	49.6	100.00%	26.6%	900.0	100.00%	7.7	46.6	102.51%	30.1%	900.0	100.00%	93.7
	37v8	62.4	100.00%	35.3%	900.0	100.00%	57.7	59.6	101.34%	36.7%	900.0	**99.67%**	43.3
	40v5	43.5	100.00%	0.0%	139.5	102.51%	22.1	43.5	100.00%	0.0%	162.0	101.19%	19.5
	40v6	44.1	**99.71%**	0.0%	113.3	**99.71%**	15.3	44.1	**99.40%**	0.0%	153.1	**99.40%**	19.0
	40v7	49.5	**99.90%**	0.0%	181.1	**99.90%**	77.3	49.2	100.54%	14.8%	900.0	100.00%	68.3
	40v8	62.3	100.73%	25.1%	900.0	**99.92%**	82.8	62.0	101.28%	27.5%	900.0	**99.95%**	46.5
	43v5	53.1	102.86%	12.2%	900.0	104.60%	55.9	52.1	104.44%	28.9%	900.0	100.63%	77.7
	43v6	55.2	100.00%	23.1%	900.0	101.01%	54.2	52.3	105.05%	28.9%	900.0	100.00%	57.9
	43v7	64.4	100.00%	0.0%	240.2	100.00%	63.6	60.7	100.00%	25.9%	900.0	100.00%	79.3
	43v8	77.2	100.00%	25.7%	900.0	100.00%	98.0	73.7	**98.97%**	36.5%	900.0	**98.97%**	73.5
35	37v9	42.6	105.80%	29.3%	900.0	105.29%	56.1	42.4	101.25%	23.2%	900.0	105.71%	37.9
	37v10	41.9	101.76%	32.6%	900.0	**99.57%**	44.7	40.9	104.25%	24.1%	900.0	102.01%	34.3
	37v11	42.9	100.00%	21.3%	900.0	100.00%	28.3	42.9	100.00%	27.6%	900.0	100.00%	72.2
	37v12	56.7	**98.24%**	0.0%	575.0	**98.24%**	34.5	55.7	100.00%	35.3%	900.0	100.00%	70.3
	40v9	42.5	100.00%	0.0%	211.6	100.00%	16.1	42.5	100.00%	0.0%	229.2	100.00%	40.9
	40v10	43.1	100.00%	0.0%	122.3	100.00%	44.3	43.1	100.00%	0.0%	204.2	100.00%	26.2
	40v11	49.2	100.00%	0.0%	211.4	100.00%	28.2	49.2	100.00%	0.0%	204.4	100.00%	80.0
	40v12	62.0	100.00%	24.9%	900.0	100.00%	31.6	62.0	100.00%	25.0%	900.0	100.00%	40.1
	43v9	46.9	**97.87%**	17.9%	900.0	**97.87%**	108.8	46.9	102.97%	24.0%	900.0	100.00%	40.9
	43v10	47.9	100.00%	17.3%	900.0	**97.91%**	48.3	47.9	101.28%	21.2%	900.0	**97.91%**	64.0
	43v11	56.4	101.54%	20.5%	900.0	100.00%	75.6	56.4	101.54%	19.7%	900.0	100.00%	78.0
	43v12	69.2	100.00%	39.1%	900.0	100.00%	111.4	69.2	102.54%	43.0%	900.0	100.00%	59.1
	Average		100.60%	11.4%	535.0	100.50%	45.8	53.7	101.37%	22.0%	781.9	100.40%	54.2

Table 3.D.7: Matheuristic with the three-index formulation on instances with 20 vertices.

$\lvert\mathcal{D}\rvert$	α	$\mathcal{E}_r$	$\lvert\mathcal{K}\rvert = 1$				$\lvert\mathcal{K}\rvert = 2$				$\lvert\mathcal{K}\rvert = 3$			
			Δ	#a	#c	t_{best}	Δ	#a	#c	t_{best}	Δ	#a	#c	t_{best}
1	2	0.2	2.8%	7.9%	0.7%	38.4	0.8%	6.7%	0.0%	47.9	0.2%	5.5%	0.2%	32.5
		0.4	7.8%	19.7%	0.3%	26.8	3.7%	16.2%	0.0%	76.8	2.8%	16.3%	0.5%	46.7
		0.6	13.8%	22.7%	0.7%	31.4	7.3%	18.3%	0.2%	96.3	4.4%	20.8%	0.5%	34.9
		1	16.4%	20.4%	0.3%	20.3	11.7%	19.9%	0.1%	94.2	5.4%	24.6%	0.3%	70.0
		0.2	7.8%	11.9%	6.1%	47.6	2.1%	12.9%	3.1%	96.7	1.7%	11.8%	4.6%	21.4
		0.4	18.6%	33.8%	5.1%	52.7	7.5%	33.2%	3.8%	107.9	5.1%	29.2%	3.8%	42.8
		0.6	27.1%	38.3%	5.1%	60.3	16.8%	38.5%	3.5%	115.7	9.8%	33.1%	6.2%	75.4
		1	34.6%	37.5%	6.3%	68.1	25.0%	41.3%	3.8%	96.3	18.1%	40.8%	4.5%	108.0
		0.2	9.4%	12.5%	10.3%	39.7	3.2%	11.4%	6.7%	70.0	1.9%	12.1%	5.9%	30.6
		0.4	23.0%	33.2%	10.6%	74.6	9.2%	32.1%	9.6%	98.7	6.0%	27.2%	8.0%	48.9
		0.6	33.0%	38.6%	10.4%	75.0	19.5%	35.8%	10.3%	98.6	12.5%	31.4%	13.8%	53.5
		1	42.7%	37.4%	13.1%	101.0	26.9%	39.1%	14.5%	89.7	21.7%	36.0%	19.8%	94.6
2	2	0.2	3.4%	12.2%	2.2%	26.2	0.7%	10.1%	0.2%	47.8	0.7%	5.2%	0.0%	33.0
		0.4	12.3%	33.9%	0.9%	29.1	5.1%	28.2%	0.3%	105.8	3.2%	24.4%	0.6%	55.6
		0.6	22.4%	37.5%	0.7%	65.0	10.0%	35.7%	0.2%	104.7	3.4%	33.8%	1.8%	47.9
		1	25.7%	34.5%	0.2%	119.3	16.2%	38.9%	1.8%	86.8	4.7%	40.8%	2.4%	73.5
		0.2	5.7%	15.4%	5.4%	29.3	1.7%	13.9%	4.5%	75.2	2.2%	11.9%	4.9%	37.7
		0.4	22.5%	41.9%	4.8%	53.7	9.5%	40.4%	2.0%	79.8	7.2%	35.1%	4.8%	67.1
		0.6	33.9%	54.2%	4.4%	58.0	20.3%	48.9%	4.8%	90.1	14.2%	44.8%	7.5%	64.9
		1	46.6%	53.5%	5.2%	80.9	32.8%	53.2%	8.7%	101.0	24.3%	50.9%	9.6%	90.9
		0.2	9.0%	14.2%	11.1%	43.2	3.0%	13.1%	7.9%	71.2	2.7%	12.1%	6.6%	66.2
		0.4	28.1%	40.7%	8.9%	75.3	11.5%	38.1%	7.1%	89.5	8.2%	35.5%	7.6%	60.4
		0.6	40.7%	52.2%	9.2%	75.6	24.8%	44.9%	14.4%	112.9	17.1%	41.9%	13.2%	48.1
		1	56.5%	50.3%	17.7%	125.2	37.2%	49.7%	21.5%	99.4	29.4%	43.4%	26.7%	87.2
Average			22.7%	31.4%	5.8%	59.0	12.8%	30.0%	5.4%	89.7	8.6%	27.9%	6.4%	58.0

Table 3.D.8: Matheuristic with the two-index formulation on instances with 20 vertices.

$\lvert\mathcal{D}\rvert$	α	$\mathcal{E}_r$	$\lvert\mathcal{K}\rvert = 1$				$\lvert\mathcal{K}\rvert = 2$				$\lvert\mathcal{K}\rvert = 3$			
			Δ	#a	#c	t_{best}	Δ	#a	#c	t_{best}	Δ	#a	#c	t_{best}
1	1	0.2	2.5%	9.8%	0.5%	30.3	0.8%	7.7%	0.1%	50.8	0.4%	8.0%	0.3%	26.4
		0.4	8.2%	20.3%	0.4%	25.0	3.5%	17.8%	0.1%	75.9	3.2%	20.8%	1.1%	47.9
		0.6	12.1%	21.8%	0.3%	24.0	6.7%	19.8%	0.5%	86.6	4.4%	22.6%	1.1%	45.5
		1	16.4%	19.9%	0.0%	17.5	11.5%	20.4%	0.0%	91.1	6.3%	23.5%	0.5%	78.6
	2	0.2	6.2%	13.7%	5.3%	51.8	2.0%	13.2%	2.9%	85.3	1.6%	12.3%	5.7%	16.5
		0.4	16.6%	34.3%	4.1%	54.9	7.2%	33.2%	3.9%	84.3	5.0%	27.4%	4.9%	56.2
		0.6	23.9%	38.5%	5.2%	28.8	16.9%	38.2%	2.9%	98.9	10.1%	31.1%	7.4%	80.9
		1	31.5%	39.1%	5.1%	55.9	25.0%	40.7%	3.7%	101.5	17.6%	38.6%	5.9%	89.5
	3	0.2	10.4%	11.3%	10.7%	64.0	3.1%	11.4%	6.4%	74.1	1.8%	11.7%	5.8%	37.0
		0.4	21.7%	33.3%	9.1%	74.7	8.6%	31.7%	7.5%	98.6	5.2%	26.4%	7.8%	60.3
		0.6	33.0%	38.3%	10.6%	90.6	19.5%	35.8%	8.7%	110.5	12.7%	30.1%	12.1%	72.2
		1	39.6%	38.3%	10.7%	83.8	26.6%	39.8%	10.1%	99.2	21.1%	37.6%	13.1%	107.8
2	1	0.2	3.8%	12.7%	1.6%	34.2	0.6%	10.4%	0.1%	61.1	0.7%	9.4%	1.7%	32.5
		0.4	12.3%	34.0%	0.6%	41.5	4.8%	29.9%	0.0%	90.3	3.3%	30.1%	1.9%	60.1
		0.6	21.6%	38.4%	0.2%	59.8	10.5%	35.5%	0.4%	110.7	3.3%	35.1%	2.2%	54.3
		1	25.8%	33.6%	0.1%	93.0	16.0%	38.0%	1.1%	102.3	5.3%	39.1%	3.1%	95.6
	2	0.2	6.6%	15.1%	7.7%	24.5	2.3%	13.7%	5.5%	68.1	2.0%	10.9%	5.8%	28.5
		0.4	21.0%	42.9%	4.4%	66.0	9.4%	39.5%	3.3%	89.3	6.4%	32.3%	8.6%	59.4
		0.6	33.5%	54.0%	4.3%	46.5	19.6%	46.5%	5.5%	96.3	13.7%	41.4%	8.0%	57.4
		1	45.3%	54.9%	4.4%	101.6	32.4%	52.8%	7.6%	105.2	24.0%	49.8%	10.6%	90.5
	3	0.2	11.7%	14.0%	11.8%	62.6	2.7%	11.8%	8.7%	73.0	2.3%	11.2%	8.4%	43.4
		0.4	26.7%	40.8%	10.0%	59.1	11.5%	38.7%	6.9%	77.9	7.8%	32.9%	10.7%	73.6
		0.6	36.8%	52.5%	8.7%	60.6	23.7%	46.1%	10.3%	99.6	16.7%	40.9%	14.2%	54.2
		1	51.9%	53.4%	13.5%	106.5	35.8%	52.1%	15.1%	119.7	28.5%	47.5%	18.5%	98.5
Average			21.6%	31.9%	5.4%	56.6	12.5%	30.2%	4.6%	89.6	8.5%	27.9%	6.6%	61.1

Table 3.D.9: Matheuristic with the three-index formulation on instances with 50 vertices.

$\lvert\mathcal{D}\rvert$	α	$\mathcal{E}_r$	$\lvert\mathcal{K}\rvert = 1$				$\lvert\mathcal{K}\rvert = 2$				$\lvert\mathcal{K}\rvert = 3$			
			Δ	#a	#c	t_{best}	Δ	#a	#c	t_{best}	Δ	#a	#c	t_{best}
1	1	0.2	6.2%	21.7%	0.2%	67.4	3.0%	19.3%	0.0%	92.5	2.0%	17.9%	0.0%	77.2
		0.4	15.4%	23.7%	0.1%	89.7	7.8%	23.6%	0.0%	102.5	4.1%	24.3%	0.0%	104.3
		0.6	16.4%	22.2%	0.0%	97.7	9.6%	23.1%	0.0%	104.9	5.8%	23.9%	0.1%	114.3
		1	16.3%	22.2%	0.1%	70.2	9.7%	23.1%	0.0%	116.1	5.9%	23.7%	0.1%	134.4
	2	0.2	14.6%	31.4%	3.3%	78.7	8.4%	29.2%	2.2%	96.0	4.4%	26.7%	1.8%	74.3
		0.4	30.4%	40.4%	2.9%	85.6	19.3%	40.6%	1.3%	116.7	12.6%	39.1%	2.0%	112.0
		0.6	30.6%	39.1%	3.0%	112.9	19.9%	41.7%	1.8%	90.3	13.6%	40.7%	2.6%	97.9
		1	29.6%	40.0%	2.8%	91.4	19.8%	41.3%	2.0%	108.4	14.2%	41.4%	2.7%	101.1
	3	0.2	16.9%	30.4%	7.7%	94.4	9.5%	28.2%	5.3%	85.3	5.3%	26.9%	5.1%	71.2
		0.4	33.6%	39.6%	9.1%	99.2	20.2%	40.3%	5.3%	85.7	13.3%	38.7%	6.1%	106.1
		0.6	34.8%	38.2%	11.2%	85.2	21.4%	40.7%	6.8%	101.8	15.2%	38.8%	8.6%	107.9
		1	34.0%	38.5%	10.6%	83.3	21.7%	40.6%	6.7%	105.5	15.7%	39.5%	9.6%	106.9
2	1	0.2	7.8%	32.2%	0.5%	87.0	4.3%	28.5%	0.2%	110.4	1.9%	27.1%	0.3%	83.9
		0.4	21.0%	42.6%	0.6%	90.2	10.7%	42.6%	0.7%	112.4	5.8%	42.0%	0.9%	131.0
		0.6	21.8%	42.3%	0.7%	123.9	12.0%	42.9%	0.9%	140.4	7.3%	43.1%	1.5%	127.4
		1	20.7%	42.5%	0.6%	140.1	12.5%	42.9%	0.7%	173.6	7.8%	44.6%	1.3%	128.6
	2	0.2	16.2%	36.2%	5.0%	78.7	9.8%	34.1%	2.5%	91.5	5.1%	31.5%	3.3%	85.9
		0.4	40.0%	56.8%	2.9%	101.6	24.5%	56.6%	2.0%	135.6	15.2%	52.5%	2.6%	91.5
		0.6	42.6%	56.1%	3.3%	86.0	27.4%	57.8%	2.1%	112.2	19.0%	55.5%	4.2%	92.1
		1	43.3%	56.4%	3.9%	91.1	27.7%	57.5%	2.4%	112.3	20.4%	57.6%	3.9%	109.0
	3	0.2	18.7%	34.4%	8.7%	71.1	10.5%	33.1%	6.2%	94.6	5.7%	30.9%	4.8%	86.8
		0.4	42.9%	56.2%	7.5%	91.5	26.2%	55.6%	5.0%	104.3	17.1%	51.2%	6.5%	87.5
		0.6	47.8%	54.8%	10.8%	133.2	30.0%	56.0%	7.7%	115.6	21.7%	53.4%	10.0%	102.0
		1	47.4%	54.2%	11.2%	91.3	30.6%	56.1%	9.5%	106.7	23.2%	55.7%	11.2%	115.9
Average			27.0%	39.7%	4.4%	93.4	16.5%	39.8%	3.0%	109.0	10.9%	38.6%	3.7%	102.1

Table 3.D.10: Matheuristic with the two-index formulation on instances with 50 vertices.

$\lvert\mathcal{D}\rvert$	α	$\mathcal{E}_r$	$\lvert\mathcal{K}\rvert = 1$				$\lvert\mathcal{K}\rvert = 2$				$\lvert\mathcal{K}\rvert = 3$			
			Δ	#a	#c	t_{best}	Δ	#a	#c	t_{best}	Δ	#a	#c	t_{best}
1	2	0.2	6.1%	21.6%	0.1%	91.5	2.8%	20.5%	0.0%	90.6	1.8%	17.6%	0.1%	112.3
		0.4	14.4%	24.5%	0.0%	91.2	7.2%	25.2%	0.1%	118.7	4.1%	24.6%	0.1%	116.1
		0.6	14.7%	22.4%	0.1%	96.0	8.2%	24.5%	0.0%	101.2	5.3%	24.0%	0.2%	116.5
		1	14.9%	22.2%	0.0%	79.5	8.5%	23.3%	0.2%	113.9	5.3%	24.1%	0.1%	119.1
		0.2	13.9%	30.9%	3.3%	93.1	8.2%	29.3%	2.0%	97.5	4.5%	26.5%	1.6%	76.8
		0.4	29.5%	39.7%	2.9%	81.4	18.9%	40.0%	1.6%	140.4	12.1%	39.0%	1.8%	94.2
		0.6	30.7%	39.4%	3.4%	105.7	19.7%	41.0%	2.1%	108.3	13.4%	39.8%	2.6%	111.6
		1	29.8%	38.9%	3.6%	91.7	19.6%	41.1%	1.9%	109.8	13.9%	40.0%	3.8%	90.8
		0.2	16.5%	30.2%	6.5%	85.0	9.2%	29.3%	4.1%	110.5	5.0%	26.5%	4.7%	82.6
		0.4	32.6%	40.3%	8.1%	85.4	20.1%	40.4%	4.7%	117.9	13.3%	38.2%	6.2%	101.1
		0.6	33.4%	38.8%	9.4%	88.1	21.6%	41.3%	6.4%	114.9	14.7%	39.6%	7.5%	89.6
		1	33.6%	39.3%	8.2%	96.6	21.4%	40.9%	5.9%	122.8	15.4%	40.0%	8.2%	112.5
2	2	0.2	6.5%	31.5%	1.2%	82.7	3.3%	30.0%	0.3%	105.1	1.8%	27.5%	0.7%	107.2
		0.4	18.9%	45.3%	0.9%	115.5	9.2%	44.0%	0.9%	118.7	5.4%	42.1%	1.2%	144.2
		0.6	20.2%	42.1%	0.9%	121.5	11.5%	44.7%	1.0%	136.6	6.8%	44.0%	1.1%	167.9
		1	18.9%	42.7%	0.8%	117.7	11.1%	43.5%	0.9%	158.3	7.4%	45.3%	1.0%	174.0
		0.2	15.6%	35.4%	5.0%	87.6	9.3%	33.8%	2.7%	104.7	4.9%	31.0%	3.4%	91.9
		0.4	39.0%	56.7%	2.2%	115.2	23.3%	56.1%	1.9%	110.8	15.0%	52.0%	2.9%	88.5
		0.6	42.6%	57.1%	3.4%	108.8	26.9%	57.6%	2.0%	112.7	18.7%	55.1%	3.6%	117.4
		1	41.8%	56.0%	3.0%	85.2	27.0%	57.7%	2.3%	122.1	19.8%	56.2%	3.6%	126.0
		0.2	18.6%	33.5%	8.4%	108.8	10.4%	32.9%	5.1%	85.3	5.3%	29.5%	6.7%	98.4
		0.4	41.8%	57.1%	4.6%	107.1	25.1%	54.4%	5.1%	97.8	16.8%	51.4%	5.5%	91.9
		0.6	45.8%	55.7%	8.0%	126.5	29.4%	56.8%	5.9%	105.8	21.0%	52.7%	9.0%	104.0
		1	45.8%	54.7%	9.1%	117.9	29.3%	57.0%	6.6%	103.5	22.6%	55.1%	9.8%	116.6
Average			26.1%	39.8%	3.9%	99.2	15.9%	40.2%	2.7%	112.8	10.6%	38.4%	3.5%	110.5

Table 3.D.11: Matheuristic with the three-index formulation on instances with 100 vertices.

$\lvert\mathcal{D}\rvert$	α	$\mathcal{E}_r$	$\lvert\mathcal{K}\rvert = 1$				$\lvert\mathcal{K}\rvert = 2$				$\lvert\mathcal{K}\rvert = 3$			
			Δ	#a	#c	t_{best}	Δ	#a	#c	t_{best}	Δ	#a	#c	t_{best}
1	1	0.2	8.1%	23.8%	0.1%	4.3	3.9%	23.0%	0.0%	3.9	1.9%	22.3%	0.2%	3.3
		0.4	13.1%	22.0%	0.0%	12.5	7.0%	22.6%	0.1%	8.5	4.3%	23.1%	0.1%	6.9
		0.6	13.0%	22.1%	0.0%	12.1	7.6%	22.3%	0.0%	10.6	4.5%	22.6%	0.1%	8.3
		1	13.1%	22.2%	0.1%	15.2	7.6%	22.4%	0.0%	10.8	5.0%	23.0%	0.0%	8.9
	2	0.2	17.9%	38.3%	2.6%	2.4	10.5%	37.0%	2.1%	5.5	5.6%	35.0%	1.7%	3.9
		0.4	23.7%	39.7%	3.0%	4.7	15.2%	40.5%	2.5%	14.9	9.9%	41.0%	1.8%	6.8
		0.6	23.4%	40.2%	3.1%	5.5	15.4%	40.9%	2.4%	8.4	12.2%	41.3%	2.1%	13.7
		1	23.9%	40.1%	2.7%	5.5	15.9%	40.5%	2.4%	12.6	11.4%	41.2%	2.4%	10.3
	3	0.2	19.4%	37.3%	6.8%	2.4	11.4%	36.6%	5.1%	2.7	6.3%	34.3%	4.3%	4.8
		0.4	25.5%	38.7%	9.2%	4.4	17.5%	40.0%	6.9%	9.7	13.0%	39.9%	6.2%	10.9
		0.6	25.8%	38.5%	9.6%	5.1	17.0%	39.9%	7.4%	10.5	15.2%	39.9%	7.0%	23.3
		1	25.6%	39.2%	9.2%	5.2	17.3%	39.9%	7.5%	6.4	15.0%	40.2%	6.4%	11.4
2	1	0.2	11.1%	41.7%	0.6%	11.4	5.3%	38.4%	0.3%	9.5	2.5%	36.1%	0.5%	7.5
		0.4	17.8%	42.6%	0.9%	52.0	10.4%	42.7%	0.6%	32.4	5.7%	42.8%	0.5%	25.2
		0.6	18.4%	42.3%	0.8%	53.5	11.0%	42.7%	0.7%	40.0	7.3%	42.5%	0.8%	35.8
		1	17.9%	42.2%	0.8%	53.9	10.6%	42.5%	0.6%	47.3	6.8%	43.2%	0.6%	30.4
	2	0.2	22.8%	50.9%	2.2%	2.7	12.9%	48.9%	1.7%	2.8	7.3%	45.6%	1.5%	3.9
		0.4	33.8%	56.3%	3.4%	5.2	26.1%	56.5%	3.0%	20.7	20.4%	56.5%	2.2%	22.2
		0.6	33.9%	56.4%	3.5%	5.8	25.7%	57.3%	2.8%	16.4	24.4%	57.3%	3.1%	30.2
		1	33.5%	56.4%	3.6%	6.2	25.2%	57.4%	2.7%	14.1	25.2%	57.2%	3.1%	33.2
	3	0.2	24.2%	49.3%	5.6%	2.8	14.1%	47.6%	4.3%	3.8	8.7%	44.2%	3.6%	5.8
		0.4	36.4%	54.5%	9.2%	4.2	26.9%	55.7%	7.1%	16.7	24.2%	54.3%	7.3%	24.5
		0.6	36.5%	54.5%	9.8%	5.1	31.6%	54.8%	9.3%	31.3	30.8%	54.9%	8.8%	64.3
		1	36.6%	53.7%	10.3%	5.2	30.7%	55.3%	8.2%	25.8	32.7%	55.2%	9.3%	43.7
Average			23.1%	41.8%	4.1%	12.0	15.7%	41.9%	3.2%	15.2	12.5%	41.4%	3.1%	18.3

Table 3.D.12: Matheuristic with the two-index formulation on instances with 100 vertices.

$\lvert\mathcal{D}\rvert$	α	$\mathcal{E}_r$	$\lvert\mathcal{K}\rvert = 1$				$\lvert\mathcal{K}\rvert = 2$				$\lvert\mathcal{K}\rvert = 3$			
			Δ	#a	#c	t_{best}	Δ	#a	#c	t_{best}	Δ	#a	#c	t_{best}
1	2	0.2	7.2%	24.8%	0.1%	5.6	3.6%	24.1%	0.1%	4.9	1.8%	22.8%	0.4%	4.2
		0.4	11.6%	22.8%	0.0%	14.5	6.4%	23.2%	0.1%	10.9	4.1%	23.0%	0.3%	8.4
		0.6	11.2%	22.3%	0.0%	15.6	6.7%	22.3%	0.1%	12.8	4.3%	22.0%	0.2%	9.9
		1	11.3%	22.2%	0.0%	15.8	6.6%	22.1%	0.0%	13.1	4.3%	22.1%	0.2%	11.4
		0.2	17.6%	38.7%	2.3%	3.1	10.2%	37.4%	1.9%	3.2	5.6%	34.6%	2.7%	2.6
		0.4	23.4%	40.4%	2.7%	5.8	14.9%	40.0%	2.6%	6.3	11.2%	39.3%	3.0%	14.1
		0.6	23.4%	40.2%	2.8%	6.6	15.6%	39.8%	2.3%	10.7	11.0%	39.7%	2.1%	6.8
		1	23.3%	40.1%	3.1%	6.8	15.1%	40.9%	2.0%	8.5	10.9%	40.3%	2.1%	11.0
		0.2	19.1%	37.7%	5.8%	3.1	10.3%	35.7%	5.3%	3.8	6.2%	33.7%	5.8%	7.1
		0.4	25.6%	39.3%	8.4%	5.7	16.2%	39.6%	6.3%	7.4	11.5%	39.4%	6.0%	8.1
		0.6	24.6%	39.4%	8.0%	6.8	16.6%	39.1%	6.5%	11.3	13.8%	39.9%	6.3%	16.0
		1	25.7%	39.4%	7.8%	7.3	18.0%	39.7%	6.7%	8.5	13.2%	39.2%	6.4%	12.4
2	2	0.2	10.5%	41.9%	0.8%	17.4	4.6%	40.0%	0.8%	13.3	2.3%	36.4%	0.9%	11.2
		0.4	16.0%	42.9%	0.9%	50.2	9.7%	42.4%	0.8%	40.6	5.5%	40.7%	0.8%	35.3
		0.6	16.5%	41.7%	1.3%	54.4	9.8%	41.9%	0.7%	43.1	6.1%	39.5%	1.1%	37.7
		1	16.6%	42.4%	1.1%	49.2	9.9%	41.6%	0.7%	44.0	6.5%	41.7%	0.9%	41.0
		0.2	23.3%	50.4%	2.8%	4.1	13.0%	47.6%	2.0%	4.9	7.9%	43.8%	2.5%	5.9
		0.4	33.0%	56.4%	3.0%	9.2	23.2%	56.0%	2.2%	10.7	19.5%	55.1%	3.0%	23.7
		0.6	34.0%	55.7%	3.2%	9.7	23.1%	56.8%	2.5%	11.9	25.0%	56.0%	2.8%	40.8
		1	33.0%	56.9%	2.6%	10.5	24.0%	56.5%	2.4%	16.5	24.9%	56.6%	2.5%	40.9
		0.2	24.7%	49.8%	5.1%	4.0	14.0%	47.3%	4.4%	7.2	7.3%	43.5%	4.5%	3.5
		0.4	36.0%	55.2%	7.7%	7.1	26.1%	55.0%	6.3%	19.9	22.2%	53.9%	5.5%	27.0
		0.6	36.3%	55.0%	7.8%	8.4	30.9%	55.4%	6.9%	26.0	27.6%	55.2%	6.9%	49.0
		1	35.8%	54.5%	8.5%	9.5	31.5%	54.6%	7.5%	32.8	26.5%	54.9%	7.1%	40.8
Average			22.5%	42.1%	3.6%	13.8	15.0%	41.6%	3.0%	15.5	11.6%	40.6%	3.1%	19.5

4 The Traveling Salesman Drone Station Location Problem

Abstract

In this paper, we introduce the *Traveling Salesman Drone Station Location Problem* (TSDSLP). The TSDSLP exhibits features of the Traveling Salesman, Facility Location, and Parallel Machine Scheduling problems. More precisely, given a truck located at a central depot, a multitude of possible drone stations (micro-depots), and a set of customer locations, the TSDSLP seeks for a feasible routing of the truck and drone operations such that all requests are fulfilled, no more than a given number of drone stations is used, and the makespan (or operational cost) is minimized. We formulate the TSDSLP as a *Mixed-Integer Linear Program* (MILP) and use a state-of-the-art solver to obtain solutions for small- and medium-sized instances. Through our numerical results, we show that the utilization of drone stations might reduce the makespan significantly.

4.1 Introduction

Drones are on the verge of becoming a proven commercial technology for civil applications in many public and private sectors. In particular, drones have already been successfully applied for surveillance or monitoring tasks in agriculture, energy, or infrastructure, and for the delivery of packages (see (Otto et al., 2018) and references therein).

Murray and Chu introduced two novel NP-hard problems where a truck is assisted by a drone in last-mile delivery (Murray and Chu, 2015). The first one is called the *Flying Sidekick Taveling Salesman Problem* (FSTSP). In this case, if the depot is remotely located from the demand centers, it can be beneficial to have the drone working in close collaboration with the truck. To this end, they assume that the drone is taken along by the truck and might be launched at some locations to initiate an autonomous delivery. After fulfilling the requested demand, the drone must return to the truck in order to be resupplied with a new package. During recent years, a fast-growing number of research papers have been published, which follow the general concept of the FSTSP such that a high degree of *synchronization* between trucks and drones is required (see, e. g., (Agatz et al., 2018; Schermer et al., 2018a; Schermer et al., 2018b; Schermer et al., 2019a; Schermer et al., 2019b; X. Wang et al., 2017)).

By contrast, if most recipients are located in close proximity to the central depot, as an alternative to the FSTSP, Murray and Chu propose the *Parallel Drone Scheduling Traveling Salesman Problem* (PDSTSP). In this case, while the truck follows a tour and serves remote locations, the drone might be used to serve targets in close proximity to the depot, continuously departing from it to initiate a delivery and returning to it for picking up a new package. As trucks and drones are less dependent in the PDSTSP compared to the FSTSP, a much smaller degree of *synchronization* is required. Therefore, given specific technical constraints such as drone's limited payload weight and range of operation, many research studies are only concerned with the routing of drone fleets from a depot (see, e. g., (Dorling et al., 2017)). Similar considerations have been applied to the cases, where it might be necessary to determine the best location for multiple drone depots without taking care of routing trucks (see, e. g., (Chauhan et al., 2019)). These considerations involve, in the first place, a strategic perspective on establishing drone depots. To overcome the limitations of drones that are restricted to an existing (central) depot, Kim and Moon (2018) introduced the *Traveling Salesman Problem with Drone Station* (TSPDS) as a generalization of the PDSTSP. In the TSPDS, there is a single truck located at a central depot and a single *drone station* (micro-depot). Once the drone station has been supplied by the truck, the drones (located at the station) are used for distributing parcels. As in the PDSTSP, the objective of the TSPDS is to serve all customers with minimal makespan. In particular, Kim and Moon can show that, under special assumptions, it is possible to determine a priori which customers should be served by truck or drone, respectively. In these special cases, it is possible to decompose the problem into an independent *Traveling Salesman Problem* (TSP) in order to determine the route of the truck and a *Parallel Machine Scheduling Problem* to generate the operations at the drone station. To the best of our knowledge, there is no other published research article addressing challenges of the location problem in the context of the PDSTSP. However, the general domain of combined *location-routing* problems is not new (see, e. g., (Min et al., 1998; Nagy and Salhi, 2006), and references therein).

In this paper, we introduce the TSDSLP in which we address a research gap by considering the routing of the truck, facility location of the drone stations, and scheduling of drones in an integrated model. In the TSDSLP, we assume that a single truck is located at a central depot and a set of demand as well as a multitude of possible drone station locations are given. We require that the truck is sufficient to serve all customers with regard to its capacity and the scheduling horizon. The TSDSLP asks for a feasible route and drone operations, such that all customers are served, no more than a permitted number of drone stations are used, and minimal makespan (or operational cost) is achieved. To this end, a subset of drone stations, which might accommodate a given number of au-

tonomous drones, can be visited by the truck. If the truck supplies a station with parcels, then the station is considered as open, and the drones, which are resting there, can begin to help deliver packages in parallel. In an effort to reduce delivery times and costs, this concept might be a viable solution for the integration of autonomous auxiliary vehicles in last-mile and same-day delivery. Indeed, our numerical study demonstrates that an effective formulation of the TSDSLP enables a state-of-the-art solver to solve small- to medium-sized instances within reasonable computation time. In particular, compared to existing spacious depots at remote locations, drone stations might be a small-sized and low-cost infrastructure that provides shelter for parcels and drones as well as a recharging possibility for the latter in close proximity to demand centers (Kim and Moon, 2018).

The remainder of this paper is organized as follows. In Section 4.2, we specify the assumptions of the TSDSLP and formulate it as a MILP. Computational experiments and their numerical results are presented in Section 4.3. Finally, concluding remarks are drawn in Section 4.4.

4.2 Problem Definition

In this section, we introduce the TSDSLP that covers a more general case than the PDSTSP and TSPDS (Kim and Moon, 2018; Murray and Chu, 2015). More precisely, the TSDSLP not only integrates possibility of drones deliveries into the TSP tours, but also assumes the presence of *multiple* drone stations, which might be opened to be used for drone deliveries (see also Figure 4.2.1). Suppose that a single truck located at a depot, a multitude of drone stations that can accommodate a fixed number of drones, and a set of customer locations, each of them with a single demand, are given. In the TSDSLP, the objective consists in minimizing the makespan (or operational cost) such that all customers are served, either by the truck or by a drone. Furthermore, we accept the following assumptions regarding the nature of drones (Agatz et al., 2018; Murray and Chu, 2015; Schermer et al., 2018a; Schermer et al., 2018b; Schermer et al., 2019a; Schermer et al., 2019b; X. Wang et al., 2017):

- When launched from a station, each drone can fulfill exactly one request, and then it needs to return to the same station from which it was launched to be resupplied for future missions. Without loss of generality, we assume that each customer is eligible to be served by a drone.
- We assume that a drone has a limited endurance of $\mathcal{E}$ *distance units* per operation. After returning to the station, the battery of the drone is recharged (or swapped) instantaneously.

- While the truck is subject to the limitations of the road network, the drones might be able to use a different trajectory for traveling between locations. Furthermore, based on their technical constraints, the speed of the truck and drone might differ. Hence, without loss of generality, the average velocity of each truck is assumed to be equal to 1 and the relative velocity of each drone is assumed to be $\alpha \in \mathbb{R}^+$ times the velocity of the truck.
- We do not consider explicit service times; however, such considerations may be easily integrated into the model.
- We assume that at most $C \in \mathbb{Z}^+$ drone stations can be opened, and they are free of charge, except if we minimize the operational cost (see Section 4.2.2). Furthermore, we require that the potential sites of these stations are predetermined.

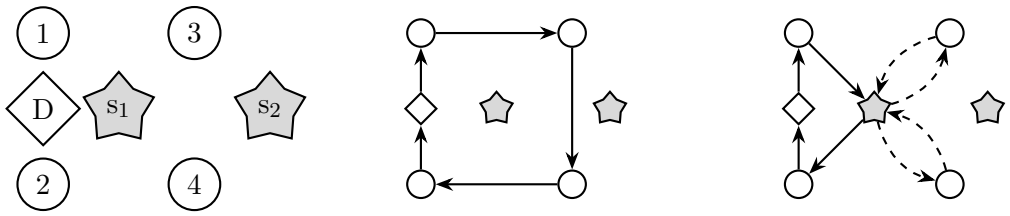

Figure 4.2.1: A TSDSLP with a depot D, four customers $V_N = \{1 \dots 4\}$, two drone stations $V_S = \{s_1, s_2\}$ that can accommodate two drones each, a TSP solution (center) and a TSDSLP solution (right) in which a station is utilized for two deliveries.

Let us present the notation that we are going to use throughout the paper. Assume that a complete and symmetric graph $\mathcal{G} = (V, E)$ is given, where V is the set of vertices and E is the set of edges. The set V contains n vertices associated with the customers, named $V_N = \{1, \dots, n\}$, a set of m possible drone stations $V_S = \{s_1, \dots, s_m\}$, and two extra vertices 0 and $n+1$ that (in order to simplify the notation and the mathematical formulation) both represent the same depot location at the start and end of the truck's tour. Thus, $V = \{0\} \cup V_N \cup V_S \cup \{n+1\}$, where $0 \equiv n+1$. To simplify the notation, we introduce the sets $V_L = V \setminus \{n+1\}$ and $V_R = V \setminus \{0\}$.

By the parameters d_{ij} and $\overline{d}_{ij}$ we define the distance required to travel from vertex i to vertex j by truck and drone, respectively. In principle, as the drones are not limited to the road network, the distances might differ. For the purpose of this work, Euclidean distance is considered for both the truck and drones. We use the parameters $v = 1$ and $\overline{v} = \alpha \cdot v$ to define the (constant) velocity of the truck and drones. Hence, the time required to traverse edge (i, j) is defined as $t_{ij} = d_{ij}$ and $\overline{t}_{ij} = \overline{d}_{ij}/\alpha$ for the truck and drones, respectively.

We assume that each drone station accommodates a limited and identical number of drones $D = \{1, \dots, D_n\}$, where $D_n \in \mathbb{Z}_{>0}$. Each drone may travel a maximum span of $\mathcal{E}$

distance units per operation, where a *drone operation* is characterized by a triple (d, s, j) as follows: the drone $d \in D$ is launched from a drone station $s \in V_S$, fulfills a request at $j \in V_N$, and returns to the same station from which it was launched.

4.2.1 Minimal Makespan TSDSLP

Table 4.2.1: Decision variables used in the TSDSLP formulation.

$\tau \in \mathbb{R}_{\geq 0}$	continuous variable that defines the makespan.
$x_{ij} \in \{0,1\}$ $\forall i \in V_L, j \in V_R$	is equal to 1, iff arc (i,j) is part of the truck's route.
$x^s_{ij} \in \{0,1\}$ $\forall i \in V_L, j \in V_R, s \in V_S$	is equal to 1, iff arc (i,j) is part of the truck's route to the drone station s.
$y^d_{sj} \in \{0,1\}$ $\forall s \in V_S, j \in V_N, d \in \mathcal{D}$	is equal to 1, iff drone d serves j from station s.
$z_s \in \{0,1\}$ $\forall s \in V_S$	is equal to 1, iff drone station s is opened.

To formulate the TSDSLP, we introduce the decision variables shown in Table 4.2.1. Using this notation, we have the following MILP formulation (4.1)–(4.15) of the TSDSLP:

$$\min \quad \tau, \tag{4.1}$$

$$\text{s.t.} \quad \sum_{i \in V_L} \sum_{\substack{j \in V_R \\ i \neq j}} t_{ij} x_{ij} \leq \tau, \tag{4.2}$$

$$\sum_{i \in V_L} \sum_{\substack{j \in V_R \\ i \neq j}} t_{ij} x^s_{ij} + \sum_{j \in V_N} 2 \cdot \bar{t}_{sj} \cdot y^d_{sj} \leq \tau \quad \forall s \in V_S, d \in D, \tag{4.3}$$

$$\sum_{\substack{i \in V_L, \\ i \neq j}} x_{ij} + \sum_{s \in V_S} \sum_{d \in D} y^d_{sj} = 1 \quad \forall j \in V_N, \tag{4.4}$$

$$\sum_{j \in V_R} x_{0j} = \sum_{i \in V_L} x_{i,n+1} = 1, \tag{4.5}$$

$$\sum_{\substack{i \in V_L \\ i \neq k}} x_{ik} - \sum_{\substack{j \in V_R \\ k \neq j}} x_{kj} = 0 \quad \forall k \in V_N \cup V_S, \tag{4.6}$$

$$\sum_{i \in \mathcal{S}} \sum_{\substack{j \in \mathcal{S} \\ i \neq j}} x_{ij} \leq |\mathcal{S}| - 1 \quad \forall \mathcal{S} \subset V, \{0, n+1\} \notin \mathcal{S}, |\mathcal{S}| > 1, \tag{4.7}$$

$$x^s_{ij} \leq x_{ij} \quad \forall s \in V_S, i \in V_L, j \in V_R, i \neq j, \tag{4.8}$$

$$\sum_{j \in V_R} x^s_{0j} = 1 \quad \forall s \in V_S, \tag{4.9}$$

$$\sum_{\substack{i \in V_L \\ i \neq k}} x_{ik}^s - \sum_{\substack{j \in V_R \\ j \neq k}} x_{kj}^s = 0 \quad \forall s \in V_S, k \in V_N \cup V_S, s \neq k, \tag{4.10}$$

$$\sum_{\substack{i \in V_L \\ i \neq s}} x_{is}^s - \sum_{\substack{i \in V_L \\ i \neq s}} x_{is} = 0 \quad \forall s \in V_S, \tag{4.11}$$

$$\sum_{\substack{i \in V_L \\ i \neq s}} x_{is} \geq z_s \quad \forall s \in V_S, \tag{4.12}$$

$$\sum_{s \in V_S} z_s \leq C, \tag{4.13}$$

$$\sum_{d \in D} \sum_{j \in V_N} y_{sj}^d \leq n z_s \quad \forall s \in V_S, \tag{4.14}$$

$$2 \cdot \overline{d}_{sj} \cdot y_{sj}^d \leq \mathcal{E} \quad \forall s \in V_S, d \in D, j \in V_N. \tag{4.15}$$

In this model, the objective function (4.1) minimizes the makespan τ. Constraints (4.2) and (4.3) describe τ mathematically. More precisely, constraint (4.2) sets the time spent traveling by the truck (to serve customer and supply stations) as a lower bound on the objective value. In constraints (4.3), for each station s, we account for the time until the truck has reached the station and then, for each drone d located at the station, the time spent fulfilling requests. These values are summed up to define lower bounds on τ.

Constraints (4.4) guarantee that each request j is served exactly once by either the truck or a drone. The flow of the truck is defined through constraints (4.5)–(4.6). More precisely, constraints (4.5) ensure that the truck starts and concludes its tour exactly once. For each customer or drone station, constraints (4.6) guarantee that the flow is preserved, i. e., the number of incoming arcs must equal the number of outgoing arcs. Moreover, constraints (4.7) serve as classical subtour elimination constraints (Dantzig et al., 1954), i. e., for each proper non-empty subset of vertices $\mathcal{S}$ (that does not contain the depot), no more than $|\mathcal{S}| - 1$ arcs can be selected within this set.

Constraints (4.8)–(4.11) specify the route of the truck that leads to each drone station s. To this end, constraints (4.8) ensure that this path must follow the path of the truck. Moreover, constraints (4.9) guarantee that the departure from the depot is always a part of each route to a station. Furthermore, for each vertex k that might be located in between the depot and the station, constraints (4.10) preserve the flow. In addition, for each station s that is visited by the truck, constraints (4.11) guarantee that there is exactly one arc leading to the station.

As specified by constraints (4.12), a drone station is opened only if it is visited by the truck. Moreover, constraint (4.13) guarantees that at most C drone stations may be opened. Constraints (4.14) restrict the number of drone operations that can only be per-

formed at opened drone stations. Constraints (4.15) determine the drone stations' range of operation. Note that these constraints might be effectively handled during preprocessing. Finally, according to the definition of the decision variables, $\tau \in \mathbb{R}_{\geq 0}$ and the other decision variables are binary.

In place of constraints (4.7), it is possible to adapt the family of *Miller-Tucker-Zemlin* (MTZ) constraints, using auxiliary integer variables u_i, as follows (Miller et al., 1960):

$$u_0 = 1, \tag{4.16}$$

$$2 \leq u_i \leq n + m + 2 \quad \forall i \in V_R, \tag{4.17}$$

$$u_i - u_j + 1 \leq (n + m + 2)(1 - x_{ij}) \quad \forall i \in V_L, j \in V_R, i \neq j, \tag{4.18}$$

$$u_i \in \mathbb{R}_{\geq 0} \quad \forall i \in V. \tag{4.19}$$

4.2.2 Minimal Operational Cost TSDSLP

As an alternative to the makespan minimization, we might be interested in cost minimization instead. In this case, we consider only the variable cost-per-mile that might be associated with the truck and drones and the fixed cost of using a station. To this end, we might use the following objective function:

$$\min \; c_t \sum_{i \in V_L} \sum_{\substack{j \in V_R \\ i \neq j}} d_{ij} x_{ij} + c_d \sum_{s \in V_S} \sum_{d \in D} \sum_{j \in V_N} (\overline{d}_{sj} + \overline{d}_{js}) y^d_{sj} + \sum_{s \in V_S} f_s z_s \tag{4.20}$$

where f_s is the fixed cost of operating the station s, and the parameters $c_t, c_d \in \mathbb{R}^+$ determine the relative cost for each mile that the truck and drones are in action. In this case, the model can remain unchanged with the exception that it is not necessary to consider the variables τ, x^s_{ij} and the respective constraints associated with these variables.

4.3 Computational Experiments

We implemented the model (4.1)–(4.15) and solved it by the MILP solver Gurobi Optimizer 8.1.0 (Gurobi Optimization, LLC, 2023). Throughout the solution process, the subtour elimination constraints (4.7) were treated as lazy constraints. More precisely, whenever the solver determines a new candidate incumbent integer-feasible solution, we examine if it contains any subtour. If no subtour is contained, we have a feasible solution. Otherwise, we calculate the set of vertices $\mathcal{S}$ that is implied by the shortest subtour contained in the current candidate solution. For this set $\mathcal{S}$, constraint (4.7) is added as a lazy constraint and the solver continues with its inbuilt branch-and-cut procedure. For comparative purposes,

we solved also the alternative formulation of the TSDSLP in which the MTZ constraints (4.16)–(4.19), in place of (4.7), are used. We carried out all experiments on single compute nodes in an Intel Xeon E5-2670 CPU cluster where each node was allocated 8 GB of RAM. A time limit of 10 minutes was imposed on the solver.

We generated the test instances according to the scheme shown in Figure 4.3.1. More precisely, we considered a 32×32 km^2 square grid where the customer locations $V_N = \{1, \ldots, n\}, n \in \{10, 30, 50\}$ were generated under uniform distribution. Furthermore, we assumed that the drone stations are located at the coordinates $(x, y) \in (8, \{8, 24\}) \cup (24, \{8, 24\})$. Moreover, we considered a central depot at $(x, y) = (16, 16)$. We investigated different cases; more precisely, the basic one follows the assumption of Murray and Chu (2015), where drones have a maximum *range of operation* of $\mathcal{E} = 16$ km. Therefore, the *radius of operation* associated with each station is $\mathcal{E}_\mathrm{r} = \mathcal{E}/2 = 8$ km. For the purpose of broadening our experiments, we tested the model with two other values of $\mathcal{E}_\mathrm{r}$.

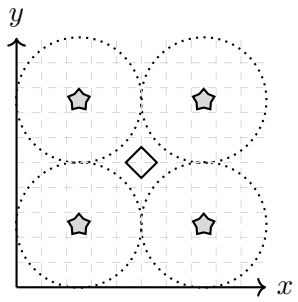

Figure 4.3.1: A visualization of the scheme according to which instances are generated.

In order to study the influence of problem parameters on the solver and the solutions, we considered their domains as follows. We tested for three different values for C, i. e., $C \in \{1, 2, 3\}$, and also did experiments for three distinct number of identical drones that a drone station can hold, i. e., $|D| \in \{1, 2, 3\}$. Moreover, we let the relative velocity be $\alpha \in \{0.5, 1, 2, 3\}$ and we assumed that the operational radius $\mathcal{E}_\mathrm{r} \in \{8, 12, 16\}$.

For each value of $n \in \{10, 30, 50\}$, we generated 10 random instances, which (along with the drone stations and the location of the depot) specify the graph $\mathcal{G}$. Furthermore, based on our choice of parameters $C, |D|, \alpha$ and $\mathcal{E}$, we have $3 \cdot 3 \cdot 4 \cdot 3 = 108$ parameter combinations; and a total of $30 \cdot 108 = 3240$ problems that are solved through both formulations.

Table 4.3.1 contains the numerical results of our computational experiments using two different formulations of the TSDSLP. In particular, for each number of customers n, the permitted number of stations C, and the operational radius $\mathcal{E}_\mathrm{r}$, this table shows the average runtime $\overline{t}$ (in seconds) as well as the average *Mixed-Integer Programming* (MIP) gap. While

comparing the results of two formulations, we observe that the differences on instances with $n = 10$ are negligible; however, on larger instances, the prohibition of subtours through lazy constraints improves the average runtimes and MIP gaps noticeably. Although medium-sized instances can be solved within reasonable time, the runtime depends strongly on n and the parameters C and $\mathcal{E}_\mathrm{r}$.

Table 4.3.1: Influence of the instance size n, the number of permitted stations C, the radius of operation $\mathcal{E}_\mathrm{r}$, and the formulation on the average runtime (in seconds) as well as MIP gap.

		MILP (4.1)–(4.15) with lazy constraints						MILP (4.1)–(4.6), (4.8)–(4.19)					
		$\mathcal{E}_\mathrm{r} = 8$		$\mathcal{E}_\mathrm{r} = 12$		$\mathcal{E}_\mathrm{r} = 16$		$\mathcal{E}_\mathrm{r} = 8$		$\mathcal{E}_\mathrm{r} = 12$		$\mathcal{E}_\mathrm{r} = 16$	
n	C	$\bar{t}$	Gap	$\bar{t}$	Gap	$\bar{t}$	Gap	$\bar{t}$	Gap	$\bar{t}$	Gap	$\bar{t}$	Gap
10	1	1.1	0.0%	1.3	0.0%	1.4	0.0%	1.9	0.0%	1.8	0.0%	2.0	0.0%
	2	1.1	0.0%	1.7	0.0%	2.3	0.0%	1.6	0.0%	2.1	0.0%	2.9	0.0%
	3	1.0	0.0%	1.7	0.0%	3.1	0.0%	1.6	0.0%	2.0	0.0%	3.0	0.0%
30	1	16.1	0.0%	28.0	0.0%	37.3	0.0%	83.2	0.0%	145.9	0.2%	198.8	0.3%
	2	43.9	0.0%	80.0	0.0%	178.8	0.6%	187.6	0.9%	301.5	2.3%	423.8	5.4%
	3	67.2	0.0%	227.6	0.9%	331.6	3.3%	228.9	1.6%	350.2	4.6%	442.0	8.7%
50	1	113.5	0.0%	292.4	0.2%	379.6	1.1%	430.8	2.1%	562.6	8.4%	591.0	13.2%
	2	289.1	0.5%	534.8	5.3%	583.9	11.4%	505.4	4.9%	600.2	18.9%	600.3	24.0%
	3	385.5	2.7%	549.8	13.7%	591.2	23.0%	526.6	7.2%	599.7	23.2%	597.0	28.4%

For the purpose of illustrating the benefits of utilizing the drone stations with regard to makespan reduction, we introduce the following metric, where τ is the objective value returned by the solver and τ^*_TSP is the optimal objective value of the TSP (that does not visit or use any drone station):

$$\Delta = 100\% - \frac{\tau}{\tau^*_\mathrm{TSP}} \tag{4.21}$$

Figure 4.3.2 highlights the numerical results. More precisely, this figure shows the average savings over the TSP, i. e., Δ, based on the number of permitted stations C, the number of drones located at each station $|D|$, as well as the drones' relative velocity α and radius of operation $\mathcal{E}_\mathrm{r}$. Overall, we can distinguish two cases. If the radius of operation is small ($\mathcal{E}_\mathrm{r} = 8$, solid lines) and the number of permitted stations C is fixed, the savings are *nearly* independent of the number of drones at each station and their velocity and radius of operation. In this case, the number of customers that can be served by the drones is limited (see Figure 4.3.1). However, even a slow-moving drone can effectively serve most (or all) customers within its radius of operation. An increase in the number of drones (or their relative velocity) will in this case not improve the overall makespan. On the other hand, if the radius of operation is large ($\mathcal{E}_\mathrm{r} = 16$, dashed lines), there is a significant impact of these parameters on the savings. In this case, the makespan can be reduced effectively

by increasing the number of drones (or their relative velocity). Furthermore, it is worth to highlight that, in many cases, significant savings are already possible with few drones (per station) that have a relative velocity of $\alpha \in \{0.5, 1\}$ but a large operational range. This contrasts problems that follow the fundamental idea of the FSTSP, where drones with relatively small endurance but fast relative velocity are often preferred (Agatz et al., 2018; Schermer et al., 2018a; Schermer et al., 2018b; Schermer et al., 2019a; Schermer et al., 2019b).

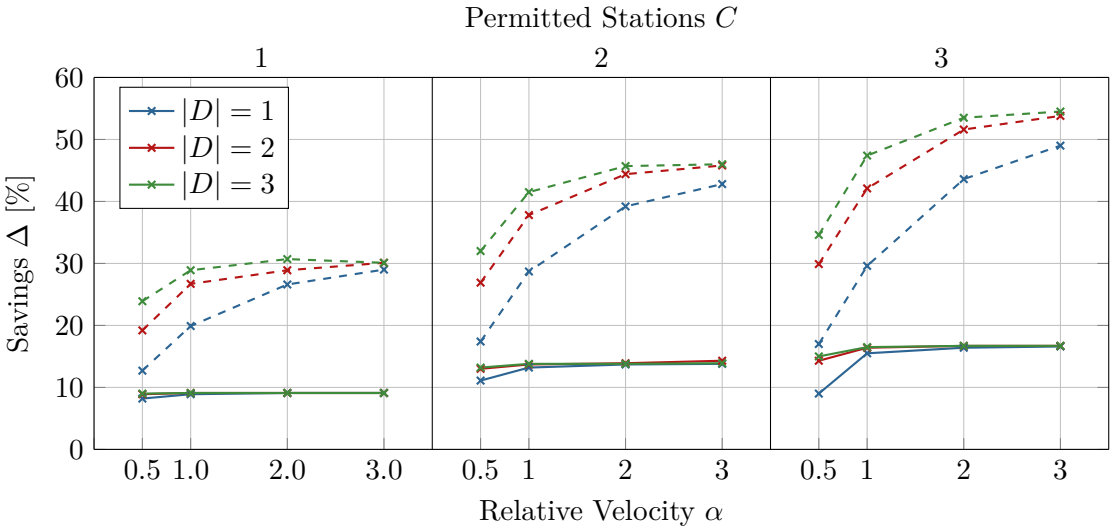

Figure 4.3.2: The savings Δ for different values of the problem parameters (averaged over all instances). Solid and dashed lines correspond to $\mathcal{E}_r = 8$ and $\mathcal{E}_r = 16$, respectively.

4.4 Conclusion

In this work, we introduced the TSDSLP, which combines the TSP and Facility Location Problem in which facilities are *drone stations*. After formulating the problem by two MILPs, we presented the results of our computational experiments. According to the numerical results, one of the formulations is far more effective than the other and can be used to solve medium-sized instances within reasonable computation time. Moreover, using suitable drone stations can bring a significant reduction in the delivery time. Particularly, this also applies for slow-moving drones, which are generally considered to be far less effective in related works.

Since TSDSLP defines a new concept, the future research directions are numerous, e. g., a research idea might consist in studying the case of using multiple trucks in place of a single one. Another research direction might focus on the design of efficient solution

methods. In fact, the standard solvers are able to solve only small- to medium-sized TSDSLP instances; hence, we might design effective heuristics, which can address large-scale instances. The research in this direction is in progress and the results will be reported in the future.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments and suggestions that helped us in improving the quality of this paper.

4.5 References

Agatz, Niels, Paul Bouman, and Marie Schmidt (2018). "Optimization Approaches for the Traveling Salesman Problem with Drone". In: *Transportation Science* 52.4, pp. 965–981.

Chauhan, Darshan, Avinash Unnikrishnan, and Miguel Figliozzi (2019). "Maximum coverage capacitated facility location problem with range constrained drones". In: *Transportation Research Part C: Emerging Technologies* 99, pp. 1–18.

Dantzig, George B., 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.

Dorling, Kevin et al. (2017). "Vehicle Routing Problems for Drone Delivery". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 47.1, pp. 70–85.

Gurobi Optimization, LLC (2023). *Gurobi Optimizer Reference Manual*.

Kim, Sungwoo and Ilkyeong Moon (2018). "Traveling Salesman Problem With a Drone Station". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 49.1, pp. 42–52.

Miller, C. E., A. W. Tucker, and R. A. Zemlin (1960). "Integer Programming Formulation of Traveling Salesman Problems". In: *Journal of the ACM* 7.4, pp. 326–329.

Min, Hokey, Vaidyanathan Jayaraman, and Rajesh Srivastava (1998). "Combined location-routing problems: A synthesis and future research directions". In: *European Journal of Operational Research* 108.1, pp. 1–15.

Murray, Chase C. and Amanda G. Chu (2015). "The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery". In: *Transportation Research Part C: Emerging Technologies* 54, pp. 86–109.

Nagy, Gábor and Saïd Salhi (2006). "Location-routing: Issues, models and methods". In: *European Journal of Operational Research* 177.2, pp. 649–672.

Otto, Alena et al. (2018). "Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey". In: *Networks* 72.4, pp. 411–458.

Schermer, Daniel, Mahdi Moeini, and Oliver Wendt (2018a). "A Variable Neighborhood Search Algorithm for Solving the Vehicle Routing Problem with Drones". Technical Report. Business Information Systems & Operations Research, Technische Universität Kaiserslautern.

Schermer, Daniel, Mahdi Moeini, and Oliver Wendt (2018b). "Algorithms for Solving the Vehicle Routing Problem with Drones". In: *Intelligent Information and Database Systems*. Lecture Notes in Computer Science 10751. Cham: Springer, pp. 352–361.

Schermer, Daniel, Mahdi Moeini, and Oliver Wendt (2019a). "A hybrid VNS/Tabu search algorithm for solving the vehicle routing problem with drones and en route operations". In: *Computers & Operations Research* 109, pp. 134–158.

Schermer, Daniel, Mahdi Moeini, and Oliver Wendt (2019b). "A matheuristic for the vehicle routing problem with drones and its variants". In: *Transportation Research Part C: Emerging Technologies* 106, pp. 166–204.

Wang, Xingyin, Stefan Poikonen, and Bruce Golden (2017). "The vehicle routing problem with drones: several worst-case results". In: *Optimization Letters* 11.4, pp. 679–697.

5 The Drone-Assisted Traveling Salesman Problem with Robot Stations

Abstract

In this paper, we study the *Drone-Assisted Traveling Salesman Problem with Robot Stations* (TSPDRS). Specifically, we assume that there is a single truck that is equipped with a drone, and one or more potential sites of stations that might accommodate some robots. The TSPDRS asks for a valid route of the truck as well as feasible utilization of the drone and robots, such that all customers are served and minimal delivery time (makespan) or cost is accomplished. We provide a *Mixed-Integer Linear Program* (MILP) formulation of the problem and perform a detailed numerical study. Through our numerical results, it is revealed that our formulation can be effectively addressed by a state-of-the-art solver. In addition, we demonstrate that optimizing the makespan coincides with reduced costs. In contrast, optimizing the operational costs might increase the makespan significantly. Furthermore, depending on the objective function, the operational utilization of the vehicles differs.

5.1 Introduction

Drones are on the brink of achieving market maturity as a commercial technology for civil applications in many public and private sectors. In particular, drones have already been successfully applied for monitoring tasks in agriculture, energy, or infrastructure, and for the delivery of packages (see (Otto et al., 2018) and references therein). Besides that, to reduce the negative impact of traffic in urban areas, there is a rising interest in using autonomous delivery robots in last-mile delivery (see, e. g., (Boysen et al., 2018a; Moeini and Salewski, 2020; Sonnenberg et al., 2019)). Due to the complementary nature of trucks, drones, and robots such as, e. g., their different speeds and carrying capacities, in a combined approach, benefits might emerge in terms of improved delivery times or reduced operational costs.

As a result, in this paper, we introduce the TSPDRS (see also Figure 5.1.1). To be more specific, we study the interoperability of a drone-assisted truck (refer to (Agatz et al., 2018; Murray and Chu, 2015)) with the possibility of using remote robot stations (refer to (Kim and Moon, 2018; Schermer et al., 2020c)) in an integrated model. In this problem, we assume that a truck with sufficient capacity, which carries a drone, is located

at a central depot. Moreover, we expect that a set of customer locations, each of them with an equal type of demand, as well as the potential sites of robot stations are known in advance. A subset of stations, which might accommodate a given number of autonomous robots, can be visited by the truck. The TSPDRS asks for a valid route of the truck, feasible operations of the drone, and admissible use of the robots, such that all customers are served and minimal delivery time (*makespan*) or cost is accomplished. Notably, this problem is concerned with various aspects of *synchronization* that determine the interplay between the vehicles (Drexl, 2012b). In other words, the interactions between the truck and auxiliary vehicles are governed by several operational constraints.

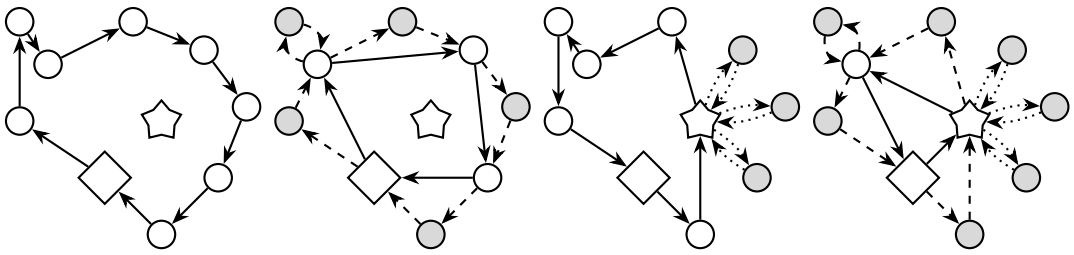

Figure 5.1.1: An illustration of TSP, TSPD, TSPDS, and TSPDRS solutions (from left to right): the paths of the truck, drones, and robots are indicated by solid, dashed, and dotted lines, respectively. In each figure, the depot and drone station (micro-depot) are indicated by the square and star shapes; however, the robots may only be utilized in the TSPDS and TSPDRS.

The remainder of this paper is organized as follows. We begin with a brief literature review in Section 5.2. Afterwards, in Section 5.3, we clarify the assumptions of the TSPDRS and provide a MILP formulation of the problem. Computational experiments and their numerical results are then presented in Section 5.4. Finally, in Section 5.5, we draw concluding remarks on our work and derive some possible implications for future research.

5.2 Related Literature

Murray and Chu (2015) introduced two novel problems, where a truck is assisted by a single drone in the context of last-mile delivery. The first one is called the *Flying Sidekick Taveling Salesman Problem* (FSTSP). Here, if the depot is remotely located from the demand centers, it might be beneficial to have the drone working in close collaboration with the truck. Therefore, for the FSTSP, we assume that the drone is taken along by the truck and might be launched at some locations to initiate a multi-leg flight. In detail, the first leg consists of the loaded drone departing from the truck to perform an unattended delivery to a customer. Subsequently, as the drone needs to be resupplied, the second (empty) leg consists of the drone autonomously returning to the truck. The objective is

to serve all customers and achieve minimal delivery time. Noteworthy, in the FSTSP, the drone cannot be recovered at the same location from which it was launched, i. e., round trips are forbidden (Murray and Chu, 2015). In contrast, in a closely related problem, named the *Traveling Salesman Problem with Drone* (TSPD) (Agatz et al., 2018), such a move is permitted (see Figure 5.1.1). In recent years, a rapidly increasing number of research papers have been published, which follow the general idea of the FSTSP and TSPD (see, e. g., (Poikonen et al., 2019; Poikonen et al., 2017; Schermer et al., 2019b; Sonnenberg et al., 2019; X. Wang et al., 2017)). These works address variants, provide theoretical insights, or develop advanced computational methods for solving problems involving a truck-drone tandem.

In contrast to the FSTSP, if the central depot is located within a dense customer cluster, the *Parallel Drone Scheduling Traveling Salesman Problem* (PDSTSP) provides a different perspective (Murray and Chu, 2015). In this case, the truck and drone perform their actions more independently of each other. Indeed, while the truck follows a predefined tour to serve remote locations, the drone might be used to fulfill requests that are located in the area surrounding the depot. While the truck serves the customers that might be inaccessible by the drone, the latter continuously performs round trips: starting from the depot, a customer is served, and the drone resupplied afterwards. Hence, a much smaller degree of synchronization is required in the PDSTSP (Murray and Chu, 2015).

In practice, it seems unlikely that the central depot is located in close proximity to demand centers. In order to overcome this issue, the *Traveling Salesman Problem with Drone Station* (TSPDS) was introduced as a generalization of the PDSTSP (Kim and Moon, 2018) (see also Figure 5.1.1). More precisely, apart from the central depot, where the truck is initially located, it is assumed that there is a single *drone station* (or *micro-depot*), which provides shelter to one or more drones. In the TSPDS, as soon as the drone station has been supplied by the truck, the drones can start their operation in a manner that mirrors the PDSTSP. In this problem, as in the FSTSP and PDSTSP, the objective is to serve all customers with minimal makespan, by effectively utilizing the drone station. Specifically, when the number of drones that are located at the station and their relative velocity are sufficiently large, it is possible to determine from the start which customers should be assigned to the truck or drone, respectively. Moreover, under these special assumptions, it is possible to decompose the problem into two independent subproblems (Kim and Moon, 2018): a *Traveling Salesman Problem* (TSP) to determine the route of the truck and a *Parallel Machine Scheduling Problem* to create a schedule for the drones that are located at the station.

Compared to existing large-scale central depots, drone stations (micro-depots) might be a low-cost infrastructure of much smaller scale. Above all, they need to provide shelter

along with the infrastructure that is needed for automated storage and retrieval of parcels as well as recharging of drones. Consequently, it is conceivable to incorporate the decision of locating (or selecting) a drone station from several potential sites into the problem at hand. Indeed, this was addressed in the *Traveling Salesman Drone Station Location Problem* (TSDSLP) that treats the routing of the truck, facility location of the drone stations, and scheduling of the drones in an integrated model (Schermer et al., 2020c). In general, works that follow the concept of the FSTSP conclude that a drone must have a sufficiently large velocity to provide significant improvements w. r. t. the makespan (see, e. g., (Agatz et al., 2018; Schermer et al., 2019a)). In contrast, Kim and Moon (2018) and Schermer et al. (2020c) can show that drone stations are also effective even when the speed of the drones is low compared to that of the truck.

To the best of our knowledge, at this point, there is no work that combines the concepts of the FSTSP (or TSPD) with the general idea of using drone stations. However, in an effort to further reduce delivery times or costs, it might be beneficial to consider an integrated delivery system that merges a (costly high-speed aerial) drone that travels along with the truck with some (cheaper ground-operated and slow-moving) robots that are coordinated directly from stations.

5.3 Problem Definition

In this section, we formally introduce the TSPDRS, which integrates the TSP with a drone (refer to (Agatz et al., 2018; Murray and Chu, 2015)) and the TSP with robot stations (refer to (Kim and Moon, 2018; Schermer et al., 2020c)). To be more precise, the TSPDRS combines the concepts of a drone traveling along with a truck with the possibility of the truck driver utilizing micro-depots that house further auxiliary vehicles, e. g., robots, which might be used for performing unattended deliveries. In this paper, we make no strong assumptions about the nature of the auxiliary vehicles. However, for the sake of readability, in what follows the term *drone* is used exclusively to refer to the (unmanned aerial) vehicle that travels along with the truck and the term *robot* to refer to one of the (unmanned ground) vehicles that are assigned to a station.

For the description of the problem, suppose that the truck-drone tandem is initially located at a central depot. Furthermore, we accept the presence of one or more robot stations that provide accommodation to a fixed number of robots. In the TSPDRS, the objective is to serve a given set of customers by utilizing the truck, drone, and robots in such a way that all vehicles have returned to their initial location as quickly as possible, i. e., with minimal *makespan*. Alternatively, we consider an objective that is based purely on the operational cost. We accept the following assumptions regarding the nature of

the drone and robots (Agatz et al., 2018; Kim and Moon, 2018; Murray and Chu, 2015; Schermer et al., 2020c):

- When the drone is launched from the truck, it can serve exactly one customer and afterwards it needs to return to the truck. Similarly, we ask that a robot can deliver exactly one parcel to a customer. Then, it also needs to return to the station from which it was launched for a resupply.
- We assume that the drone and robots have a limited endurance of $\mathcal{E}_d$ and $\mathcal{E}_r$ *time units* per operation, respectively. Moreover, we ask that the batteries of the drone and robots are recharged (or swapped) instantaneously after each operation.
- While the truck is restricted to follow the road network, the drone and robots might be able to use different routes with different speeds. Hence, the times required to travel between two locations can differ for the truck, drone, and robots.
- In this problem, we assume that at most $C \in \mathbb{Z}_{\geq 0}$ robot stations can be utilized by the truck (which might be a managerial decision).

Let us present the notation that we are going to use throughout the paper. Assume that a complete graph $\mathcal{G} = (V, E)$ is given, where V is the set of vertices and E is the set of edges. We identify several pairwise disjoint subsets in V:

- $\{0, n+1\} \subset V$ marks the depot at the start and end of the tour, respectively.
- $V_N = \{1, \ldots, n\} \subset V$ is the set of all customers. Moreover, $V_D \subseteq V_N$ is the subset of customers that may be served by either the truck, drone, or robot. But, $V_N \setminus V_D$ is the subset of customers that may be served by truck only.
- $V_S = \{s_1, \ldots, s_m\} \subset V$ is the set of potential robot station sites that are present in the network.

Thus, $V = \{0\} \cup V_N \cup V_S \cup \{n+1\}$, where $0 \equiv n+1$. To simplify the notation, we introduce two further sets named $V_L = V \setminus \{n+1\}$ and $V_R = V \setminus \{0\}$ that mark the locations from which a drone may be *launched* and *retrieved*, respectively. A *drone operation* (or *sortie*) is characterized by a triple $(i, w, j) \in P$, where P is the set of all feasible operations and the following conditions must be fulfilled (Agatz et al., 2018; Murray and Chu, 2015):

- The location i, from which the drone is launched, must be contained in the set V_L.
- The customer $w \in V_D$, that is served by the drone, must be different from i.
- At any retrieval location j, $w \neq j$ must hold. In addition, if $i = j$, we call it a *cyclic* (loop) operation or a *round trip*. Otherwise, if $i \neq j \in V_R$, we call it an *acyclic* or *multi-leg* operation.

By the parameters d_{ij}, $\overline{d}_{ij}$, and $\tilde{d}_{ij}$, for all $i, j \in V$, we define the distance required to travel from vertex i to vertex j by truck, drone, and robot, respectively. Similarly, we use t_{ij}, $\overline{t}_{ij}$, and $\tilde{t}_{ij}$ to define the times required to travel from vertex i to j depending on the mode of transportation. Furthermore, we assume that $\overline{t}_l$ (respectively, $\overline{t}_r$) time units are required to launch (respectively, recover) the drone (Murray and Chu, 2015).

We assume that each robot station accommodates a limited and identical number of robots $K = \{1, \ldots, k\}$, where $|K| \in \mathbb{Z}_{\geq 0}$. A *robot delivery* is characterized by a triple (k, s, w) as follows: the robot $k \in K$ starts from a station $s \in V_S$, fulfills a request at $w \in V_D$, and returns to the same station from which it was launched (Kim and Moon, 2018; Schermer et al., 2020c).

5.3.1 Minimal Makespan TSPDRS

Table 5.3.1: Decision variables used in the TSPDRS formulation.

$\tau \in \mathbb{R}_{\geq 0}$	defines the makespan.
$x_{ij} \in \{0,1\}$ $\forall i \in V_L, j \in V_R, i \neq j$	is equal to 1, iff arc (i, j) is used by the truck.
$y_{iw} \in \{0,1\}$ $\forall i \in V_L, w \in V_D, i \neq w$	is equal to 1, iff the drone serves w from i, i. e., arc (i, w) is the start of a multi-leg.
$y'_{wj} \in \{0,1\}$ $\forall w \in V_D, j \in V_R, w \neq j$	is equal to 1, iff the drone returns from w to j, i. e., arc (w, j) is the end of a multi-leg.
$\hat{y}_{iw} \in \{0,1\}$ $\forall i \in V_L, w \in V_D, i \neq w$	is equal to 1, iff the drone performs a round trip (i, w, i).
$y_i \in \{0,1\}$ $\forall i \in V$	is equal to 1, iff the drone is available for an operation at i.
$z^k_{sw} \in \{0,1\}$ $\forall k \in K, s \in V_S, w \in V_D$	is equal to 1, iff robot k serves customer w from station s.
$a_i \in \mathbb{R}_{\geq 0}$ $\forall i \in V$	indicates the arrival time of the truck at i.
$s_i \in \mathbb{R}_{\geq 0}$ $\forall i \in V$	states the earliest possible departure time of the truck from i.
$\overline{r}_i \in \mathbb{R}_{\geq 0}$ $\forall i \in V$	indicates the earliest possible release time of the drone at i.
$\overline{s}_i \in \mathbb{R}_{\geq 0}$ $\forall i \in V$	states the earliest possible departure time of the drone from i.

To formulate the TSPDRS, we use the decision variables shown in Table 5.3.1. Using this notation, we introduce the following MILP formulation (5.1)–(5.27).

In the TSPDRS, according to (5.1), the objective is to minimize the makespan τ. The makespan has multiple lower bounds which are given by the time at which the truck-drone tandem has returned to the depot as well as the time at which each robot station

has concluded its operation. These bounds are defined in constraints (5.2) and (5.3) for all vehicle types.

$$\min \quad \tau \tag{5.1}$$

$$\text{s. t.} \quad s_{n+1} \leq \tau \tag{5.2}$$

$$a_s + \sum_{w \in V_D} (\tilde{t}_{sw} + \tilde{t}_{ws}) z_{sw}^k \leq \tau \quad \forall s \in V_S, k \in K \tag{5.3}$$

The flow of the truck is defined through constraints (5.4) and (5.5). More precisely, according to constraints (5.4), the truck should commence and conclude its tour exactly once. Moreover, for each vertex h, that is visited by the truck in between, the flow must be conserved according to constraints (5.5).

$$\sum_{j \in V_R} x_{0j} = \sum_{i \in V_L} x_{i,n+1} = 1 \tag{5.4}$$

$$\sum_{\substack{i \in V_L, \\ i \neq h}} x_{ih} = \sum_{\substack{j \in V_R, \\ h \neq j}} x_{hj} \quad \forall h \in V_N \cup V_S \tag{5.5}$$

In the TSPDRS, every customer must be served exactly once. Constraints (5.6) cover the customers that can be served exclusively by the truck. For the remaining customers, constraints (5.7) guarantee that they are served exactly once by either truck, drone, or robot.

$$\sum_{\substack{i \in V_L, \\ i \neq w}} x_{iw} = 1 \quad \forall w \in V_N \setminus V_D \tag{5.6}$$

$$\sum_{\substack{i \in V_L, \\ i \neq w}} (x_{iw} + y_{iw} + \hat{y}_{iw}) + \sum_{s \in V_S} \sum_{k \in K} z_{sw}^k = 1 \quad \forall w \in V_D \tag{5.7}$$

The TSPDRS has intensive operational *synchronization* requirements. For the truck and the drone, these are handled through constraints (5.8)–(5.10). To be more precise, through constraints (5.8) (respectively, (5.9)), the truck is required to visit every vertex i that serves as a starting point for a multi-leg (respectively, round trip) operation. Furthermore, when a drone is retrieved at some vertex j after a multi-leg operation, the truck is required to visit the vertex due to constraints (5.10). Finally, constraints (5.11) guarantee that the

flow is conserved during multi-leg operations.

$$y_{iw} \leq \sum_{\substack{j \in V_R \\ i \neq j}} x_{ij} \quad \forall i \in V_L, w \in V_D, i \neq w \tag{5.8}$$

$$\hat{y}_{iw} \leq \sum_{\substack{j \in V_R \\ i \neq j}} x_{ij} \quad \forall i \in V_L, w \in V_D, i \neq w \tag{5.9}$$

$$y'_{wj} \leq \sum_{\substack{i \in V_L \\ i \neq j}} x_{ij} \quad \forall w \in V_D, j \in V_R, w \neq j \tag{5.10}$$

$$\sum_{\substack{i \in V_L \\ i \neq w}} y_{iw} = \sum_{\substack{j \in V_R \\ w \neq j}} y'_{wj} \quad \forall w \in V_D \tag{5.11}$$

Temporal constraints for the truck are given through constraints (5.12) and (5.13). The first set of constraints applies the arrival time as a lower bound on the departure time. In addition, the second set of constraints provides a lower bound on the arrival time at j for every arc (i, j) that is used by the truck.

$$a_i \leq s_i \quad \forall i \in V \tag{5.12}$$

$$M(x_{ij} - 1) + s_i + t_{ij} \leq a_j \quad \forall i \in V_L, j \in V_R, i \neq j \tag{5.13}$$

For every multi-leg (i, w, j) performed by the drone, constraints (5.14) and (5.15) provide lower bounds on the release time of the drone at the customer and retrieval location. These constraints incorporate the departure time of the drone as well as the overhead times and flight times associated with each leg.

$$M(y_{iw} - 1) + \overline{s}_i + (\overline{t}_l + \overline{t}_{iw}) \leq \overline{r}_w \quad \forall i \in V_L, w \in V_D, i \neq w \tag{5.14}$$

$$M(y'_{wj} - 1) + \overline{s}_w + (\overline{t}_{wj} + \overline{t}_r) \leq \overline{r}_j \quad \forall w \in V_D, j \in V_R, w \neq j \tag{5.15}$$

Due to constraints (5.16), the earliest possible departure time $\overline{s}_i$ of the drone depends on its release time $\overline{r}_i$ at i and the time spent performing round trips at the same vertex. Moreover, the earliest possible departure time of the truck is bounded by the earliest possible release time of the drone. If a drone starts a multi-leg, the overhead time to launch the drone must be applied which is stated through constraints (5.17). Similarly, expressed through constraints (5.18), when a drone is retrieved after a multi-leg, the overhead time to retrieve the drone must be considered before the drone can be released

again.

$$\overline{r}_i + \sum_{\substack{w \in V_D, \\ i \neq w}} (\overline{t}_l + \overline{t}_{iw} + \overline{t}_{wi} + \overline{t}_r)\hat{y}_{iw} \leq \overline{s}_i \quad \forall i \in V \tag{5.16}$$

$$\overline{s}_i + \overline{t}_l \sum_{\substack{w \in V_D, \\ i \neq w}} y_{iw} \leq s_i \quad \forall i \in V \tag{5.17}$$

$$a_j + \overline{t}_r \sum_{\substack{w \in V_D, \\ w \neq j}} y'_{wj} \leq \overline{r}_j \quad \forall j \in V \tag{5.18}$$

Constraints (5.19) and (5.20) define the maximum endurance for round trips and multi-leg drone flights.

$$(\overline{t}_{iw} + \overline{t}_{wi})\hat{y}_{iw} \leq \mathcal{E}_d \quad \forall i \in V_L, w \in V_D, i \neq w \tag{5.19}$$

$$\sum_{\substack{i \in V_L, \\ i \neq w, \\ i \neq j}} \overline{t}_{iw} y_{iw} + (\overline{r}_j - \overline{s}_w) \leq \mathcal{E}_d + M(1 - y'_{wj}) \quad \forall w \in V_D, j \in V_R \tag{5.20}$$

Constraints (5.21) account for the availability of the drone: for every arc (i, j) used by the truck, these constraints place a valid upper bound on the variable y_j depending on the previous availability, i. e., y_i, as well as starts of a multi-leg at i and recoveries after a multi-leg at j.

$$(1 - x_{ij}) + y_i + \sum_{\substack{w \in V_D, \\ w \neq i \\ w \neq j}} (-y_{iw} + y'_{wj}) \geq y_j \quad \forall i \in V_L, j \in V_R, i \neq j \tag{5.21}$$

If a drone is available, it can start or end a multi-leg at most once and perform an arbitrary number of round trips at the same location. This behavior is expressed through constraints (5.22)–(5.24).

$$\sum_{\substack{w \in V_D, \\ w \neq i}} y_{iw} \leq y_i \quad \forall i \in V_L \tag{5.22}$$

$$\sum_{\substack{w \in V_D, \\ w \neq j}} y'_{wj} \leq y_j \quad \forall j \in V_R \tag{5.23}$$

$$\sum_{\substack{w \in V_D, \\ i \neq w}} \hat{y}_{iw} \leq n y_i \quad \forall i \in V_L \tag{5.24}$$

At this point, we have fully specified the interaction of the truck and drone. Next, we will integrate the robot stations into the TSPDRS. We say that a station is *used* if it is visited by the truck. Through constraints (5.25), we require that at most $C \in \mathbb{Z}_{\geq 0}$ stations may be used. Furthermore, constraints (5.26) guarantee that an arbitrary number of robot deliveries might be performed at each station s that is visited by the truck.

$$\sum_{i \in V_L} \sum_{\substack{s \in V_S \\ i \neq s}} x_{is} \leq C \tag{5.25}$$

$$\sum_{k \in K} \sum_{w \in V_D} z_{sw}^k \leq n \sum_{\substack{i \in V_L, \\ i \neq s}} x_{is} \quad \forall s \in V_S \tag{5.26}$$

We conclude our model with constraints (5.27) that guarantee the maximum endurance to be respected for all robot operations.

$$(\tilde{t}_{sw} + \tilde{t}_{ws}) z_{sw}^k \leq \mathcal{E}_r \quad \forall k \in K, s \in V_S, w \in V_D \tag{5.27}$$

Even though MILP (5.1)–(5.27) formulates the TSPDRS, the linear relaxation can be improved by including some simple yet highly effective valid inequalities (see (Schermer et al., 2019b)). For the purpose of this work, we adapt them through constraints (5.28) and (5.29) as follows:

$$\begin{aligned} \sum_{i \in V_L} \sum_{\substack{j \in V_R, \\ i \neq j}} t_{ij} x_{ij} + \sum_{i \in V_L} \sum_{\substack{w \in V_D, \\ i \neq w}} \bar{t}_l y_{iw} + \sum_{w \in V_D} \sum_{\substack{j \in V_R, \\ w \neq j}} \bar{t}_r y'_{wj} \\ + \sum_{i \in V_L} \sum_{\substack{w \in V_D, \\ i \neq w}} (\bar{t}_l + \bar{t}_{iw} + \bar{t}_{wi} + \bar{t}_r) \hat{y}_{iw} \leq \tau \end{aligned} \tag{5.28}$$

$$\begin{aligned} \sum_{i \in V_L} \sum_{\substack{w \in V_D, \\ i \neq w}} (\bar{t}_l + \bar{t}_{iw}) y_{iw} + \sum_{w \in V_D} \sum_{\substack{j \in V_R, \\ w \neq j}} (\bar{t}_{wj} + \bar{t}_r) y'_{wj} \\ + \sum_{i \in V_L} \sum_{\substack{w \in V_D, \\ i \neq w}} (\bar{t}_l + \bar{t}_{iw} + \bar{t}_{wi} + \bar{t}_r) \hat{y}_{iw} \leq \tau \end{aligned} \tag{5.29}$$

These constraints state that the makespan is bounded by the time spent traveling by either the truck or drone and the time shared by both vehicles when they perform *synchronized* actions: these actions occur during the launches and recoveries associated with multi-leg operations, i. e., whenever the overhead times $\bar{t}_l$ and $\bar{t}_r$ are applied, as well as during round trips.

As a general remark, our model loosely follows the MILP models presented in (Kim

and Moon, 2018; Schermer et al., 2019b; Schermer et al., 2020c). However, with respect to modeling the interaction between the truck and drone, the model provided in this paper is far more compact. In particular, through the introduction of constraints (5.21), we explicitly account for the availability of the drone at every vertex without tracking *circumjacent* drone operations (refer to (Schermer et al., 2019a; Schermer et al., 2019b)). Through this procedure, we *significantly* reduce the number of variables and constraints involved in the formulation. In fact, we only require $\mathcal{O}(|V|^2)$ variables and constraints for modeling the problem. Furthermore, in contrast to existing models (see, e. g., (Kim and Moon, 2018; Murray and Chu, 2015; Schermer et al., 2020c)), we require no explicit *subtour elimination constraints*. In fact, constraints (5.13)–(5.15) already eliminate the possibility of any subtour. Finally, we need only a few big-M constraints. To be more precise, these constraints only appear in the temporal constraints (5.13)–(5.15) and endurance constraints (5.20), which helps to strengthen the linear relaxation further. As a result of these improvements, to the best of our knowledge, at this time, we are the first to effectively solve a multitude of instances with 10 or more customers to optimality within acceptable runtime and to approach instances with up to 20 customers reasonably well through a general-purpose MILP solver (refer to Section 5.4).

5.3.2 Minimal Operational Cost TSPDRS

As an alternative to the *makespan minimization*, we might be interested in *cost minimization* instead. While minimizing the cost, we might consider the variable cost-per-mile that might be associated with the truck, drone, and every robot. To this end, we might use the following objective function:

$$\begin{aligned}\min \quad & c_t \sum_{i \in V_L} \sum_{\substack{j \in V_R \\ i \neq j}} d_{ij} x_{ij} + c_d \sum_{i \in V_L} \sum_{\substack{w \in V_D, \\ i \neq w}} \left(\left(\overline{d}_{iw} + \overline{d}_{wi} \right) \hat{y}_{iw} + \overline{d}_{iw} y_{iw} \right) \\ & + c_d \sum_{w \in V_D} \sum_{\substack{j \in V_R, \\ w \neq j}} \overline{d}_{wj} y'_{wj} + c_r \sum_{s \in V_S} \sum_{k \in K} \sum_{w \in V_D} (\tilde{d}_{sw} + \tilde{d}_{ws}) z^k_{sw}\end{aligned} \tag{5.30}$$

In this objective function, the parameters $c_t, c_d, c_r \in \mathbb{R}^+$ determine the relative cost for each mile that the truck, drone, and robots are in operation, respectively.

As a concluding remark, as the TSPDRS (with either objective) generalizes the $\mathcal{NP}$-hard TSP, the TSPDRS is also $\mathcal{NP}$-hard.

5.4 Computational Experiments and their Numerical Results

We implemented the model (5.1)–(5.29) and solved it by the MILP solver Gurobi Optimizer 8.1.0 (Gurobi Optimization, LLC, 2023) with a time limit of 30 minutes. We carried out all experiments on compute nodes in an Intel Xeon E5-2670 CPU cluster where each node was allocated 8 GB of RAM.

5.4.1 Instance Generation

Several different assumptions have been made with regard to the technical coefficients that apply to trucks, drones, and robots. For trucks, these assumptions include speeds that range from 15 to 35 mph (refer to (Boysen et al., 2018a; Ha et al., 2018; Moeini and Salewski, 2020; Murray and Chu, 2015; Sacramento et al., 2019)) and, as drones are not exposed to the limitations of the road network, it is commonly assumed that they move faster than trucks. In the literature, for drones, the speed varies from 25 to 50 mph along with 15 to 40 minutes of flight time (refer to (Ha et al., 2018; Moeini and Salewski, 2020; Murray and Chu, 2015; Sacramento et al., 2019)).

Table 5.4.1: Overview of technological coefficients used for studying the TSPDRS.

Scenario	Truck		Drone			Robots		
	v_t [mph]	c_t	v_d [mph]	$\mathcal{E}_D$ [min]	c_d	v_r [mph]	$\mathcal{E}_r$ [min]	c_r
Slow robot	20	1	25	20	1/10	5	40	1/20
Fast robot	20	1	25	20	1/10	15	40	1/20

Generally, when it is of interest to minimize the makespan, faster drones are more favorable; however, the marginal utility is diminishing as we speed up drones (Agatz et al., 2018; Schermer et al., 2019b). For the purpose of this work, we conservatively assume that we have a truck that moves at 20 mph and a drone that moves at 25 mph with a maximum flight time of 20 minutes. Moreover, we differentiate between robots that move close to walking speed, i. e., 5 mph, and faster robots that move at 15 mph (Boysen et al., 2018a; Schermer et al., 2020c). For both cases, we set the robots' endurance to 40 minutes. Further, we assume that the Manhattan distance applies to the truck while the Euclidean distance applies to the drone and robots. This is a reasonable assumption to make because drones might move as the crow flies and robots might follow routes that are inaccessible to the truck such as, e. g., pedestrian zones. For both of these networks, we assume that the distances are symmetric. Finally, we set the overhead times associated with a launch and recovery of the drone to $\overline{t}_l = \overline{t}_r = 1$ minute (Ha et al., 2018; Murray and Chu, 2015).

To the best of our knowledge, there is no work that quantifies the variable cost-per-mile

associated with trucks in comparison to (delivery) drones and robots. However, related works consider drones to be 10 to 25 times more cost-efficient than trucks (refer to (Ha et al., 2018; Sacramento et al., 2019)). In this study, we accept these assumptions and use the cost ratios $c_t = 1, c_d = 1/10, c_r = 1/20$ for the minimal cost objective in (5.30). Table 5.4.1 contains the technological coefficients, under which the problem is studied.

Apart from the problem parameters, an instance is specified by a graph $\mathcal{G}(V, E)$ that determines the locations of the depot, stations, and customers as well as the distances between them. As the TSPDRS is a new concept, we generated several graphs that match to the problem.[1] More precisely, for each number of customers $n \in \{10, 15, 20\}$, we generated 10 instances as follows:

- The locations of the depot and n customers have been *randomly* placed within an 8×8 mile square region (Murray and Chu, 2015).

- For comparative purposes, we placed two robot stations that might be used. The first robot station is placed *randomly* within the square grid. The second robot station is placed at the coordinates (x_s, y_s) that depend on the locations of the n customers according to formula (5.31). This station might be located closer to customer clusters.

$$(x_s, y_s) \coloneqq \frac{1}{n} \sum_{w \in V_N} (x_w, y_w) \tag{5.31}$$

Afterwards, we solved these instances in four different ways. As a baseline, we treated them as a TSP. For comparative purposes, we also solved them (under the minimal makespan and cost objectives) as a TSPD, i. e., when $C = 0$. For the TSPDRS, under the minimal makespan objective, we did the experiments with $C \in \{1, 2\}$ and $|K| \in \{1, 2\}$. In contrast, under the minimal cost objective, as no fixed cost is incurred for deploying the second robot, we only tested with $|K| = 1$.

For the TSPDRS, based on our choice of parameters $C, |K|$, and the two different scenarios portrayed in Table 5.4.1, we have 2^3 parameter combinations for the minimal makespan objective and 2^2 parameter combinations for the minimal cost objective. Therefore, given the 30 different problem instances, we have a total of 30 TSP, 60 TSPD, as well as 120 minimal cost TSPDRS and 240 minimal makespan TSPDRS, which are all solved by the MILP solver. Notably, for a problem that is solved according to the minimal makespan objective, we also calculate the cost that can be associated with the solution. Analogously, under the minimal cost objective, we also compute the corresponding makespan. For solving the instances, we assumed that V_N coincides with V_D, i. e., all customers are

[1] See `https://doi.org/10.5281/zenodo.3446016`

drone- and robot-eligible. From a computational point of view, due to the complicating constraints (5.7), this is the most challenging case. As the decision of whether a customer should be served by the truck, drone, or robot must be made for all customers, $V_N = V_D$ provides the largest solution space.

5.4.2 Numerical Results

In this section, we provide the numerical results of our computational study. In particular, we comment on the performance of the MILP solver in addressing the proposed instances and the potential makespan and cost savings. Afterwards, we analyze the way in which the drone and robots are utilized depending on the selected objective function.

5.4.2.1 MILP Solver Performance

Table 5.4.2 provides some insights into the performance of the MILP solver. More precisely, this table contains the average runtime and *Mixed-Integer Programming* (MIP) gaps for the studied cases. Moreover, the table indicates how often the MILP solver was able to identify the provably optimal solution within the runtime limit of 30 minutes.

Table 5.4.2: Detailed results obtained through Gurobi Optimizer 8.1.0. This table contains the average runtime (in seconds), the average MIP gap after runtime, and the number of times (opt.) in which the provably optimal solution was found (values are averaged over 10 instances). The parameters n, v_r, $|K|$, and C are the number of customers, velocity of the robots, the amount of robots, and the number of stations that can be utilized.

			TSPD ($C = \|K\| = 0$)						TSPDRS								
			min cost			min make.			min cost			min make. ($\|K\| = 1$)			min make. ($\|K\| = 2$)		
n	v_r	C	time	gap	opt.	time	gap	opt.	time	gap	opt.	time	gap	opt.	time	gap	opt.
10	5	1	1.2	0.0%	10/10	18.9	0.0%	10/10	1.7	0.0%	10/10	101.8	0.0%	10/10	62.9	0.0%	10/10
		2							2.2	0.0%	10/10	144.3	0.0%	10/10	108.6	0.0%	10/10
	15	1							1.1	0.0%	10/10	12.4	0.0%	10/10	5.6	0.0%	10/10
		2							1.2	0.0%	10/10	13.2	0.0%	10/10	4.8	0.0%	10/10
15	5	1	12.5	0.0%	10/10	739.1	1.5%	7/10	29.2	0.0%	10/10	883.8	1.8%	8/10	711.1	1.8%	8/10
		2							40.9	0.0%	10/10	949.0	2.0%	7/10	896.0	2.3%	7/10
	15	1							4.7	0.0%	10/10	774.0	2.0%	8/10	204.8	0.0%	10/10
		2							4.9	0.0%	10/10	699.0	1.8%	8/10	160.3	0.0%	10/10
20	5	1	193.8	0.0%	10/10	1,645.2	9.0%	1/10	365.1	0.0%	10/10	1,678.4	9.5%	1/10	1,571.9	9.8%	2/10
		2							451.4	0.0%	10/10	1,637.1	11.1%	2/10	1,614.8	9.4%	2/10
	15	1							46.4	0.0%	10/10	1,711.1	10.1%	2/10	1,780.6	10.6%	1/10
		2							45.9	0.0%	10/10	1,800.0	15.9%	0/10	1,580.2	10.5%	3/10
Average			69.2	0.0%	30/30	801.1	3.5%	18/30	82.9	0.0%	120/120	867.0	4.5%	76/120	725.1	3.7%	83/120

Regarding the TSPD cases, overall, the numerical results show that our MILP formulation is well-suited for solving small-sized problems. Under the minimal cost objective, we can solve all instances to proven optimality within reasonable runtime. Moreover, under

the minimal makespan objective, our formulation can solve instances with 10 customers to optimality within short runtime. Furthermore, we are able to approach instances with 15 customers with low MIP gaps and can even solve 7 out of 10 to proven optimality. However, it also becomes apparent that performance degrades as the instance size is increased to $n = 20$, in particular, when we are optimizing the makespan instead of cost.

For the TSPDRS, we are able to solve instances with 10 to 15 customers reasonably well and achieve optimality in almost all cases. Indeed, under the minimal cost objective, we can even approach instances with 20 customers and achieve performance that rivals the TSPD cases. Specifically, in the presence of fast robots, i. e., $v_r = 15$, the problem often becomes easier to solve. However, it also becomes apparent that, in general, the minimal makespan problem is much more difficult to solve: for $n = 20$ the remaining MIP gaps after runtime increase and the number of instances that are solved to optimality decreases noticeably.

5.4.2.2 Makespan versus Cost Minimization

For the purpose of illustrating the benefits of utilizing the drone and robots with regard to *cost versus makespan reduction*, we introduce the following metrics:

$$\Delta_\tau = \frac{\tau}{\tau^*_{\text{TSP}}} \quad \text{and} \quad \Delta_c = \frac{c}{c^*_{\text{TSP}}}, \tag{5.32}$$

where, τ and c are the objective values returned by the solver and τ^*_{TSP} as well as c^*_{TSP} stand for the optimal makespan and cost of the TSP solution, i. e., where no drone is present and no station is visited or used.

Figure 5.4.1 highlights the numerical results for the minimal makespan objective. Specifically, this figure shows the average savings over the TSP, i. e., Δ_τ and Δ_c, based on the number of available robots $|K|$ and their velocity v_r for a given value of using at most $C = 1$ stations. Note that, in this figure, the values for $K = 0$ are taken from the TSPD solution (where $C = |K| = 0$ but the drone can be used). Notably, under the makespan objective, the costs are also reduced significantly. Based on the selected technological coefficients (refer to Table 5.4.1), the savings are of similar magnitude. In particular, robots that move close to walking speeds of 5 mph have a hardly noticeable influence under the minimal makespan objective. In contrast, as their speed is increased to 15 mph, they become increasingly useful.

Figure 5.4.2 visualizes the results for the minimal cost objective. As expected, under this objective, even smaller costs can be realized. However, in contrast to the makespan objective, the costs and delivery time show a contrary development (see also (Ha et al.,

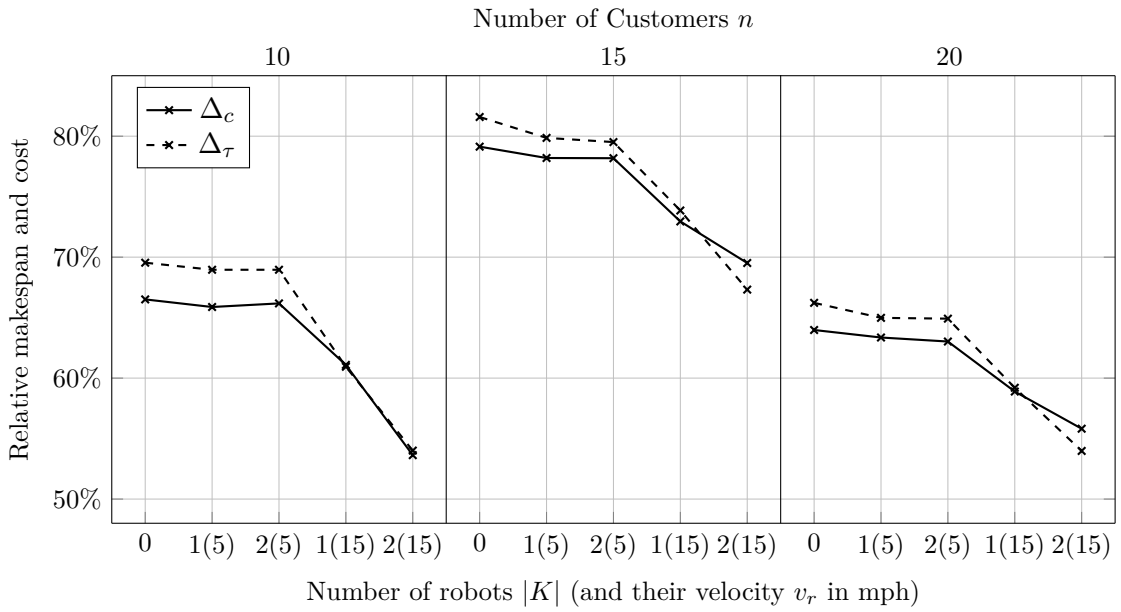

Figure 5.4.1: The relative makespan Δ_τ and cost Δ_c for the minimal makespan objective. These values are shown for $C = 1$ ($C = 0$ for the TSPD solution, i. e., where $|K| = 0$) and averaged over 10 instances.

2018)): as we increase the speed of the robots, the relative costs decrease further, but there is a sharp increase in the makespan. As a possible explanation, an increase in speed also raises the robots' range (see constraints (5.27)); and while operating robots might be cheap (due to the cost factor c_r), this throttles the makespan nonetheless.

5.4.2.3 Station Utilization

Regarding the use of the robot stations, we present Table 5.4.3, on which we can make the following observations. When the objective is to minimize the operational cost or if the number of used stations is limited to $C = 1$, it is much more likely that the weighted station is utilized. In contrast, this ratio becomes less definite under the minimal makespan objective, where it can also make sense to visit the randomly placed station. In these cases, it might also be useful to utilize a station as an intermediate hub in order to launch or recover the drone (see also (Schermer et al., 2019a)).

5.4.2.4 Operational Characteristics

We are also interested to investigate the operational characteristics of the drone and robots. For this purpose, Table 5.4.4 illustrates the share of customers that are served by the truck, through a multi-leg or a round trip, and by a robot.

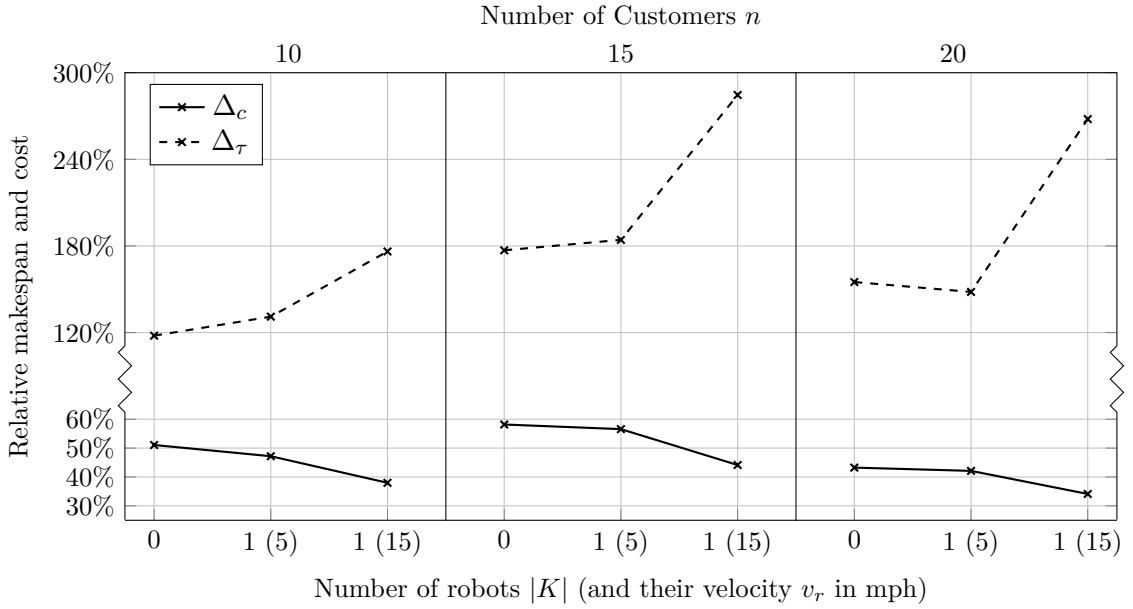

Figure 5.4.2: The relative makespan Δ_τ and cost Δ_c for the minimal cost objective. These values are shown for $C = |K| = 1$ ($C = 0$ for the TSPD solution, i. e., where $|K| = 0$) and averaged over 10 instances.

Table 5.4.3: The share of times the weighted or randomly located stations are used under the minimal cost and minimal makespan (time) objective (averaged over 30 instances for $n \in \{10, 15, 20\}$).

v_r	C	min cost		min time ($\|K\| = 1$)		min time ($\|K\| = 2$)	
		weighted	random	weighted	random	weighted	random
5	1	63.3%	16.7%	16.7%	43.3%	26.7%	40.0%
	2	70.0%	16.7%	40.0%	50.0%	36.7%	43.3%
15	1	96.7%	3.3%	63.3%	33.3%	83.3%	16.7%
	2	96.7%	10.0%	93.3%	66.7%	100.0%	76.7%

If the objective is to achieve minimal operational cost, the utilization of the truck is expensive. Moreover, the cheaper modes of delivery, i. e., the drone and in particular the robots, are utilized to a large extent. Indeed, the predominant mode of operating the drone is the round trip. Noteworthy, to the best of our knowledge, most research works (existing in the literature) that study related problems under a minimal cost objective, do not permit drones to perform round trips (see, e. g., (Ha et al., 2018; Sacramento et al., 2019)). However, if only the variable cost-per-mile is considered, a round trip clearly is the most cost-effective operation that a drone can perform. More precisely, consider a feasible multi-leg $(i, w, j) \in P$. Based on the objective defined in (5.30), if $\overline{d}_{iw} < \overline{d}_{wj}$, it is always feasible and, in particular, more cost-efficient to perform the round trip (i, w, i)

Table 5.4.4: Share of customers served by each vehicle (averaged over 10 instances for $C = 1$).

		minimal cost ($\|K\| = 1$)				minimal makespan ($\|K\| = 1$)				minimal makespan ($\|K\| = 2$)			
n	v_r	truck	multi-leg	round trip	Robot	truck	multi-leg	round trip	Robot	truck	multi-leg	round trip	Robot
10	5	20.0%	0.0%	74.0%	6.0%	63.0%	31.0%	0.0%	6.0%	63.0%	31.0%	0.0%	6.0%
	15	14.0%	0.0%	16.0%	70.0%	54.0%	22.0%	0.0%	24.0%	40.0%	25.0%	0.0%	35.0%
15	5	21.3%	0.0%	72.7%	6.0%	67.3%	26.7%	0.0%	6.0%	64.7%	25.3%	0.0%	10.0%
	15	12.0%	0.0%	14.7%	73.3%	60.7%	21.3%	0.0%	18.0%	50.7%	18.7%	0.0%	30.7%
20	5	23.5%	0.0%	65.5%	11.0%	69.0%	25.0%	0.0%	6.0%	67.5%	25.5%	0.0%	7.0%
	15	12.0%	0.0%	15.5%	72.5%	62.5%	21.5%	0.0%	16.0%	53.0%	16.5%	0.0%	30.5%

instead. Note that, analogous considerations are possible for the case of $\overline{d}_{iw} > \overline{d}_{wj}$.

On the contrary, when the objective is to achieve minimal makespan, the utilization of the truck ramps up significantly. Equally important, the predominant mode of drone delivery is the multi-leg and the share of customers served by robots declines noticeably. It is interesting to note that when fast robots are used, the shares of customers served by the drone and robot, respectively, are of a similar magnitude.

5.5 Conclusion

In this work, we introduced the TSPDRS, which combines the concepts of a truck-drone tandem with the possibility of robot stations. After a brief literature review, we formulated the problem as a MILP, and presented the numerical results of our computational study. We have shown that a state-of-the-art solver yields acceptable results within reasonable runtime for instances with up to 20 customers. According to the numerical results, combining a truck-drone tandem along with properly situated robot stations can bring significant reductions in the delivery time or operational costs. Furthermore, drone round trips are a suitable mode of operation when the objective is to minimize the costs – surprisingly, this is only scarcely considered in the literature. However, we believe that one must be careful with the impact that this might have on the delivery time.

As the TSPDRS defines a new concept, the future research directions are numerous. A subsequent sensitivity analysis on parameters (e. g., drone speed) might provide some more in-depth insights on their respective impact. While our MILP formulation has shown favorable behavior, it will be necessary to rely on heuristic methods in order to address larger instances. Moreover, in the context of same-day-delivery, it might be of interest to formulate the TSPDRS as a multi-objective problem: to achieve an improved trade-off, we might weigh time against cost savings and include advanced cost metrics such as, e. g., a cost that is imposed for waiting (refer to (Ha et al., 2018)). Finally, it is also interesting to consider the impact of multiple trucks, several drones per truck or the

possibility of enhanced robots with multiple compartments (see, e. g., (Schermer et al., 2019b; Sonnenberg et al., 2019)).

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments and suggestions that helped us in improving the quality of this paper.

5.6 References

Agatz, Niels, Paul Bouman, and Marie Schmidt (2018). "Optimization Approaches for the Traveling Salesman Problem with Drone". In: *Transportation Science* 52.4, pp. 965–981.

Boysen, Nils, Stefan Schwerdfeger, and Felix Weidinger (2018a). "Scheduling last-mile deliveries with truck-based autonomous robots". In: *European Journal of Operational Research* 271.3, pp. 1085–1099.

Drexl, Michael (2012b). "Synchronization in Vehicle Routing—A Survey of VRPs with Multiple Synchronization Constraints". In: *Transportation Science* 46.3, pp. 297–316.

Gurobi Optimization, LLC (2023). *Gurobi Optimizer Reference Manual*.

Ha, Quang Minh et al. (2018). "On the min-cost Traveling Salesman Problem with Drone". In: *Transportation Research Part C: Emerging Technologies* 86, pp. 597–621.

Kim, Sungwoo and Ilkyeong Moon (2018). "Traveling Salesman Problem With a Drone Station". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 49.1, pp. 42–52.

Moeini, Mahdi and Hagen Salewski (2020). "A Genetic Algorithm for Solving the Truck-Drone-ATV Routing Problem". In: *Optimization of Complex Systems: Theory, Models, Algorithms and Applications*. Advances in Intelligent Systems and Computing 991. Cham: Springer, pp. 1023–1032.

Murray, Chase C. and Amanda G. Chu (2015). "The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery". In: *Transportation Research Part C: Emerging Technologies* 54, pp. 86–109.

Otto, Alena et al. (2018). "Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey". In: *Networks* 72.4, pp. 411–458.

Poikonen, Stefan, Bruce Golden, and Edward A. Wasil (2019). "A Branch-and-Bound Approach to the Traveling Salesman Problem with a Drone". In: *INFORMS Journal on Computing* 31.2, pp. 335–346.

Poikonen, Stefan, Xingyin Wang, and Bruce Golden (2017). "The vehicle routing problem with drones: Extended models and connections". In: *Networks* 70.1, pp. 34–43.

Sacramento, David, David Pisinger, and Stefan Ropke (2019). "An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones". In: *Transportation Research Part C: Emerging Technologies* 102, pp. 289–315.

Schermer, Daniel, Mahdi Moeini, and Oliver Wendt (2019a). "A hybrid VNS/Tabu search algorithm for solving the vehicle routing problem with drones and en route operations". In: *Computers & Operations Research* 109, pp. 134–158.

Schermer, Daniel, Mahdi Moeini, and Oliver Wendt (2019b). "A matheuristic for the vehicle routing problem with drones and its variants". In: *Transportation Research Part C: Emerging Technologies* 106, pp. 166–204.

Schermer, Daniel, Mahdi Moeini, and Oliver Wendt (2020c). "The Traveling Salesman Drone Station Location Problem". In: *Optimization of Complex Systems: Theory, Models, Algorithms and Applications*. Advances in Intelligent Systems and Computing 991. Cham: Springer, pp. 1129–1138.

Sonnenberg, Marc-Oliver et al. (2019). "Autonomous Unmanned Ground Vehicles for Urban Logistics: Optimization of Last Mile Delivery Operations". In: *Proceedings of the Hawaii International Conference on System Sciences*, pp. 1538–1547.

Wang, Xingyin, Stefan Poikonen, and Bruce Golden (2017). "The vehicle routing problem with drones: several worst-case results". In: *Optimization Letters* 11.4, pp. 679–697.

6 A Branch-and-Cut Approach and Alternative Formulations for the Traveling Salesman Problem with Drone

Abstract

In this paper, we are interested in studying the *Traveling Salesman Problem with Drone* (TSPD). Given a set of customers and a truck that is equipped with a single drone, the TSPD asks that all customers are served exactly once and minimal delivery time is achieved. We provide two compact *Mixed-Integer Linear Program* (MILP) formulations that can be used to address instances with up to ten customers within a few seconds. Notably, we introduce a third formulation for the TSPD with an exponential number of constraints. The latter formulation is suitable to be solved by a *Branch-and-Cut* (B&C) algorithm. Indeed, this approach can be used to find optimal solutions for several instances with up to 20 customers within one hour, thus challenging the current state-of-the-art in solving the TSPD. A detailed numerical study provides an in-depth comparison on the effectiveness of the proposed formulations. Moreover, we reveal further details on the operational characteristics of a drone-assisted delivery system. By using three different sets of benchmark instances, consideration is given to various assumptions that affect, e. g., technological drone parameters and the impact of distance metrics.

6.1 Introduction

Probing drones and other autonomous auxiliary vehicles in last-mile delivery is a current trend that is investigated by retailers and logistics service providers such as Amazon Inc. or Deutsche Post DHL Group (see (Murray and Chu, 2015) and references therein). The integration of such vehicles into existing logistic architectures might yield synergistic benefits that could emerge in terms of time or cost savings, both of which might be crucial for effectively carrying out cost-efficient same-day deliveries in the future (refer to (Aurambout et al., 2019)). In fact, drones have already become a proven technology in many related public and private sectors including energy, agriculture and forestry, environmental protection, and emergency response (see (Otto et al., 2018) and references therein).

Generally, we might accept that a drone can move faster between two locations than

a truck as a drone has a larger average velocity, can move as the crow flies, and is not subjected to the limitations of a road network such as, e. g., congestion. In addition, drones are far more lightweight than trucks, which might cause them to consume less energy when moving from one point to another. However, a drone's payload weight and cargo capacity in terms of volume are inherently limited. Moreover, as multirotor drones rely on comparatively small batteries for powering their flight, their range of operation is quite restricted compared to a traditional delivery truck.

Consequently, due to the complementary nature of both vehicles, combining a truck with a drone gives rise to several new problems in logistics. In recent years, these problems have received a rising interest from the optimization community. Indeed, several new problems have been proposed in the academic literature in which different possibilities of integrating drones are studied (see, e. g., (Agatz et al., 2018; Kim and Moon, 2018; Murray and Chu, 2015; X. Wang et al., 2017)). In an attempt to classify these problems, Otto et al. (2018) proposed the following criteria that allow for a differentiation (for more details, we point to the aforementioned paper and the references therein):

1. *Drones as the main performance drivers*, i. e., trucks supporting the operations of drones: this might be the case if, e. g., a truck only serves as a (passive) carrier for a drone.

2. *Trucks as the main performance drivers*, i. e., drones supporting the operations of trucks: this case might occur if, e. g., a truck is merely resupplied by a drone during its tour.

3. *Drones and trucks performing independent tasks*: in such scenarios, a truck and a drone might be located at a common distribution center but perform deliveries without interacting with one another.

4. *Drones and trucks as synchronized working units*: these approaches have received significant attention in the literature and might be further classified as follows:

   - *Sidekick-based approaches* (see, e. g., (Agatz et al., 2018; Murray and Chu, 2015)): in these cases, a drone is assumed to be carried along by a delivery truck. However, the truck is not just a passive carrier, i. e., customers can be served by either truck or drone.
   - *Station-based approaches* (see, e. g., (Kim and Moon, 2018; Schermer et al., 2020b)): here, the truck and drone(s) might be located at distinct distribution centers resembling multiple echelons. While the truck could initially be at a central depot, the drone(s) might be located at a remote *drone station*. A

drone station could be a low-cost infrastructure (a micro-depot) located in close proximity to demand centers that contains one or more drones.

This paper is concerned with the TSPD that is a sidekick-based approach. As initially identified by Murray and Chu (2015) and confirmed in many subsequent works (see, e. g., (Agatz et al., 2018; Bouman et al., 2018a; Poikonen et al., 2019)), finding optimal solutions to these kinds of problems poses a significant computational challenge. In fact, these problems are strongly involved with several aspects of *synchronization* (refer to (Drexl, 2012b)). Consequently, from an algorithmic point of view, currently, no efficient method exists for computing optimal solutions even to small problems in short computation time. While it might be reasonable to accept (unless $\mathcal{P} = \mathcal{NP}$) that no such method may ever exist, given the $\mathcal{NP}$-hard nature of the underlying problem, we still believe that there is meaning in accelerating the search process for provably optimal solutions to small instances. Therefore, the contributions of our work are as follows:

1. As an alternative to existing formulations in the literature (see, e. g., (Agatz et al., 2018; Murray and Chu, 2015)), we introduce two novel MILP formulations as well as some effective *Valid Inequalities* (VIEQs) (see (Schermer et al., 2019b; Schermer et al., 2020b)). At this point, instances with just 9 to 10 customers are considered to be quite difficult to be solved optimally within reasonable time. By themselves, using a state-of-the-art solver, our proposed formulations can be used to solve such instances within a few seconds.

2. Notably, we propose a third formulation with an exponential number of constraints that is suitable be integrated in a B&C solver. Arguably, motivated by the VIEQs, this method leverages the strengths of existing and *compact* MILP formulations (e. g., (Murray and Chu, 2015; Schermer et al., 2019b; Schermer et al., 2020b)) by merging them with the basic idea of the *set-covering* based *Integer Linear Programming* (ILP) approach proposed by Agatz et al. (2018). Through this combined approach, we can address several instances with up to 20 customers in less than one hour. As a consequence, several previously unsolved instances can be solved to proven optimality.

3. In any case, through our numerical study, we confirm previous observations that a drone-assisted truck might allow for a significantly reduced delivery time, and provide some further insights into the operational characteristic of such a delivery system. Furthermore, we show that, using our formulations, different benchmark instances and their associated assumptions have a noticeable influence on the performance of the MILP solver and the effectiveness of the truck-drone delivery system.

The remainder of this paper is organized as follows. We begin in Section 6.2 with a brief overview of the related literature. Afterwards, Section 6.3 introduces the notation, two compact mathematical models, and the proposed VIEQs for the TSPD. As an alternative, a third formulation with an exponential number of constraints is discussed in Section 6.4. We present the detailed results of our numerical studies in Section 6.5. Finally, concluding remarks on this work and future research implications are drawn in Section 6.6.

6.2 Related Literature

For the purpose of this paper, we restrict ourselves to works that consider the *sidekick-based* approach (see Figure 6.2.1) under a minimal delivery time objective (refer to (Agatz et al., 2018; Murray and Chu, 2015)) as these problems have intensive *synchronization* requirements (refer to (Drexl, 2012b)). In Section 6.2.1, we provide a brief account on the development of novel MILP formulations or any other exact method that comes with an optimality certificate. Afterwards, in Section 6.2.2, we highlight some additional works that are primarily concerned with heuristics. The works presented in these sections have some overlaps. In fact, many of the authors that introduced a novel MILP formulation, also developed heuristics for solving the problem (see, e. g., (Agatz et al., 2018; Murray and Chu, 2015)). More general literature reviews on the potentials of utilizing drones in a civil manner in and outside the context of vehicle routing problems can be found in (Otto et al., 2018; Rojas Viloria et al., 2020).

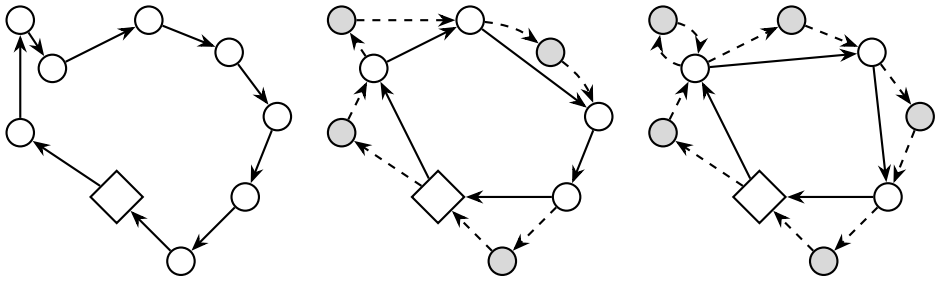

Figure 6.2.1: An illustration of a possible TSP, FSTSP, and TSPD solution (from left to right). The paths of the truck and drone are indicated by solid and dashed lines, respectively. In each figure, the depot is indicated by the square shape.

6.2.1 Model-Based Literature

The *Flying Sidekick Taveling Salesman Problem* (FSTSP), which aims at finding a drone-assisted *Traveling Salesman Problem* (TSP) tour, was introduced by Murray and Chu (2015). In this problem, it is assumed that a depot and a set of customers are given,

each of whom must be served exactly once (see Figure 6.2.1). Initially, the truck-drone tandem is located at a central depot. The objective is to find a feasible routing of the truck and drone such that all customers are served and both vehicles have returned to the depot as quickly as possible, i. e., with minimal *makespan*. The authors propose a MILP formulation for solving the FSTSP and specify a drone *sortie* by a triple (i, j, k): the drone is launched at vertex i, serves customer $j, i \neq j$, and is retrieved by the truck at a third vertex $k, j \neq k$. Notably, in their model, *round trips* are forbidden, i. e., for every sortie (i, j, k) the following condition must hold: $i \neq k$. Using a state-of-the-art MILP solver, the authors cannot find any provably optimal solution within a 30-minute time limit for instances with just 10 customers. As an alternative, Murray and Chu (2015) propose a heuristic for solving the problem.

The TSPD was proposed by Agatz et al. (2018) and shares most of the assumptions of the FSTSP. However, there are a few key differences to the FSTSP. Firstly, the drone can be retrieved at the same location from which it was launched (see Figure 6.2.1). Secondly, in principle, the truck may visit customers more than once as this could be beneficial in operating the drone (see Figure 6.2.2) (see also (Morandi et al., 2023)).

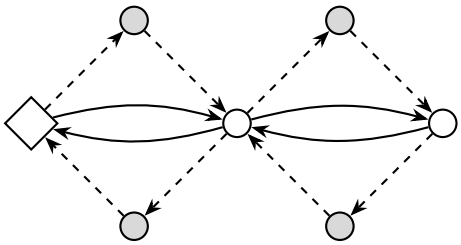

Figure 6.2.2: In the TSPD, it might be useful to visit vertices repeatedly to increase the utilization of the drone.

Agatz et al. (2018) propose a set-covering based ILP formulation for solving the TSPD. Within this approach, they use an *operation* as the essential building block (see also Figure 6.4.1): an operation is characterized by an ordered sequence of at least two (or more) vertices as visited by the truck. Within this sequence, at most one customer can be served by the drone through a sortie that connects the start and end of the sequence. Using the ILP formulation, the authors can tackle instances with 9 customers within no more than 40 seconds. However, due to the exponential number of operations w. r. t. the number of customers, they also report that this approach does not scale well. In fact, on instances with just 11 customers, several hours are required to obtain optimal solutions. Notably, some theoretical insights and heuristics that leverage local search and *Dynamic Programming* (DP) are also provided in this work.

In a follow-up paper, Bouman et al. (2018a) provide another method that is based on

DP. The authors highlight that the new DP method is able to find optimal solutions much faster compared to the ILP approach presented in (Agatz et al., 2018). In fact, the speedup is about fortyfold on instances with 9 customers. However, it also becomes apparent that the performance degrades as the instance size is increased: for provably optimal solutions, instances with 12 customers already take 1 minute, an increase to 15 customers requires more than two hours of computation time, and no solution is found within 12 hours beyond this size. To overcome this issue, they propose a heuristic variant of their DP approach where only a subset of feasible operations is considered. The authors can show that a considerable speedup can be achieved through this procedure with just a slight loss on the solution quality w. r. t. optimal solutions.

Es Yurek and Ozmutlu (2018) provide an exact method for solving the TSPD that is based on a decomposition of the problem into two components. First, all feasible tours to the TSP are generated that visit every possible subset of customers. Each tour provides a *Lower Bound* (LB) on a TSPD solution in which the customers that are not covered by the tour must be served by the drone. Starting with the shortest tour (smallest LB), mathematical programming is used to identify the optimal drone sorties. This procedure continues until the best remaining LB is worse than the incumbent TSPD solution. The authors compare their results to (Agatz et al., 2018; Murray and Chu, 2015) and highlight that they can solve instances with 10 customers within 5 minutes and instances with 12 customers within 30 minutes. Although, the authors stress that performance depends largely on the type of instance as well as parameters (e. g., speed of the drone). In particular, their approach performs poorly on densely clustered instances: here, for the case of 12 customers, the algorithm rarely terminates within one hour.

In the work of Poikonen et al. (2019), a *Branch-and-Bound* (B&B) method is introduced for solving the TSPD. Each node in the B&B search tree represents a delivery sequence as visited by the truck, which might contain only a subset of customers. At each node, for the customers not covered by this sequence, the optimal TSPD solution can be computed through the DP algorithm proposed in (Agatz et al., 2018). During the search, child nodes are created by inserting customers into the parental sequence. Poikonen et al. (2019) stress that, in the TSPD, the objective value could also improve if an additional customer is inserted into the sequence. Therefore, as we climb the tree, for some children, the objective value obtained through the DP can in fact be smaller than that of its ancestors. The authors introduce the *tree exploration ratio* as a parameter that provides a tradeoff on optimality guarantee versus speed. With an optimality guarantee, they can solve instances with 9 customers in just one minute. However, the authors also report that the solution time becomes intractably large when the number of customers is increased to 14.

Dell'Amico et al. (2021b) propose a modified FSTSP model that builds on the foun-

dation laid by Murray and Chu (2015). In particular, they propose a new formulation in which some explicit constraints are replaced with exponentially many constraints that can be added in a cutting plane manner. The affected constraints concern restrictions regarding, e. g., the availability of the drone and the correct orientation of sorties (w. r. t. the route of the truck). Dell'Amico et al. (2021b) can show that these changes lead to an improvement in performance as they can solve several instances with 10 customers optimally within one hour. Moreover, in a subsequent work, Dell'Amico et al. (2019) propose a more compact formulation. This is achieved by explicitly accounting for the drone's availability at each vertex (a similar approach is used in (Schermer et al., 2019b)). The authors can show that the B&C formulation yields favorable results as instances from (Murray and Chu, 2015) with 10 customers can be solved in about 15 minutes on average. Notably, the endurance of the drone has a significant influence. Moreover, a few instances with up to 20 customers can be solved optimally within one hour of runtime.

In (Schermer et al., 2020b) a variant of the TSPD named the *Drone-Assisted Traveling Salesman Problem with Robot Stations* (TSPDRS) is proposed. The TSPDRS merges the sidekick-based with the station-based approaches (refer to Section 6.1). The compact MILP formulation presented in that paper also covers the truck-drone as a special case. Using a state-of-the-art solver, the formulation can be used to solve these special cases in relatively short computation time. Indeed, problem instances with ten customers can be solved within 20 seconds and instances with up to 15 customers in less than 15 minutes.

Even though most assumptions are commonly accepted, the existing literature is in fact quite divergent: several further subtle nuances differentiate the various problem statements from one another, making a comparison only valid to a limited extent – in fact, besides this work, there are four different problem statements, all of which are called "TSPD" by the respective authors (refer to (Agatz et al., 2018; Es Yurek and Ozmutlu, 2018; Ha et al., 2018; Poikonen et al., 2019)). Apart from the problem instances and concrete choice of parameters (e. g., the drone's velocity), details of the problem statements differ in whether *round trips* are permitted, if vertices can be *revisited* by the truck, and whether both vehicles operate on the same type of *network*. Moreover, there is no consensus on if an *overhead time* is incurred for launching and recovering the drone and how such penalties should be applied concretely. Furthermore, there is no agreement on if a drone is permitted to wait at a customer after making a delivery, thus conserving energy before the return flight, or if, instead, it must hover while waiting for the truck to pick it up again. In any case, there is no generally recognized way of specifying a drone's *endurance* precisely.

As will be unveiled in much more detail in the following sections, in this paper we provide three new formulations; in particular, one of them can be used to design a B&C scheme for solving the TSPD effectively. Indeed, we can optimally solve instances with

$n = 9$ or $n = 10$ customers in just a few seconds, instances with $n = 14$ customers within around 5 minutes, and several instances with $n = 19$ or $n = 20$ customers within an hour.

6.2.2 Methodology-Based Literature

Several further works exist that study variants of the problem (multiple trucks, drones, or different objective functions) or are primarily concerned with heuristics. For the reasons listed at the beginning of this section, we just mention them briefly.

Ponza (2016) discusses the possibilities of solving the FSTSP heuristically using meta-heuristics such as *Simulated Annealing* (SA) or *Ant Colony Optimization*. Of these possible algorithms, SA is implemented and evaluated. For small-scale instances with at most 10 customers, on which optimal solutions can be obtained through a MILP formulation (that follows the one proposed by Murray and Chu (2015)), the SA algorithm can obtain optimal solutions in most cases. After the investigation of these small instances, a more detailed parameter study is performed on larger instances with up to 200 customers.

Freitas and Penna (2020) propose a *Variable Neighborhood Search* (VNS) for solving the FSTSP and TSPD (see also (Freitas and Penna, 2018)). In this algorithm, the first step consists in determining the optimal TSP solution. Next, a greedy improvement procedure is proposed that takes customers from the truck's route and assigns them to drone sorties. Finally, VNS serves as an in-depth improvement procedure, for which the authors propose several problem-specific neighborhood structures. Freitas and Penna provide a detailed numerical study and show the effectiveness of their approach by testing on the problem instances proposed in (Agatz et al., 2018; Murray and Chu, 2015; Ponza, 2016).

The work by Ha et al. (2018) studies a variant of the TSPD with a minimal cost (as opposed to time) objective. The authors provide a MILP formulation that is based on the one provided in (Murray and Chu, 2015). However, the primary contribution of the work is the design of an effective *Greedy Randomized Adaptive Search Procedure* (GRASP) that can be used to insert drone operations into an existing TSP solution as well as some local search operators for refining the solution quality afterwards. In a subsequent work, the same authors investigate the performance of a *Hybrid Genetic Algorithm* as an alternative to their GRASP (Ha et al., 2020). The authors show that this novel algorithm outperforms existing approaches in terms of solution quality on instances with up to 100 customers.

Similarly, Sacramento et al. (2019) study a variant of the TSPD that involves multiple vehicles from a cost-per-mile-oriented perspective. For this purpose, they extend the MILP formulation that was proposed in (Murray and Chu, 2015). In particular, they propose an Adaptive Large Neighborhood Search procedure that is capable of solving large-scale instances with up to 250 customers.

Following (Poikonen et al., 2017; X. Wang et al., 2017), Schermer et al. (2019b) investigate a variant of the TSPD that features multiple vehicles and drones per vehicle named the *Vehicle Routing Problem with Drones* (VRPD). A MILP formulation and some effective VIEQs are introduced. However, the primary contribution is the design of a *Matheuristic* that embeds the *Drone Assignment and Scheduling Problem*. This heuristic is evaluated on instances with up to 100 customers. Related to this work, Schermer et al. (2019a) also study a variant of the VRPD where *en route* operations are permitted, i. e., the possibility of launching or recovering drones at some discrete points on each arc (see also (Marinelli et al., 2018)). Even though a MILP formulation is introduced, which can be used to address small-scale instances, the primary contribution is the design of an effective heuristic based on the concept of VNS.

6.3 The Traveling Salesman Problem with Drone

In this section, we provide a formal specification of the TSPD. We begin in Section 6.3.1 with a concise problem description. Afterwards, in Sections 6.3.2 and 6.3.3, we provide two compact MILP formulations of the problem.

6.3.1 Problem Definition

Suppose that there is a set of customers, each of whom must be served exactly once. In the TSPD, we assume that there is a single truck that is equipped with a single drone for performing the deliveries. Both vehicles are initially located at a central depot and we look for a feasible route of the truck and admissible use of the drone such that all customers are served and both vehicles have returned to the depot as quickly as possible, i. e., minimal makespan is achieved. Furthermore, consistent with most of the previous assumptions in the literature, we accept the following limitations in regard to the nature of the drone (Agatz et al., 2018; Murray and Chu, 2015; X. Wang et al., 2017):

- When launched from the truck, the drone can serve exactly one customer before it needs to return.
- We assume that the drone's *flight time* is limited by $\mathcal{E}$ *time units* per sortie. Furthermore, after returning to the truck, the battery of the drone is recharged (or swapped) instantaneously.
- Launching and recovering the drone might take $\bar{t}_l$ and $\bar{t}_r$ time units, respectively.
- It is only admissible to operate the drone at vertices (for a generalization, refer to, e. g., (Marinelli et al., 2018; Schermer et al., 2019a)).

In the TSPD, the locations of the depot and the customers are defined by a complete graph $\mathcal{G}(V, A)$. In this graph, V is the set of vertices and A the set of arcs. The vertex set V contains n vertices associated with the customers, labeled $V_N = \{1, \dots, n\} \subset V$ and two extra vertices 0 and $n+1$ that represent the depot location at the start and end of the tour, respectively. To sum up, $V = \{0, 1, \dots, n, n+1\}$, where $0 \equiv n+1$. We define $V_L = V \setminus \{n+1\}$ and $V_R = V \setminus \{0\}$.

A *drone-sortie* is characterized by a triple (i, w, j) as follows: the drone is launched from the truck at $i \in V_L$, performs a delivery to $w \in V_N$, and is retrieved at vertex $j \in V$ $(i \neq w, w \neq j)$. If $j = i$, i. e., when the sortie is (i, w, i), we call it a *round trip*; otherwise, if $j \in V_R, i \neq j$, we call it a *multi-leg* sortie. Finally, by the parameters d_{ij} and $\overline{d}_{ij}$ we define the distance required to travel from i to j by truck and drone, respectively. In a similar manner, we use t_{ij} and $\bar{t}_{ij}$ to define the time that is spent when traveling from i to j by either vehicle. We assume that the triangle inequality holds for these values.

6.3.2 A MILP formulation using a disjoint representation of multi-legs

For the MILP formulation of the TSPD, we use the binary decision variables according to Table 6.3.1 and the continuous variables according to Table 6.3.2.

Table 6.3.1: Binary decision variables used in the TSPD formulation.

$x_{ij} \in \{0,1\}$ $\forall i \in V_L, j \in V_R, i \neq j$	is equal to 1, iff arc (i, j) is used by the truck.
$y_{iw} \in \{0,1\}$ $\forall i \in V_L, w \in V_N, i \neq w$	is equal to 1, iff arc (i, w) is the start of a multi-leg sortie.
$\tilde{y}_{wj} \in \{0,1\}$ $\forall w \in V_N, j \in V_R, i \neq w$	is equal to 1, iff arc (w, j) is the end of a multi-leg sortie.
$z_{iw} \in \{0,1\}$ $\forall i \in V_L, w \in V_N, i \neq w$	is equal to 1, iff the drone performs a round trip (i, w, i).
$\nu_i \in \{0,1\}$ $\forall i \in V$	is equal to 1, iff the drone is available for a sortie at i.

Table 6.3.2: Continuous decision variables used in the TSPD formulation.

$a_i \in \mathbb{R}^{\geq 0}$ $\forall i \in V$	indicates the time at which i is reached by either truck or drone. If the drone is retrieved at i, this marks the lower bound at which the last vehicle (truck or drone) has arrived.
$s_i \in \mathbb{R}^{\geq 0}$ $\forall i \in V$	indicates the earliest possible departure time from i.

Using these variables, we can formulate the TSPD as in (6.1)–(6.16), where (6.1) is the objective function and the constraints are given by (6.2)–(6.16). In the following, we refer

to this model as *MILP-A*, and it draws inspiration from (Schermer et al., 2020b).

$$\min \quad s_{n+1}, \tag{6.1}$$

$$\text{s.t.} \quad \sum_{j \in V_R} x_{0j} = \sum_{i \in V_L} x_{i,n+1} = 1, \tag{6.2}$$

$$\sum_{\substack{i \in V_L, \\ i \neq h}} x_{ih} - \sum_{\substack{j \in V_R, \\ h \neq j}} x_{hj} = 0 \qquad \forall h \in V_N, \tag{6.3}$$

$$\sum_{\substack{i \in V_L, \\ i \neq w}} y_{iw} - \sum_{\substack{j \in V_R, \\ w \neq j}} \tilde{y}_{wj} = 0 \qquad \forall w \in V_N, \tag{6.4}$$

$$\sum_{\substack{i \in V_L, \\ i \neq w}} (x_{iw} + y_{iw} + z_{iw}) = 1 \qquad \forall w \in V_N, \tag{6.5}$$

$$M(x_{ij} - 1) + s_i + t_{ij} \leq a_j \qquad \forall i \in V_L, j \in V_R, i \neq j, \tag{6.6}$$

$$M(y_{iw} - 1) + s_i + \bar{t}_{iw} \leq a_w \qquad \forall i \in V_L, w \in V_N, i \neq w, \tag{6.7}$$

$$M(\tilde{y}_{wj} - 1) + s_w + \bar{t}_{wj} \leq a_j \qquad \forall w \in V_N, j \in V_R, w \neq j, \tag{6.8}$$

$$\begin{aligned} a_i + \bar{t}_r \sum_{\substack{w \in V_N, \\ w \neq i}} \tilde{y}_{wi} + \bar{t}_l \sum_{\substack{w \in V_N, \\ i \neq w}} y_{iw} \\ + \sum_{\substack{w \in V_N, \\ i \neq w}} (\bar{t}_l + \bar{t}_{iw} + \bar{t}_{wi} + \bar{t}_r) z_{iw} \leq s_i \end{aligned} \qquad \forall i \in V, \tag{6.9}$$

$$\sum_{\substack{i \in V_L, \\ i \neq w}} \bar{t}_{iw} y_{iw} + \sum_{\substack{j \in V_R, \\ w \neq j}} \bar{t}_{wj} \tilde{y}_{wj} \leq \mathcal{E} \qquad \forall w \in V_N, \tag{6.10}$$

$$\sum_{\substack{i \in V_L, \\ i \neq w}} (\bar{t}_{iw} + \bar{t}_{wi}) z_{iw} \leq \mathcal{E} \qquad \forall w \in V_N, \tag{6.11}$$

$$(1 - x_{ij}) + \nu_i - \sum_{\substack{w \in V_N, \\ w \neq i, \\ w \neq j}} y_{iw} + \sum_{\substack{w \in V_N, \\ w \neq i, \\ w \neq j}} \tilde{y}_{wj} \geq \nu_j \qquad \forall i \in V_L, j \in V_R, i \neq j, \tag{6.12}$$

$$\sum_{\substack{i \in V_L, \\ i \neq j}} x_{ij} \geq \nu_j \qquad \forall j \in V_R, \tag{6.13}$$

$$\sum_{\substack{w \in V_N, \\ i \neq w}} y_{iw} \leq \nu_i \qquad \forall i \in V_L, \tag{6.14}$$

$$\sum_{\substack{w \in V_N, \\ w \neq j}} \tilde{y}_{wj} \leq \nu_j \qquad \forall j \in V_R, \tag{6.15}$$

$$\sum_{\substack{w \in V_N, \\ i \neq w}} z_{iw} \leq \nu_i \quad \forall i \in V_L. \tag{6.16}$$

In the TSPD, the truck and the drone should serve all customers in the shortest possible time and then return to the depot, which is stated in (6.1). Constraints (6.2) and (6.3) guarantee that the flow of the truck is conserved. Moreover, constraints (6.4) sustain the flow of the drone during each multi-leg. As is stated through constraints (6.5), every customer must be served exactly once. Note that, without loss of generality, we assume that every customer is eligible to be served by the drone. If this does not apply, one can simply bind the respective variables y_{iw}, y_{wj}, and z_{iw} to zero for each customer $w \in V_N \backslash V_D$, where $V_D \subseteq V_N$ is the subset of customers that are eligible to be served by drone.

The temporal constraints are defined through (6.6)–(6.9). More precisely, (6.6) as well as (6.7) and (6.8) define the transitions of the truck and drone, respectively. Note that any subtour would contradict constraints (6.6)–(6.8). Moreover, constraints (6.9) account for the overhead times as well as the time spent performing round trips for every vertex.

The endurance constraints are given by (6.10) for multi-leg and (6.11) for round trip sorties and incorporate the time during which the drone is in flight. Constraints (6.12) and (6.13) account for the availability of the drone at each vertex. In particular, if an arc (i, j) is used by the truck, constraints (6.12) provide an upper bound on the availability of the drone at j, i. e., ν_j. If the drone is available at i, i. e., $\nu_i = 1$, the upper bound $\nu_j \leq 1$ will only hold if either the drone is not launched at i or if it is immediately recovered at j; otherwise, ν_j will be bound to zero. In contrast, if the drone is unavailable at i, i. e., $\nu_i = 0$, then there must be a recovery at j (from a sortie that began before i) for the drone to be available again. Moreover, constraints (6.13) are logical constraints: the drone can only be available at vertices that are visited by the truck. Finally, with these limits on the availability of the drone in place, constraints (6.14)–(6.16) guarantee that the drone can only be launched from and recovered at vertices where it is also available.

Even though MILP (6.1)–(6.16) formulates the TSPD, some effective VIEQs exist that significantly strengthen the linear relaxation (refer to (Dell'Amico et al., 2021b; Schermer et al., 2019b; Schermer et al., 2020b)). For the purpose of this work, we integrate them into *MILP-A* as follows:

$$\sum_{i \in V_L} \sum_{\substack{j \in V_R, \\ i \neq j}} t_{ij} x_{ij} + \sum_{i \in V_L} \sum_{\substack{w \in V_N, \\ i \neq w}} \Big(\left(\bar{t}_l + \bar{t}_{iw} + \bar{t}_{wi} + \bar{t}_r \right) z_{iw} + \bar{t}_l y_{iw} \Big) + \sum_{w \in V_N} \sum_{\substack{j \in V_R, \\ w \neq j}} \bar{t}_r \tilde{y}_{wj} \leq s_{n+1}, \tag{6.17}$$

$$\sum_{i \in V_L} \sum_{\substack{w \in V_N, \\ i \neq w}} \Big(\big(\bar{t}_l + \bar{t}_{iw} + \bar{t}_{wi} + \bar{t}_r\big)\, z_{iw} + \big(\bar{t}_l + \bar{t}_{iw}\big)\, y_{iw} \Big) + \sum_{w \in V_N} \sum_{\substack{j \in V_R, \\ w \neq j}} \big(\bar{t}_{wj} + \bar{t}_r\big)\, \tilde{y}_{wj} \leq s_{n+1}. \tag{6.18}$$

Here, constraints (6.17) and (6.18) account for the time spent traveling on arcs by the truck and drone, respectively. Moreover, these constraints incorporate the time when the truck and drone perform their actions in a *synchronized* manner, i. e., when the drone performs round trips while the truck remains stationary (waiting for the drone to return) and whenever the overhead times $\bar{t}_l$ and $\bar{t}_r$ are applied.

6.3.3 A MILP formulation using a joint representation of multi-legs

In *MILP-A*, we use the variables y_{iw} and $\tilde{y}_{wj}$ (according to Table 6.3.1) to differentiate between drone arcs that are part of outbound and inbound flights (each part of a multi-leg). In principle, instead of using two disjoint sets, it is also possible to use a single set of variables that simply state if an arc is used at all during a multi-leg flight. Hence, a smaller number of columns is required for formulating the problem. To the best of our knowledge, at this point, there are no models using a two-index formulation for the TSPD (or related problems) in the literature that investigated this possibility.

To this end, we might use the binary decision variables x_{ij}, z_{iw}, and ν_i as well as the continuous decision variables a_i and s_i according to Tables 6.3.1 and 6.3.2, respectively. Moreover, we introduce the additional binary decision variables according to Table 6.3.3. Then, the alternative MILP formulation of the TSPD is given by (6.1)–(6.3), (6.6), (6.11), (6.13), (6.16), and (6.19)–(6.31). We denote this formulation by *MILP-B*.

Table 6.3.3: Additional binary decision variables used in the joint drone arc formulation.

$y_{ij} \in \{0,1\}$ $\forall i \in V_L, j \in V_R, i \neq j$	is equal to 1, iff arc (i,j) is part of a multi-leg sortie (either outbound or inbound).
$r_i \in \{0,1\}$ $\forall i \in V$	indicates if the drone is recovered after a multi-leg sortie at i.

$$\sum_{\substack{i \in V_L, \\ i \neq w}} \Big(x_{iw} + (y_{iw} + z_{iw}) \Big) - r_w = 1 \qquad \forall w \in V_N, \tag{6.19}$$

$$\sum_{\substack{i \in V_L, \\ i \neq w}} y_{iw} - \sum_{\substack{j \in V_R, \\ w \neq j}} y_{wj} - \nu_w \leq 0 \qquad \forall w \in V_N, \tag{6.20}$$

$$\nu_w + \sum_{\substack{i \in V_L, \\ i \neq w}} y_{iw} - \sum_{\substack{j \in V_R, \\ w \neq j}} y_{wj} \geq 0 \qquad \forall w \in V_N, \tag{6.21}$$

$$\sum_{\substack{i \in V_L, \\ i \neq j}} (x_{ij} + y_{ij}) - 1 \leq r_j \qquad \forall j \in V_R, \tag{6.22}$$

$$\sum_{\substack{i \in V_L, \\ i \neq j}} x_{ij} \geq r_j \qquad \forall j \in V_R, \tag{6.23}$$

$$\sum_{\substack{i \in V_L, \\ i \neq j}} y_{ij} \geq r_j \qquad \forall j \in V_R, \tag{6.24}$$

$$r_j \leq \nu_j \qquad \forall j \in V_R, \tag{6.25}$$

$$M(y_{ij} - 1) + s_i + \bar{t}_{ij} \leq a_j \qquad \forall i \in V_L, j \in V_R, i \neq j, \tag{6.26}$$

$$\begin{aligned} a_i + \bar{t}_r r_i + \bar{t}_l(\nu_i - 1 + \sum_{\substack{w \in V_N, \\ i \neq w}} y_{iw}) \\ + \sum_{\substack{w \in V_N, \\ i \neq w}} (\bar{t}_l + \bar{t}_{iw} + \bar{t}_{wi} + \bar{t}_r) z_{iw} \leq s_i \end{aligned} \qquad \forall i \in V, \tag{6.27}$$

$$\sum_{\substack{i \in V_L, \\ i \neq w}} \bar{t}_{iw} y_{iw} + \sum_{\substack{j \in V_R, \\ w \neq j}} \bar{t}_{wj} y_{wj} \leq \mathcal{E} + M r_w \qquad \forall w \in V_N, \tag{6.28}$$

$$2 \cdot (1 - x_{ij}) + \nu_i - \sum_{\substack{w \in V_N, \\ w \neq i, \\ w \neq j}} y_{iw} + r_j \geq \nu_j \qquad \forall i \in V_L, j \in V_R, i \neq j, \tag{6.29}$$

$$y_{ij} \leq \nu_i + r_j \qquad \forall i \in V_L, j \in V_R, \tag{6.30}$$

$$\sum_{\substack{j \in V_R, \\ i \neq j}} y_{ij} \leq 1 \qquad \forall i \in V_L. \tag{6.31}$$

Through the objective function (6.1) and constraints (6.2)–(6.3), we still aim at minimizing the makespan while preserving the flow of the truck. Moreover, the updated model requires several new constraints that restrict the binary variables r_i. Constraints (6.19) state that every customer must be served exactly once. If the drone is recovered at a vertex j, i. e., $r_j = 1$, there are two incoming arcs at j: one for the truck and drone, respectively. In this case, we subtract the binary variable r_j to achieve an equilibrium.

Constraints (6.20)–(6.21) provide conservation of flow for the drone arcs. More precisely, if the drone is not available at some vertex w, i. e., $\nu_w = 0$, then the flow must be conserved and both constraints force the number of incoming (drone) arcs to be equal to the number of outgoing arcs. On the other hand, flow must not necessarily be conserved if the drone is

available at w, i. e., if $\nu_w = 1$ (see also Figure 6.3.1). Constraints (6.22)–(6.24) determine the value on the variables r_j: these constraints force the variables to take value 1 if there is exactly one incoming truck and drone arc and to value 0 otherwise. Moreover, constraints (6.25) are valid cuts that are logically implied by the remaining model. These constraints state that the drone can only be recovered if it also available at the same vertex.

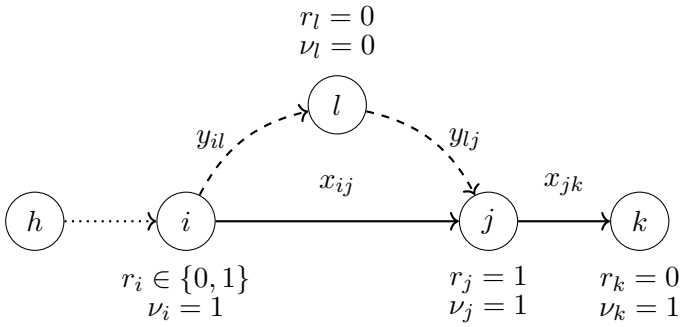

Figure 6.3.1: A part of a feasible TSPD solution: the variables r_i are used to indicate if a drone is recovered. If the drone is available, i. e., where $\nu_i = 1$, flow must not necessarily be conserved: in these situations, there can be an incoming (outgoing) drone arc without an outgoing (incoming) drone arc. Note that the case of $r_i = 1, \nu_i = 0$ does not exist $\forall i \in V$.

In this model, the temporal constraints for the truck defined in (6.6) still hold. Moreover, it suffices to use constraints (6.26) and (6.27) for the transitions of the drone. The endurance constraints are given by (6.11) and (6.28) for round trip and multi-leg sorties, respectively. Note that, in this model, (6.28) now requires an additional Mr_w term, i. e., the constraint only applies for two consecutive drone arcs that visit a customer (and not the truck) in between (see also Figure 6.3.1). Finally, constraints (6.13), (6.16), and (6.29)–(6.31) are used to determine the availability of the drone and to restrict operations accordingly. If the formulation presented in this section is used, before integrating the VIEQs (6.17) and (6.18) into *MILP-B*, slight adjustments are necessary. In particular, due to the nature of the variables y_{ij}, it is no longer possible to associate the overhead time $\bar{t}_l$ and $\bar{t}_r$ with outgoing and incoming drone arcs, respectively. However, during each multi-leg, the drone is launched and recovered exactly once. Hence, the total number of multi-leg arcs y_{ij} with value 1 will always be even, and thus we can adapt these VIEQs as follows:

$$\sum_{i \in V_L} \sum_{\substack{j \in V_R, \\ i \neq j}} t_{ij} x_{ij} + \sum_{i \in V_L} \sum_{\substack{w \in V_N, \\ i \neq w}} \left(\left(\bar{t}_l + \bar{t}_{iw} + \bar{t}_{wi} + \bar{t}_r \right) z_{iw} \right) + \sum_{i \in V_L} \sum_{\substack{j \in V_R, \\ i \neq j}} \frac{\bar{t}_l + \bar{t}_r}{2} y_{ij} \leq s_{n+1}, \tag{6.32}$$

$$\sum_{i \in V_L} \sum_{\substack{w \in V_N, \\ i \neq w}} \left(\left(\bar{t}_l + \bar{t}_{iw} + \bar{t}_{wi} + \bar{t}_r \right) z_{iw} \right) + \sum_{i \in V_L} \sum_{\substack{j \in V_R, \\ i \neq j}} \left(\frac{\bar{t}_l + \bar{t}_r}{2} + \bar{t}_{ij} \right) y_{ij} \leq s_{n+1}. \tag{6.33}$$

6.3.4 Discussion

As we will see in Section 6.5, only small-sized instances can be solved through *MILP-A* and *MILP-B*. In particular, several constraints involve large M-coefficients; hence, the linear relaxation is quite weak, which results in a slow convergence of the LB. Nevertheless, the proposed VIEQs strengthen the linear relaxation significantly and, according to Lemma 6.1, these VIEQs are almost *tight*.

Lemma 6.1. In an optimal solution, the slack value between left-hand-side of the VIEQs (6.17) and (6.32) and the makespan, i. e., s_{n+1}, is equal to the total time that the truck remains stationary at vertices while waiting for the drone to arrive (from a multi-leg sortie).

Proof. The total operating time of the truck can be decomposed into three components as follows:

1. First, the *travel time*, i. e., the time spent moving on arcs.
2. Second, the *synchronized waiting time*, i. e., the time shared by the truck and drone in a coordinated manner while performing interdependent actions:
   - during launch and retrieval of the drone (whenever overhead times are applied),
   - whenever the truck is waiting for a drone to return from a round trip.
3. Finally, the *non-synchronized waiting time*, i. e., the time spent at each vertex by the truck while waiting for the drone to join (arrive from a multi-leg).

Based on the problem definition in Section 6.3, no other mode of operation does exist for the truck. Note that the VIEQs (6.17) and (6.32) account for the travel time and the synchronized waiting time. Therefore, the slack must be equal to the non-synchronized waiting time. This concludes the proof of Lemma 6.1. ■

It is also worth to mention that, based on VIEQs (6.18) and (6.33), a similar lemma might be proposed for the drone. However, compared to Lemma 6.1, one further mode of operation exists for which these VIEQs do not account for: the time during which the drone is carried along an arc by the truck.

6.4 A Branch-and-Cut approach for the TSPD

The MILP formulations presented in Section 6.3 might be solved by any standard MILP solver, e. g., Gurobi Optimizer (Gurobi Optimization, LLC, 2023). However, in order to gain more efficiency, we introduce a third MILP formulation that permits a customized design of the B&C algorithm. This proposal is motivated by three main observations:

1. In optimal solutions, typically, the travel time and synchronized waiting time have the largest contribution to the makespan and, due to Lemma 6.1, they can be approximated well through the linear constraints (6.17) and (6.32).

2. The temporal constraints in *MILP-A* and *MILP-B*, presented in Section 6.3, introduce several M-coefficients (as is the case in related models, e. g., (Dell'Amico et al., 2019; Dell'Amico et al., 2021b; Ha et al., 2018; Murray and Chu, 2015)). Consequently, the quality of the linear relaxation is quite weak and the convergence of the LB is slow. However, these formulations are *compact*, i. e., of polynomial size. In fact, in our formulations, only $\mathcal{O}(|V|^2)$ variables and constraints are required for formulating the problem.

3. In the set-covering based ILP formulation proposed by Agatz et al. (2018), the so-called *operations* are used as the essential building block. Operations already explicitly incorporate the *non-synchronized waiting times*; therefore, no big-M constraints are necessary. However, there is an exponential number of operations for a given instance; hence, a static formulation does not scale very well to a B&C framework.

Based on these observations, we introduce a MILP formulation that attempts to leverage the strengths of the MILP and ILP approaches listed above through Lemma 6.1. Before we present the model, we adapt the concept of an *operation* o for our purpose (based on (Agatz et al., 2018)). Let an operation o contain a feasible sequence (w. r. t. Section 6.3) of one or more truck-arcs that we call a *directed path* P. Each operation o contains exactly one *multi-leg* drone sortie that starts at the beginning of P, targets a customer (not contained in P), and concludes at the end of P. We introduce the notation $\delta(o) \in V_R$ to refer to the vertex at the end of the operation o (the path P). Finally, we denote the cardinality of an operation by $|o|$ and define it as the total number of directed arcs that are contained. Figure 6.4.1 illustrates an example of this concept. Let O denote the set of all feasible operations and take note that this set grows exponentially with $|V|$.

In this section, we introduce a MILP formulation of the TSPD that features an exponential number of *constraints*. As a result, we propose these constraints to be separated dynamically, i. e., through a B&C framework. For introducing this formulation, we use

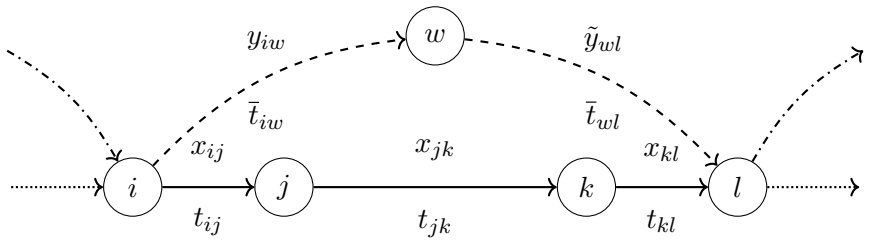

Figure 6.4.1: An example of an operation $o = \{x_{ij}, x_{jk}, x_{kl}, y_{iw}, \tilde{y}_{wl}\}$ with a cardinality of $|o| = 5$ that ends at $\delta(o) = l \in V_R$. The directed path P is defined by the solid arcs.

the decision variables according to Tables 6.3.1 and 6.4.1. Using this notation, the MILP generally follows *MILP-A* and is given by (6.2)–(6.5), (6.10)–(6.16), (6.34)–(6.39), and the *Miller-Tucker-Zemlin* (MTZ) constraints (6.42)–(6.46). Due to the B&C nature, we call this formulation *MILP-BC*.

$$\min \quad \tau + \omega, \tag{6.34}$$

$$\text{s. t.} \quad \sum_{i \in V_L} \sum_{\substack{j \in V_R, \\ i \neq j}} t_{ij} x_{ij} + \sum_{r \in V_R} w_r^t = \tau, \tag{6.35}$$

$$\sum_{i \in V_L} \sum_{\substack{w \in V_N, \\ i \neq w}} t_{iw} y_{iw} + \sum_{w \in V_N} \sum_{\substack{j \in V_R, \\ w \neq j}} t_{wj} \tilde{y}_{wj} + \sum_{r \in V_R} w_r^d \leq \tau, \tag{6.36}$$

$$\sum_{i \in V_L} \sum_{\substack{w \in V_N, \\ i \neq w}} (\bar{t}_l + \bar{t}_{iw} + \bar{t}_{wi} + \bar{t}_r) z_{iw} + \bar{t}_l \sum_{i \in V_L} \sum_{\substack{w \in V_N, \\ i \neq w}} y_{iw} + \bar{t}_r \sum_{w \in V_N} \sum_{\substack{j \in V_R, \\ w \neq j}} \tilde{y}_{wj} = \omega. \tag{6.37}$$

Table 6.4.1: Additional variables used for the formulation *MILP-BC*.

$\tau \in \mathbb{R}^{\geq 0}$	defines the least upper bound on the *travel time* and *non-synchronized waiting time* for the truck and drone.
$\omega \in \mathbb{R}^{\geq 0}$	indicates the total *synchronized waiting time* shared by the truck and drone.
$w_j^t \in \mathbb{R}^{\geq 0}$ $\forall j \in V_R$	is the non-synchronized waiting time of the truck at vertex j (while waiting for the drone to arrive).
$w_j^d \in \mathbb{R}^{\geq 0}$ $\forall j \in V_R$	defines the non-synchronized waiting time of the drone at vertex j (while waiting for the truck to arrive).
$u_i \in \mathbb{R}^{\geq 0}$ $\forall i \in V$	sets the position of vertex i in the delivery sequence.

Let us explain the model in more detail. Generally, this formulation closely follows *MILP-A*. In fact, we only replace the objective function and drop all the temporal constraints (and thus, M-coefficients). Hence, in *MILP-BC*, the objective function is given

by (6.34) and consists of two terms: τ and ω (note that a similar objective is used in (Dell'Amico et al., 2019; Dell'Amico et al., 2021b)). The value τ has two bounds which are defined through constraints (6.35) and (6.36) for the truck and drone, respectively. These constraints account for the time spent traveling on arcs as well as the non-synchronized waiting times w_r^t and w_r^d at each vertex $r \in V_R$. Note that the bound is binding in case of constraint (6.35) while (6.36) only provides a LB on τ (refer to Lemma 6.1 and comments). The total synchronized waiting time ω is defined through equation (6.37). This constraint accounts for the time spent performing round trips and the total overhead time.

For each operation $o \in O$, constraints (6.38) and (6.39) provide LBs on the non-synchronized waiting times $w_{\delta(o)}^t$ for the truck and $w_{\delta(o)}^d$ for the drone:

$$\phi(o) \left(1 + \sum_{x_{ij} \in o} x_{ij} + \sum_{y_{iw} \in o} y_{iw} + \sum_{\tilde{y}_{wj} \in o} \tilde{y}_{wj} - |o| \right) \leq w_{\delta(o)}^t \qquad \forall o \in O, \tag{6.38}$$

$$\theta(o) \left(1 + \sum_{x_{ij} \in o} x_{ij} + \sum_{y_{iw} \in o} y_{iw} + \sum_{\tilde{y}_{wj} \in o} \tilde{y}_{wj} - |o| \right) \leq w_{\delta(o)}^d \qquad \forall o \in O. \tag{6.39}$$

To explain these constraints, let us assume that an operation o is currently part of the solution, i. e., the parentheses in (6.38) and (6.39) evaluate to 1; then, for the given operation o, the non-synchronized waiting times at the vertex $\delta(o) \in V_R$ are defined by the functions $\phi(o)$ and $\theta(o)$ for the truck and drone, respectively. These functions determine the non-synchronized waiting times at the vertex $\delta(o) \in V_R$ as follows:

- If one vehicle (either the truck or the drone) arrives at $\delta(o)$ at the same time as or after the other one, then the non-synchronized waiting time (either $\phi(o)$ or $\theta(o)$) is equal to 0 as it does not have to wait for synchronization.
- In contrast, if one vehicle (either the truck or the drone) arrives at $\delta(o)$ before the other one, then the non-synchronized waiting time (either $\phi(o)$ or $\theta(o)$) is equal to the difference in arrival time, i. e., the time spent waiting.

As a result, the functions $\phi(o)$ and $\theta(o)$ are embedded within the scheme shown in Algorithm 6.1. Indeed, using a state-of-the-art solver, this algorithm can be applied through a *callback* whenever a new feasible candidate incumbent solution is identified. In more detail, for each operation o that is contained in the current solution:

1. First, the travel times of the truck and drone are calculated. Here, we do not account for the overhead times, as they are already included in the term ω in the objective function (see objective (6.34) and constraint (6.37)).

2. Depending on which vehicle arrives first, either cut (6.38) or (6.39) is added using the difference in travel time, i. e., the non-synchronized waiting time at $\delta(o)$ of either the truck or the drone, given that the operation o is part of the solution.

Algorithm 6.1 Constraint generation scheme (applied at each MIP incumbent node).

for each operation $o \in O$ contained in the current solution **do**
  Calculate $t_t = \sum_{x_{ij} \in o} t_{ij} x_{ij}$ and $\bar{t}_d = \sum_{y_{iw} \in o} \bar{t}_{iw} y_{iw} + \sum_{\tilde{y}_{wj} \in o} \bar{t}_{wj} \tilde{y}_{wj}$
  if $t_t < \bar{t}_d$ **then** ▷ In operation o, the truck arrives before the drone.
    $\phi(o) = (\bar{t}_d - t_t)$
    Add cut (6.38) for operation o
  else if $t_t > \bar{t}_d$ **then** ▷ In operation o, the drone arrives before the truck.
    $\theta(o) = (t_t - \bar{t}_d)$
    Add cut (6.39) for operation o
  end if
end for

Using this algorithm, the corresponding constraints, i. e., (6.38) and (6.39), are added only as necessary or not before needed to the model. This scheme is guaranteed to converge towards the optimal solution because of Lemmas 6.1 and 6.2 as well as Proposition 6.1, described in what follows.

Lemma 6.2. An operation $o \in O$ is never a subset of any other operation $o' \in O, o \neq o'$.

Proof. By contradiction, assume that $\exists o' \in O$ such that $o \subset o'$ and $o \neq o'$ (the case of $o' \subset o$ can be treated analogously; hence, we skip it). We differentiate the following cases:

1. either o' contains at least one more drone sortie,
2. or o' contains at least one more arc that is used by the vehicle.

The first case contradicts the definition of an operation as at most one (multi-leg) drone sortie can be contained in each operation. For the second case, assume that we were to add an additional arc x_{ij} to the path P. For the path to remain feasible, the only possible location to append one further arc is at the end or the beginning of P. However, this would also contradict the definition of an operation as the drone would no longer be launched (retrieved) at the start (end) of the path. This concludes the proof of Lemma 6.2. ■

Proposition 6.1. The constraints (6.38)–(6.39) are *optimality cuts*, i. e., they are:

1. *binding* if the operation $o \in O$ is contained in the current solution,
2. *feasible* for all other $o' \in O, o' \neq o$, otherwise.

Proof. The first statement follows directly from the definition of an operation at the beginning of Section 6.4 and the fact that constraints (6.38)–(6.39) are *indicator* constraints, i. e., they become active if and only if o is part of the solution. In this case, i. e., if the operation o is part of the solution, constraints (6.38)–(6.39) provide valid LBs on w_r^t and w_r^d for the truck and drone, respectively.
The second statement is true because of Lemma 6.2. In fact, since o is never a subset of any other o', the cut is non-binding and feasible for all other operations. This completes the proof of Proposition 6.1. ■

As an example, let us consider the operation $o \in O$ shown in Figure 6.4.1. Here, the operation is given as $o = \{x_{ij}, x_{jk}, x_{kl}, y_{iw}, \tilde{y}_{wj}\}$ with a cardinality of $|o| = 5$. In this example, we assume that the arrival time of the truck is $t_t = t_{ij} + t_{jk} + t_{kl}$ and the arrival time of the drone is $\bar{t}_d = \bar{t}_{iw} + \bar{t}_{wj}$. As the overhead times $\bar{t}_l$ and $\bar{t}_r$ are shared by both vehicles and incorporated in the synchronized waiting time ω, we do not have to consider them here. If $t_t > \bar{t}_d$, then the drone has to wait for $\theta(o) = (t_t - \bar{t}_d)$ time units at $\delta(o) = l \in V_R$ and the following drone-cut can be added (refer to Algorithm 6.1):

$$(t_t - \bar{t}_d)(x_{ij} + x_{jk} + x_{kl} + y_{iw} + \tilde{y}_{wj} - 4) \leq w_l^d. \tag{6.40}$$

Otherwise, if $t_t < \bar{t}_d$, the truck has to wait for $\phi(o) = (\bar{t}_d - t_t)$ time units at l for the drone to arrive. Thus, we can add the following truck-cut:

$$(\bar{t}_d - t_t)(x_{ij} + x_{jk} + x_{kl} + y_{iw} + \tilde{y}_{wj} - 4) \leq w_l^t. \tag{6.41}$$

Note that, in the (rare) event where $\bar{t}_d = t_t$, the arrival of the truck and drone is perfectly synchronous; hence, no cut must be added as neither vehicle has to wait for the other one.

Finally, the temporal constraints (6.6)–(6.9) that also prevent subtours are no longer used in the model *MILP-BC*. As a result, we adapt and use the family of MTZ constraints (refer to (Miller et al., 1960)) as follows:

$$u_0 = 0, \tag{6.42}$$

$$1 \leq u_i \leq n + 1 \qquad \forall i \in V_R, \tag{6.43}$$

$$u_i - u_j + 1 \leq (n+1)(1 - x_{ij}) \quad \forall i \in V_L, j \in V_R, i \neq j, \tag{6.44}$$

$$u_i - u_w + 1 \leq (n+1)(1 - y_{iw}) \quad \forall i \in V_L, w \in V_N, i \neq w, \tag{6.45}$$

$$u_w - u_j + 1 \leq (n+1)(1 - \tilde{y}_{wj}) \quad \forall w \in V_N, j \in V_R, w \neq j. \tag{6.46}$$

This concludes the description of *MILP-BC*. However, in optimal solutions, many op-

erations are of cardinality $|o| = 3$, i. e., the truck does not serve any additional customer in between launch and recovery of the drone (see also Figure 6.5.3). Even though, the constraint generation scheme proposed in Algorithm 6.1 also accounts for such operations, it might be beneficial to introduce some additional VIEQs that match to these types of operations. Therefore, we propose constraints (6.47) and (6.48) as VIEQs to *MILP-BC* and call the resulting formulation *MILP-BC-VI*:

$$\begin{aligned} &\sum_{\substack{w \in V_N, \\ i \neq w, \\ w \neq j}} \left(\bar{t}_{iw} y_{iw} + \bar{t}_{wj} \tilde{y}_{wj}\right) - t_{ij} x_{ij} + \\ M \cdot (x_{ij} + &\sum_{\substack{w \in V_N, \\ i \neq w, \\ w \neq j}} (y_{iw} + \tilde{y}_{wj}) - 3) \leq w_j^t \quad \forall i \in V_L, j \in V_R, \end{aligned} \tag{6.47}$$

$$\begin{aligned} &t_{ij} x_{ij} - \sum_{\substack{w \in V_N, \\ i \neq w, \\ w \neq j}} \left(\bar{t}_{iw} y_{iw} - \bar{t}_{wj} \tilde{y}_{wj}\right) \\ +M \cdot (x_{ij} + &\sum_{\substack{w \in V_N, \\ i \neq w, \\ w \neq j}} (y_{iw} + \tilde{y}_{wj}) - 3) \leq w_j^d \quad \forall i \in V_L, j \in V_R. \end{aligned} \tag{6.48}$$

In particular, constraints (6.47) and (6.48) account for the waiting time of the truck and drone, respectively, at each vertex j, if a sortie o of cardinality $|o| = 3$ is performed (i. e., if the expression with coefficient M evaluates to 0). Note that, in these cases, a sufficiently large value of M is relatively small; in fact, $M = \arg\max_{i,j \in V} 2 \cdot \bar{t}_{ij}$ and $M = \arg\max_{i,j \in V} t_{ij}$ are sufficient for the value in (6.47) and (6.48), respectively.

Finally, as a general remark, note that the variables w_j^t and w_j^d might also prove useful when some constraints are imposed that might limit the maximum waiting time of the truck or drone or when it is of interest to associate a specific cost with waiting (as it is the case in, e. g., (Ha et al., 2018)).

6.5 Computational Experiments and their Numerical Results

All experiments were performed on an Intel Xeon E5-2640v3 Processor with access to 8 GB of RAM. As MILP solver, we chose Gurobi Optimizer 9.0.0 (Gurobi Optimization, LLC, 2023) with a runtime limit of one hour. For the design of the B&C algorithm introduced in Section 6.4, a natural choice is to use the *callback* framework of the solver, i. e., we apply the cut scheme proposed in Algorithm 6.1 at each new *Mixed-Integer Programming* (MIP) incumbent node in the B&C tree. Through this process, we rely on an established

external framework for primal heuristics, the generation of general MILP cuts, branching-decisions, and so forth (Fischetti et al., 2016; Fischetti et al., 2017). As we have outlined towards the end of Section 6.2, there are no commonly accepted problem assumptions. As a consequence, we selected three different types of benchmark instances to conduct our computational experiments. In Section 6.5.1, we begin with instances introduced by Bouman et al. (2018b) (used in (Agatz et al., 2018)). Afterwards, in Section 6.5.2, we test on the instances proposed by Poikonen et al. (2019). Finally, in Section 6.5.3, we conclude our experiments using instances of Murray and Chu (2015).

6.5.1 Instances proposed by Bouman et al.

In this section, we describe our computational experiments on the instances proposed by Bouman et al. (2018b), which have been used in, e. g., (Agatz et al., 2018; Bouman et al., 2018a; Poikonen et al., 2019). We begin in Section 6.5.1.1 by reporting the experimental setup. Afterwards, the numerical results are discussed in Section 6.5.1.2.

6.5.1.1 Experimental Setup

Bouman et al. (2018b) provide a wide range of instances with different numbers of vertices. For our experiments, we use instances with $n \in \{9, 19\}$ customers (i. e., $n+1 \in \{10, 20\}$ vertices), where the coordinates were drawn uniformly and independently from $\{0, \dots, 100\}$. It is assumed that both vehicles follow the Euclidean distance metric and the drone's endurance is limited by flight distance. In particular, the velocity of the drone is $\alpha \in \{1, 2, 3\}$ times that of the truck (which moves at a velocity of one distance unit per time unit). Moreover, the overhead times are assumed to be negligible, i. e., $\overline{t}_l = \overline{t}_r = 0$. Regarding the endurance, let $\mathcal{E} = \arg\max_{i,j \in V} \overline{d}_{ij}$ be the edge with maximum length in the graph G. Using this value of $\mathcal{E}$, for each instance, the drone's range of operation is restricted by a parameter $R \in \{20\%, 40\%, 60\%, 100\%, 150\%, 200\%\}$ that is multiplied by $\mathcal{E}$. In order to incorporate this parameter into our models, for *MILP-A* and *MILP-BC*, constraints (6.10) are replaced as follows:

$$\sum_{\substack{i \in V_L, \\ i \neq w}} \overline{d}_{iw} y_{iw} + \sum_{\substack{j \in V_R, \\ w \neq j}} \overline{d}_{wj} \tilde{y}_{wj} \leq R \cdot \mathcal{E}, \quad \forall w \in V_N. \tag{6.49}$$

Moreover, for *MILP-B*, we adjust constraints (6.28) as follows:

$$\sum_{\substack{i \in V_L, \\ i \neq w}} \overline{d}_{iw} y_{iw} + \sum_{\substack{j \in V_R, \\ w \neq j}} \overline{d}_{wj} y_{wj} \leq R \cdot \mathcal{E} + M r_w, \quad \forall w \in V_N. \tag{6.50}$$

Finally, for all formulations, the endurance for round trips defined in (6.11) is instead restricted as follows:

$$\sum_{\substack{i \in V_L, \\ i \neq w}} \left(\overline{d}_{iw} + \overline{d}_{wi}\right) z_{iw} \leq R \cdot \mathcal{E}, \quad \forall w \in V_N. \tag{6.51}$$

6.5.1.2 Numerical Results

The resulting set of 360 instances was solved using formulations *MILP-A*, *MILP-B* (with their respective VIEQs) as well as with *MILP-BC* and *MILP-BC-VI*, yielding a total of 1440 experiments. Table 6.5.1 provides detailed results on the performance of each approach. Based on the instance size n, the relative velocity α, and relative endurance R, this table shows the MIP gap and runtime (averaged over 10 instances) as well as the share of instances solved to optimality by each approach. Moreover, for *MILP-BC* and *MILP-BC-VI*, the average number of cuts of types (6.38) and (6.39), which are added during the search, are shown.

Based on the results presented in Table 6.5.1, we can make the following observations. The formulation *MILP-A* can be used to effectively solve instances with $n = 9$ customers, requiring no more than 2 seconds on average. In contrast, the alternative formulation *MILP-B* is slightly worse in terms of average runtime. Beyond this size, i. e., for $n = 19$, the performance of *MILP-A* is noticeably better: 16.1% of these instances can be solved optimally with an average remaining MIP gap of 16.1%. Overall, the B&C algorithm that utilizes *MILP-BC* shows promising performance. While *MILP-A* is on average slightly faster for $n = 9$ customers, the B&C approach shows a slightly superior performance in tackling instances with $n = 19$ customers. To be more precise, through *MILP-BC* we can solve a slightly larger share (22.2%) to proven optimality with an average remaining MIP gap of 16.2%. In contrast, *MILP-BC-VI* solves fewer instances to optimality (19.2%); however, with an improved MIP gap of 15.7%. A closer look at these two formulations reveals that the cuts of type (6.38) and (6.39), which are added during the search, are about half if *MILP-BC-VI* is used. Table 6.5.1 also displays the quality of the relative root relaxation $\%_r$ associated with each formulation. We provide this information by using the following Formula (6.52) as a performance metric:

$$\%_r := \frac{\text{root relaxation objective value}}{\text{objective value of the best feasible solution after runtime}} \tag{6.52}$$

Overall, the quality of the root relaxation associated with *MILP-B* is noticeably worse. In fact, on average, the relative root relaxation of *MILP-A* is about 13% better which also helps to explain why *MILP-A* performs significantly better. When we look at the

Table 6.5.1: Average MIP gap, average runtime (in seconds), the share of instances solved to optimality (opt.), and the average relative root relaxation $\%_r$ depending on the number of customers n, the relative velocity α, and the relative endurance R on the (Bouman et al., 2018b) instance set (10 instances per row). For *MILP-BC* and *MILP-BC-VI*, this table shows also the average number of constraints (6.38) and (6.39) added as cuts.

Instance		*MILP-A*				*MILP-B*				*MILP-BC*				*MILP-BC-VI*			
n	α	R	gap	time	opt.	$\%_r$	gap	time	opt.	$\%_r$	gap	time	opt.	$\%_r$	cuts	gap	time	opt.	$\%_r$	cuts
9	1	20%	0.0%	0.2	100.0%	71.6%	0.0%	0.2	100.0%	69.7%	0.0%	0.2	100.0%	72.8%	4.5	0.0%	0.2	100.0%	72.9%	4.5
		40%	0.0%	0.3	100.0%	63.5%	0.0%	1.0	100.0%	37.1%	0.0%	0.3	100.0%	65.0%	23.4	0.0%	0.4	100.0%	66.9%	10.3
		60%	0.0%	1.6	100.0%	47.1%	0.0%	3.7	100.0%	33.7%	0.0%	1.7	100.0%	47.7%	38.2	0.0%	2.7	100.0%	48.1%	21.6
		100%	0.0%	5.2	100.0%	45.2%	0.0%	9.3	100.0%	35.1%	0.0%	5.2	100.0%	45.4%	162.3	0.0%	7.9	100.0%	45.4%	106.4
		150%	0.0%	6.4	100.0%	47.1%	0.0%	7.7	100.0%	36.6%	0.0%	7.3	100.0%	47.4%	531.8	0.0%	10.8	100.0%	47.4%	365.4
		200%	0.0%	6.0	100.0%	47.1%	0.0%	13.6	100.0%	36.6%	0.0%	7.3	100.0%	47.4%	529.4	0.0%	9.0	100.0%	47.4%	325.3
	2	20%	0.0%	0.2	100.0%	71.5%	0.0%	0.2	100.0%	69.4%	0.0%	0.1	100.0%	72.7%	2.3	0.0%	0.1	100.0%	72.7%	1.7
		40%	0.0%	0.3	100.0%	61.3%	0.0%	0.6	100.0%	33.5%	0.0%	0.3	100.0%	64.0%	8.3	0.0%	0.3	100.0%	65.1%	5.6
		60%	0.0%	0.7	100.0%	43.2%	0.0%	2.0	100.0%	26.6%	0.0%	0.9	100.0%	44.0%	21.6	0.0%	1.7	100.0%	44.7%	14.2
		100%	0.0%	1.7	100.0%	40.6%	0.0%	3.4	100.0%	29.5%	0.0%	2.3	100.0%	40.8%	67.6	0.0%	3.6	100.0%	41.0%	36.3
		150%	0.0%	3.1	100.0%	42.3%	0.0%	3.9	100.0%	31.7%	0.0%	3.8	100.0%	42.6%	152.4	0.0%	4.7	100.0%	42.8%	69.3
		200%	0.0%	2.7	100.0%	42.5%	0.0%	4.5	100.0%	31.9%	0.0%	4.2	100.0%	42.8%	186.9	0.0%	4.2	100.0%	43.0%	60.3
	3	20%	0.0%	0.2	100.0%	71.1%	0.0%	0.2	100.0%	69.0%	0.0%	0.1	100.0%	72.6%	3.3	0.0%	0.2	100.0%	72.6%	3.3
		40%	0.0%	0.3	100.0%	59.6%	0.0%	0.7	100.0%	32.1%	0.0%	0.3	100.0%	62.8%	9.9	0.0%	0.4	100.0%	63.5%	7.8
		60%	0.0%	0.7	100.0%	40.5%	0.0%	1.4	100.0%	22.6%	0.0%	0.9	100.0%	41.8%	20.7	0.0%	1.2	100.0%	42.6%	16.3
		100%	0.0%	1.1	100.0%	35.7%	0.0%	1.7	100.0%	24.7%	0.0%	1.4	100.0%	35.9%	51.5	0.0%	2.3	100.0%	36.1%	26.6
		150%	0.0%	2.1	100.0%	38.7%	0.0%	2.4	100.0%	28.6%	0.0%	2.9	100.0%	39.2%	85.4	0.0%	3.1	100.0%	39.3%	47.5
		200%	0.0%	2.1	100.0%	39.1%	0.0%	2.9	100.0%	28.9%	0.0%	3.6	100.0%	39.6%	146.2	0.0%	2.9	100.0%	39.8%	58.4
Avg. ($n = 9$)			0.0%	1.9	100.0%	50.4%	0.0%	3.3	100.0%	37.6%	0.0%	2.4	100.0%	51.4%	113.7	0.0%	3.1	100.0%	51.7%	65.6
19	1	20%	1.4%	1,527.9	60.0%	60.7%	1.9%	1,556.7	60.0%	38.8%	0.0%	618.5	100.0%	62.2%	21.2	0.2%	811.0	90.0%	62.8%	12.3
		40%	18.0%	3,600.0	0.0%	44.0%	20.3%	3,600.0	0.0%	31.7%	12.1%	3,545.3	10.0%	44.4%	241.6	12.5%	3,600.0	0.0%	44.3%	131.5
		60%	26.5%	3,600.0	0.0%	43.7%	30.9%	3,600.0	0.0%	33.6%	22.2%	3,600.0	0.0%	43.8%	618.3	22.7%	3,600.0	0.0%	43.8%	343.7
		100%	23.9%	3,600.0	0.0%	45.1%	32.2%	3,600.0	0.0%	35.0%	23.6%	3,600.0	0.0%	45.1%	1,448.1	23.7%	3,600.0	0.0%	45.2%	837.9
		150%	22.8%	3,600.0	0.0%	45.8%	30.8%	3,600.0	0.0%	35.6%	23.3%	3,600.0	0.0%	45.9%	1,472.5	22.3%	3,600.0	0.0%	45.9%	765.7
		200%	24.3%	3,600.0	0.0%	45.5%	31.6%	3,600.0	0.0%	35.4%	23.8%	3,600.0	0.0%	45.6%	1,505.2	23.4%	3,600.0	0.0%	45.6%	1,013.3
	2	20%	0.2%	1,079.2	90.0%	60.2%	1.0%	1,236.4	80.0%	37.8%	0.0%	740.6	100.0%	61.7%	12.2	0.1%	784.7	90.0%	62.0%	9.9
		40%	9.4%	3,129.4	20.0%	40.5%	11.3%	3,306.3	10.0%	24.6%	5.7%	2,605.3	50.0%	41.5%	121.1	7.1%	2,920.9	30.0%	41.5%	51.9
		60%	22.9%	3,600.0	0.0%	38.8%	27.1%	3,600.0	0.0%	27.9%	19.0%	3,600.0	0.0%	38.9%	248.3	18.1%	3,600.0	0.0%	39.0%	136.3
		100%	24.2%	3,600.0	0.0%	40.8%	36.0%	3,600.0	0.0%	30.4%	22.8%	3,600.0	0.0%	40.9%	504.9	24.9%	3,600.0	0.0%	40.9%	203.5
		150%	23.8%	3,600.0	0.0%	40.9%	38.2%	3,600.0	0.0%	30.5%	24.3%	3,600.0	0.0%	41.0%	705.1	23.3%	3,600.0	0.0%	41.0%	252.0
		200%	24.4%	3,600.0	0.0%	40.6%	35.8%	3,600.0	0.0%	30.3%	24.5%	3,600.0	0.0%	40.7%	659.7	22.0%	3,600.0	0.0%	40.7%	249.4
	3	20%	0.5%	1,107.0	90.0%	58.6%	1.1%	1,274.6	70.0%	37.5%	0.3%	751.6	90.0%	61.1%	10.2	0.2%	801.4	90.0%	61.3%	9.7
		40%	6.6%	3,091.2	30.0%	38.3%	7.6%	3,141.3	30.0%	20.7%	5.2%	2,588.5	50.0%	39.9%	100.2	5.0%	2,934.8	50.0%	40.1%	58.9
		60%	19.6%	3,600.0	0.0%	34.8%	24.3%	3,600.0	0.0%	23.9%	16.0%	3,600.0	0.0%	34.9%	206.4	14.7%	3,600.0	0.0%	35.2%	94.3
		100%	25.3%	3,600.0	0.0%	36.9%	39.1%	3,600.0	0.0%	26.9%	24.2%	3,600.0	0.0%	37.0%	435.9	22.0%	3,600.0	0.0%	37.0%	135.2
		150%	24.0%	3,600.0	0.0%	37.0%	40.6%	3,600.0	0.0%	27.0%	22.9%	3,600.0	0.0%	37.1%	533.8	20.6%	3,600.0	0.0%	37.1%	194.2
		200%	23.9%	3,600.0	0.0%	37.4%	39.7%	3,600.0	0.0%	27.2%	21.0%	3,600.0	0.0%	37.4%	678.7	20.3%	3,600.0	0.0%	37.5%	160.1
Avg. ($n = 19$)			17.9%	3,151.9	16.1%	43.9%	25.0%	3,184.2	13.9%	30.8%	16.2%	3,002.8	22.2%	44.4%	529.1	15.7%	3,058.5	19.4%	44.5%	258.9

formulations *MILP-BC* and *MILP-BC-VI*, we notice that their relative root relaxations are just slightly better (about 1%) than those associated with *MILP-A*.

In Figure 6.5.1, we provide an insight into the savings (w. r. t. the TSP solution) that are calculated as follows:

$$\Delta := 100\% - \frac{\text{objective value of the feasible solution after runtime}}{\text{optimal objective value of the TSP solution}} \tag{6.53}$$

Moreover, Figure 6.5.2 highlights the share of customers served by the drone. These figures are plotted for the results obtained through *MILP-BC-VI*. Regarding the savings shown in Figure 6.5.1, we can make the following observations. As might be expected, the savings generally increase as the drone's velocity and endurance are increased. However, for a given value of α, no significant gains can be obtained once a certain threshold of the endurance parameter R is reached. In any case, savings of about 35% w. r. t. the TSP solution are possible. This magnitude of savings is consistent with previous observations (see, e. g., (Agatz et al., 2018; Schermer et al., 2019b)). In Figure 6.5.2, overall, the number of customers served by the drone is significant and grows as the relative velocity α of the drone is increased. What is interesting about these figures is that the shares of operations do not grow monotonic with α and R. In particular, we can observe that as these values change, round trips tend to be substituted with multi-leg sorties.

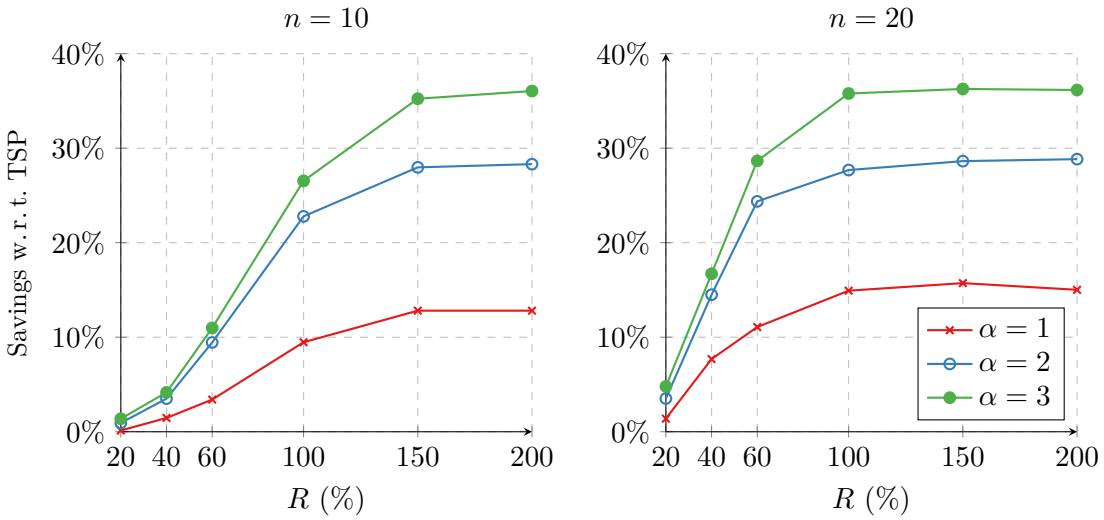

Figure 6.5.1: The savings w. r. t. the TSP solution depending on the drone's relative velocity α and its relative endurance R for the results obtained through *MILP-BC-VI*.

Finally, let us investigate the makespan components according to the objective function (6.34). For this purpose, we use the results obtained through the formulation *MILP-BC-*

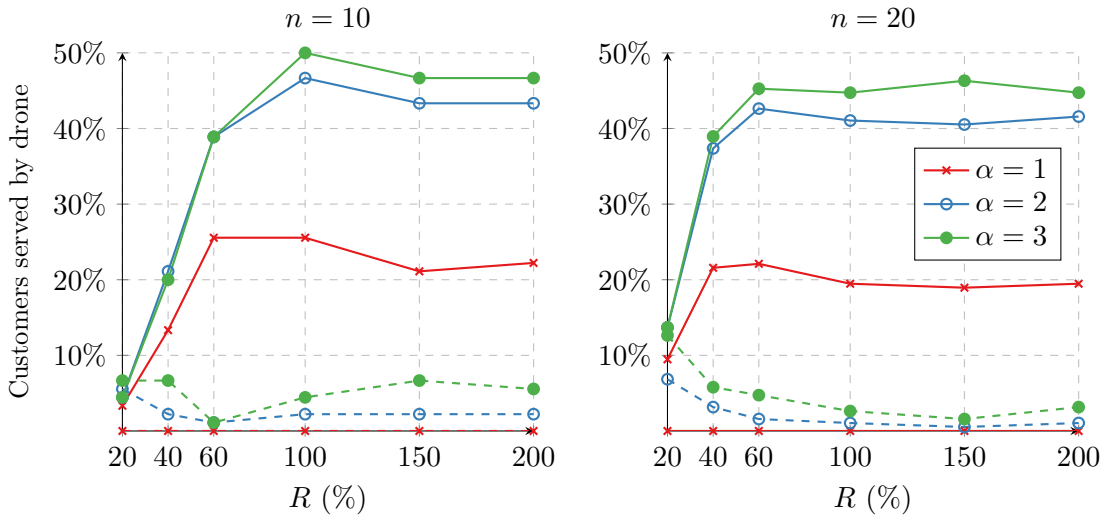

Figure 6.5.2: The share of customers served by the drone depending on its relative velocity α and its relative endurance R for the results obtained through *MILP-BC-VI*. Solid lines represent multi-leg sorties and dashed lines indicate round trips.

VI. Table 6.5.2 provides a breakdown of the makespan (recall that $\bar{t}_l = \bar{t}_r = 0$) and is best viewed in conjunction with Figure 6.5.2.

The time spent traveling on arcs has the largest contribution with more than 97% on average. Moreover, the time spent waiting by the truck for the drone to join from a multi-leg is less than 1.5% on average (similar observations were made in (Es Yurek and Ozmutlu, 2018)). However, there is a strong dependence on the relative velocity α as a faster drone (and a drone with an increased relative endurance R) typically also causes the (non-synchronized) waiting time to increase.

6.5.2 Instances proposed by Poikonen et al.

If we assume that a drone can make an autonomous delivery to a customer, it might need to land at the customer's location. After landing, if we ask that the drone can delay its departure for a sufficient amount of time, only the time in-flight or the distance covered (given that the velocity is constant) are decision-relevant (for a detailed discussion, refer to (Schermer et al., 2019b)). This assumption is present in several papers (see, e. g., (Agatz et al., 2018; Murray and Chu, 2015)) and was investigated in Section 6.5.1. However, in some situations, e. g., for the sake of operational or safety reasons, it might be required that the drone must depart immediately; therefore, the drone only performs a *flyover* at the customer's location. In such a scenario, if the drone arrives at the retrieval location before the truck and is not permitted to land it will need to *hover*. While hovering,

Table 6.5.2: The proportion of the time spent moving, waiting for the drone to arrive from a multi-leg (non-synchronized waiting time), and the time spent waiting while the drone is doing round trips. The presented results were obtained through *MILP-BC-VI* and have been averaged over 10 instances per entry.

Instance		$n = 9$			$n = 19$		
α	R	move	wait	round trip	move	wait	round trip
1	20%	99.7%	0.3%	0.0%	99.6%	0.4%	0.0%
	40%	99.8%	0.2%	0.0%	99.2%	0.8%	0.0%
	60%	97.8%	2.2%	0.0%	98.7%	1.3%	0.0%
	100%	98.5%	1.5%	0.0%	98.5%	1.5%	0.0%
	150%	99.1%	0.9%	0.0%	99.5%	0.5%	0.0%
	200%	98.9%	1.1%	0.0%	99.4%	0.6%	0.0%
2	20%	99.0%	0.0%	1.0%	97.4%	0.1%	2.5%
	40%	98.9%	0.1%	0.9%	96.8%	0.5%	2.7%
	60%	97.0%	1.8%	1.2%	96.4%	2.3%	1.3%
	100%	95.8%	1.8%	2.4%	98.0%	1.6%	0.4%
	150%	96.5%	2.5%	0.9%	99.5%	0.5%	0.0%
	200%	96.6%	2.5%	0.9%	98.0%	1.6%	0.4%
3	20%	99.1%	0.0%	0.9%	96.5%	0.0%	3.5%
	40%	97.9%	0.0%	2.1%	96.3%	0.7%	3.0%
	60%	98.5%	0.7%	0.8%	95.1%	2.0%	2.9%
	100%	96.7%	1.2%	2.1%	97.6%	1.5%	0.9%
	150%	90.6%	4.4%	5.1%	98.1%	1.7%	0.3%
	200%	91.0%	5.1%	3.9%	96.6%	2.3%	1.1%
Average		97.3%	1.4%	1.2%	97.8%	1.1%	1.1%

battery charge is consumed at approximately the same rate as during straight flight (refer to (Dorling et al., 2017)). Such a case is investigated by Poikonen et al. (2019) and in this section, we experiment using the instances provided in (Poikonen et al., 2019). We begin by describing the experimental setup in Section 6.5.1.1. Afterwards, we present the numerical results in Section 6.5.1.2.

6.5.2.1 Experimental Setup

Poikonen et al. (2019) propose a large set of instances where coordinates are distributed uniformly on a 50 by 50 grid. From this set, we test on 25 instances for each size of $n \in \{9, 14, 19\}$ customers. In these instances, as opposed to Section 6.5.1, it is generally assumed that the truck follows the Manhattan metric instead. The drone is still modelled with a relative velocity of $\alpha \in \{1, 2, 3\}$ w. r. t. the truck and the overhead times are furthermore assumed to be negligible, i. e., $\overline{t}_l = \overline{t}_r = 0$. However, in contrast to Section 6.5.1, the drone has a battery life of $\mathcal{E} \in \{10, 20, 30\}$ time units and must be recovered by the truck within a time window of length $\mathcal{E}$ after each launch.

Due to the relatively poor performance of *MILP-B* in Section 6.5.1 and the computational overhead, for the remainder of this paper, we only consider *MILP-A*, *MILP-BC*, and *MILP-BC-VI* for solving the resulting set of 675 problems; hence, a total number of 2025 experiments were performed. In order to match our models to the changed endurance assumption, some adjustments are necessary. For this purpose, in *MILP-A*, we should replace the endurance constraints (6.10) with the following ones:

$$(a_j - s_i) - M(2 - y_{iw} - \tilde{y}_{wj}) \leq \mathcal{E} \qquad \forall i \in V_L, w \in V_N, j \in V_R, i \neq w, i \neq j, w \neq j. \quad (6.54)$$

These constraints state that if a drone performs a sortie (i, w, j) then the total flight and hovering time for that sortie may not exceed $\mathcal{E}$. Preliminary experiments have shown that, to enable a fairer comparison, it is in fact appropriate to keep constraints (6.10) in *MILP-A* as VIEQs, i. e., $\mathcal{E}$ is a valid upper bound on the time in motion (excluding hovering). As constraints (6.10) are not affected by the M-coefficient, this has a positive impact on the effectiveness of Gurobi Optimizer in solving *MILP-A*. In addition, for *MILP-BC*, we propose the following endurance constraints:

$$\sum_{\substack{i \in V_L, \\ i \neq w, \\ i \neq j}} \bar{t}_{iw} y_{iw} + \bar{t}_{wj} \tilde{y}_{wj} + w_j^d \leq \mathcal{E} \qquad \forall w \in V_N, j \in V_R, w \neq j. \quad (6.55)$$

These constraints state that the flyover time and hovering (non-synchronized waiting) time at the retrieval location may not exceed the maximum endurance. Note that, round trips do by their nature not include a drone hovering time. Hence, the endurance constraints for round trips displayed in (6.9) remain unaffected for either formulation.

6.5.2.2 Numerical Results

The detailed numerical results are presented in Tables 6.5.3 and 6.5.4. To be more precise, Table 6.5.3 shows the MIP gap, runtime, the share of optimal solutions, and the relative root relaxation as average values. In addition, for formulations *MILP-BC* and *MILP-BC-VI*, the average number of cuts of type (6.38) and (6.39) is shown. Moreover, Table 6.5.4 provides an insight in the composition of the makespan and the share of customers served by the truck and drone, respectively.

Compared to the observations made in Section 6.5.1, as Table 6.5.3 reveals, the change in the assumptions has a noticeable influence on the solver. In more detail, small instances with $n = 9$ customers are still solved in less than 2 seconds on average by either formulation. For $n = 14$, almost all instances can be solved to proven optimality within one

Table 6.5.3: Average MIP gap, average runtime (in seconds), the share of instances solved to optimality (opt.), and the average relative root relaxation $\%_r$ depending on the number of customers n, the relative velocity of the drone α, and the endurance R on the Poikonen et al. (2019) instance set (25 instances per row). For *MILP-BC* and *MILP-BC-VI*, this table shows also the average number of constraints (6.38) and (6.39) added as cuts.

Instance			*MILP-A*				*MILP-BC*					*MILP-BC-VI*				
n	α	$\mathcal{E}$	gap	time	opt.	$\%_r$	gap	time	opt.	cuts	$\%_r$	gap	time	opt.	cuts	$\%_r$
9	1	10	0.0%	0.1	100%	72.7%	0.0%	0.1	100%	7.0	74.5%	0.0%	0.2	100%	3.3	75.1%
		20	0.0%	0.3	100%	63.3%	0.0%	0.4	100%	20.0	65.4%	0.0%	0.4	100%	16.6	66.1%
		30	0.0%	0.8	100%	53.6%	0.0%	0.8	100%	46.3	54.9%	0.0%	1.0	100%	31.8	55.7%
	2	10	0.0%	0.3	100%	62.4%	0.0%	0.3	100%	20.4	65.1%	0.0%	0.4	100%	12.1	65.7%
		20	0.0%	1.3	100%	43.4%	0.0%	1.6	100%	82.4	43.9%	0.0%	2.2	100%	37.4	44.4%
		30	0.0%	3.8	100%	41.7%	0.0%	3.8	100%	147.7	42.0%	0.0%	4.1	100%	79.2	42.3%
	3	10	0.0%	0.7	100%	49.4%	0.0%	0.9	100%	37.4	51.5%	0.0%	1.1	100%	16.5	52.4%
		20	0.0%	3.1	100%	36.0%	0.0%	4.0	100%	172.0	36.3%	0.0%	4.3	100%	65.6	36.7%
		30	0.0%	3.8	100%	41.3%	0.0%	5.7	100%	213.0	41.9%	0.0%	4.0	100%	94.4	42.3%
Avg. ($n = 9$)			0.0%	1.6	100%	51.5%	0.0%	2.0	100%	82.9	52.8%	0.0%	2.0	100%	39.7	53.4%
14	1	10	0.0%	1.8	100%	73.0%	0.0%	1.3	100%	4.3	74.4%	0.0%	1.4	100%	4.8	74.5%
		20	0.0%	22.6	100%	59.6%	0.0%	10.6	100%	38.7	61.7%	0.0%	12.2	100%	21.8	62.5%
		30	0.6%	445.1	96.0%	48.1%	0.3%	359.7	96.0%	162.0	48.5%	0.4%	343.4	96.0%	65.9	48.8%
	2	10	0.0%	23.5	100%	57.0%	0.0%	15.9	100%	23.4	59.7%	0.0%	16.8	100%	13.0	60.4%
		20	0.0%	404.3	100%	41.8%	0.5%	412.1	96.0%	291.0	41.9%	1.0%	552.5	92.0%	85.3	42.1%
		30	0.5%	890.1	92.0%	45.9%	0.2%	664.1	96.0%	513.7	46.0%	0.0%	370.3	100%	165.4	46.1%
	3	10	0.3%	227.7	96.0%	37.1%	0.0%	136.7	100%	166.3	38.5%	0.6%	211.1	96.0%	45.0	39.4%
		20	0.0%	873.8	100%	39.0%	1.4%	1,510.4	76.0%	714.4	39.1%	0.0%	492.4	100%	167.0	39.4%
		30	0.0%	559.2	100%	43.8%	0.0%	432.7	96.0%	612.6	43.9%	0.0%	137.9	100%	130.8	44.2%
Avg. ($n = 14$)			0.1%	383.1	98.2%	49.5%	0.3%	393.7	95.6%	280.7	50.4%	0.2%	237.5	98.2%	77.7	50.8%
19	1	10	0.0%	28.0	100%	71.9%	0.0%	15.0	100%	13.2	73.2%	0.0%	15.7	100%	10.0	73.9%
		20	2.0%	1,071.3	84.0%	51.6%	1.0%	648.1	88.0%	150.0	52.8%	0.8%	539.9	92.0%	89.1	53.6%
		30	10.1%	3,259.9	20.0%	49.7%	6.4%	2,837.3	36.0%	414.5	49.9%	5.4%	2,716.9	44.0%	184.7	50.0%
	2	10	1.0%	910.0	88.0%	49.4%	0.5%	542.8	96.0%	104.4	51.5%	0.6%	480.1	96.0%	50.2	52.3%
		20	12.7%	3,350.9	16.0%	43.3%	8.8%	3,120.0	36.0%	527.0	43.4%	7.7%	2,991.7	32.0%	168.8	43.5%
		30	14.8%	3,479.7	4.0%	47.0%	12.1%	3,309.4	16.0%	525.1	47.1%	7.6%	2,688.0	52.0%	160.9	47.2%
	3	10	5.6%	2,405.1	52.0%	36.2%	3.6%	1,997.9	68.0%	399.2	36.9%	3.9%	2,078.3	64.0%	133.8	37.1%
		20	19.7%	3,600.1	0.0%	40.2%	15.8%	3,600.0	0.0%	634.4	40.3%	13.1%	2,917.0	32.0%	194.2	40.4%
		30	14.4%	3,407.5	12.0%	43.2%	10.7%	3,365.3	12.0%	917.6	43.3%	6.3%	2,097.6	56.0%	211.2	43.4%
Avg. ($n = 19$)			8.9%	2,390.3	41.8%	48.1%	6.5%	2,159.6	50.2%	409.5	48.7%	5.0%	1,836.1	63.1%	133.6	49.0%

hour. In particular, *MILP-BC-VI* is more effective than *MILP-BC* and solves the same amount of instances (a share of 98.2%) to optimality as *MILP-A* in about half the time (237.5 seconds versus 383.1 seconds) on average. In contrast, *MILP-BC* solves a fraction of instances less to optimality (a share of 95.6%) with an average runtime that is similar to *MILP-A* (393.7 seconds). However, the difference in performance is especially visible for $n = 19$. Here, *MILP-BC-VI* dominates the other approaches and we can solve about 63.1% of all instances to proven optimality within one hour (in 1836.1 seconds with a remaining gap of just 5% on average). This is about 21% (respectively, 13%) more than can be solved by *MILP-A* (respectively, *MILP-BC*) in much shorter time. In general, the relative root relaxations are of a similar order of magnitude for all formulations with slight

Table 6.5.4: The proportion of the time spent moving, waiting for the drone to arrive from a multi-leg, and the time spent waiting while the drone is doing round trips. Moreover, the share of customers served by the truck and drone (through multi-legs and round trips, respectively) is shown. The presented results were obtained through *MILP-BC-VI* and have been averaged over 25 instances per entry.

Instance			makespan compositon			customers served by		
n	α	R	move	wait	round trip	truck	multi-leg	round trip
9	1	10	99.7%	0.2%	0.1%	96.9%	2.2%	0.9%
		20	97.0%	2.2%	0.8%	82.2%	16.0%	1.8%
		30	95.2%	3.4%	1.3%	72.9%	25.8%	1.3%
	2	10	96.5%	0.8%	2.7%	80.9%	8.9%	10.2%
		20	94.4%	2.7%	2.9%	63.6%	29.8%	6.7%
		30	92.1%	4.4%	3.4%	54.7%	39.6%	5.8%
	3	10	92.9%	0.6%	6.5%	68.0%	11.1%	20.9%
		20	85.5%	3.3%	11.2%	49.8%	32.4%	17.8%
		30	84.7%	3.9%	11.4%	44.0%	39.6%	16.4%
Avg. ($n = 9$)			93.1%	2.4%	4.5%	68.1%	22.8%	9.1%
14	1	10	99.4%	0.1%	0.5%	95.7%	3.4%	0.9%
		20	96.0%	2.3%	1.7%	77.1%	20.9%	2.0%
		30	94.1%	4.4%	1.5%	65.7%	33.4%	0.9%
	2	10	92.5%	0.7%	6.8%	76.3%	10.0%	13.7%
		20	92.4%	3.5%	4.0%	58.9%	36.9%	4.3%
		30	94.2%	3.9%	1.9%	52.6%	44.3%	3.1%
	3	10	87.4%	0.9%	11.8%	61.4%	14.3%	24.3%
		20	89.9%	4.4%	5.7%	50.0%	41.1%	8.9%
		30	90.4%	3.1%	6.5%	45.1%	45.1%	9.7%
Avg. ($n = 14$)			92.9%	2.6%	4.5%	64.8%	27.7%	7.5%
19	1	10	98.8%	0.5%	0.7%	93.7%	4.8%	1.5%
		20	95.9%	3.0%	1.1%	73.3%	25.9%	0.8%
		30	94.4%	4.0%	1.6%	65.7%	33.5%	0.8%
	2	10	92.7%	1.5%	5.8%	71.6%	17.9%	10.5%
		20	94.0%	3.8%	2.2%	56.2%	40.0%	3.8%
		30	96.0%	2.4%	1.7%	54.1%	43.2%	2.7%
	3	10	90.1%	1.8%	8.1%	61.5%	23.6%	14.9%
		20	93.1%	3.0%	3.8%	50.9%	41.1%	8.0%
		30	93.8%	3.1%	3.0%	49.1%	45.1%	5.9%
Avg. ($n = 19$)			94.3%	2.6%	3.1%	64.0%	30.5%	5.5%

differences in favor of *MILP-BC-VI*. In particular, the relative velocity α and endurance R have a noticeable influence on the quality of the root relaxation. When we compare the cuts inserted through Algorithm 6.1, we notice that *MILP-BC-VI* generates noticeably fewer cuts than *MILP-BC*.

Noteworthy, for several cases in Table 6.5.3 where the relative velocity $\alpha \in \{2, 3\}$,

instances with $\mathcal{E} = 20$ appear to be more difficult to solve (based on increased runtimes or gaps) than instances with $\mathcal{E} = 30$. Intuitively, one might argue that a reduced endurance should also make it easier to solve the TSPD, as the resulting solution space is more confined. However, a change in endurance often results in structural changes to the nature of optimal solutions. Indeed, when the endurance is large, the number of round trips is more limited, many long-range sorties are performed, and the drone is rarely taken along by the truck. As an example, Figure 6.5.3 shows two sample solutions.

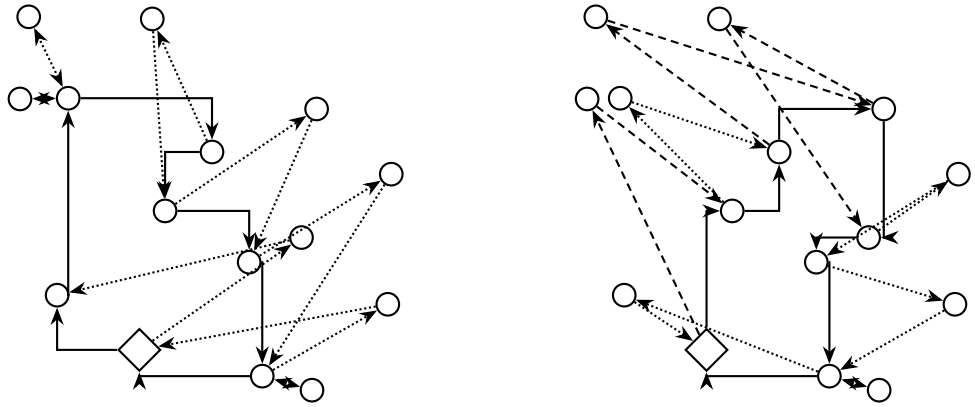

Figure 6.5.3: Optimal solutions for a problem instance with $n = 15$ vertices where the relative velocity is $\alpha = 3$ and the endurance $\mathcal{E}$ is 20 and 30 units in the left-hand and right-hand solutions, respectively. The sorties are indicated by dotted and dashed lines: the former can be performed with an endurance $\mathcal{E} \in [0, 20]$ and latter require an endurance $\mathcal{E} \in]20, 30]$.

Figure 6.5.4 provides insights into the savings Δ (refer to formula (6.53)). At first glance, the strong dependence on both the endurance R and relative velocity α is visible. Overall, for these instances where the truck follows the Manhattan instead of the Euclidean metric, we see that the savings are slightly larger than those observed in Figure 6.5.1.

Finally, Table 6.5.4 reveals that the time in motion has the largest contribution to the makespan. Moreover, waiting times are generally small and increase as R and α increase – that also causes an increase in the share of customers served through multi-legs. Similar observations apply to round trips, even though, their share is generally larger compared to Table 6.5.2.

6.5.3 Instances proposed by Murray and Chu

In this section, we complete our numerical study by testing on the instances proposed by Murray and Chu (2015). We describe the experimental setup in Section 6.5.3.1. Afterwards, the numerical results will be presented in Section 6.5.3.2.

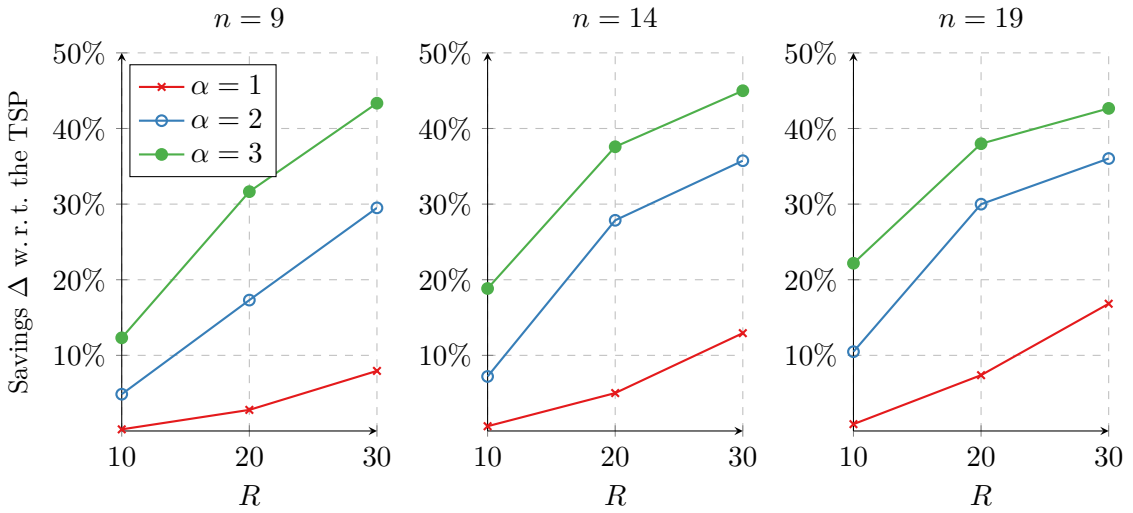

Figure 6.5.4: The average savings Δ w. r. t. the TSP solution depending on the drone's relative velocity α and its endurance $\mathcal{E}$ for the results obtained through *MILP-BC-VI* on the instances proposed by Poikonen et al. (2019).

6.5.3.1 Experimental Setup

Murray and Chu (2015) propose small instances with $n = 10$ customers for the FSTSP and some larger instances with $n = 20$ customers for the *Parallel Drone Scheduling Problem* (refer to (Murray and Chu, 2015)) that we adapt. In these instances, customers are uniformly and independently distributed across an 8-mile square region. Moreover, the depot location corresponds either to the center of gravity, the average of the customer's x-coordinate with y-coordinate of zero (edge), or at the southwest corner of the region (origin). For these instances, it is assumed that the truck moves at 25 miles per hour on the Manhattan metric. When $n = 10$, a drone flying with 15, 25, or 35 miles per hour on the Euclidean metric is considered. In contrast, for $n = 20$, the drone is fixed at 25 miles per hour (Murray and Chu, 2015). Furthermore, the endurance is either 20 or 40 minutes. For these instances, it is also assumed that the overhead times are $\bar{t}_l = \bar{t}_r = 1$ minute.

We follow the assumption of Murray and Chu (2015) and do not permit round trips, i. e., we set $z_{iw} = 0, \forall i \in V_L, w \in V_N, i \neq w$. Additionally, in these instances, only a limited subset $V_D \subset V_N$ of customers (that contains 80% to 90% of all customers) is eligible to be served by the drone. Hence, we must add the following restrictions to the models:

$$y_{iw} = 0 \quad \forall i \in V_L, w \in V_N \setminus V_D, i \neq w, \tag{6.56}$$

$$\tilde{y}_{wj} = 0 \quad \forall w \in V_N \setminus V_D, j \in V_R, w \neq j. \tag{6.57}$$

In total, Murray and Chu (2015) provide 72 instances with $n = 10$ customers and 240 instances with $n = 20$ customers. To limit the number of experiments, we only solve these instances through *MILP-A* and *MILP-BC-VI*, as they have shown the best performance.

6.5.3.2 Numerical Results

Tables 6.5.5 and 6.5.6 present the results for the instances with $n = 10$ and $n = 20$ customers, respectively. In these tables, we present the MIP gaps, runtime, and relative root relaxation (as well as the cuts for *MILP-BC-VI*) as average values. Moreover, we also indicate the components that contribute to the makespan as well as the share of customers served through multi-legs.

In Table 6.5.5, we observe that both formulations can solve instances with $n = 10$ customers to optimality in short computation time. As in the previous sections, here, *MILP-BC-VI* also outperforms *MILP-A* and can solve all small instances in 16 seconds on average. However, on these instances, the time to compute the optimal solution is almost one order of magnitude larger compared to the small instances ($n = 9$) considered in Sections 6.5.1 and 6.5.2.

Table 6.5.5: Average MIP gap, average runtime (in seconds), and the average relative root relaxation $\%_r$ depending on the drone velocity v_d and endurance $\mathcal{E}$ on the Murray and Chu (2015) instance set (12 instances per entry). For *MILP-BC-VI*, the average number of constraints (6.38) and (6.39) added as cuts is shown. Moreover, this table presents the makespan composition and the share of customers served by the drone through multi-legs (based on the results of *MILP-BC-VI*).

Instance		*MILP-A*			*MILP-BC-VI*				makespan			
v_d	$\mathcal{E}$	gap	time	$\%_r$	gap	time	$\%_r$	cuts	move	wait	overhead	multi-legs
15	20	0%	11.3	54.7%	0%	8.2	56.4%	47.1	97.9%	0.0%	2.1%	6.7%
	40	0%	53.9	53.1%	0%	33.8	53.8%	575.6	96.3%	0.0%	3.7%	10.8%
25	20	0%	29.9	53.2%	0%	16.6	53.6%	172.3	91.7%	0.8%	7.5%	20.0%
	40	0%	17.4	54.5%	0%	16.9	54.9%	304.6	89.8%	1.3%	8.9%	23.3%
35	20	0%	12.5	54.8%	0%	10.0	55.3%	197.2	84.2%	3.2%	12.6%	30.8%
	40	0%	16.3	55.1%	0%	10.6	55.6%	241.8	84.5%	2.8%	12.7%	30.8%
Avg. ($n = 10$)		0%	23.6	54.2%	0%	16.0	54.9%	256.4	90.7%	1.4%	7.9%	20.4%

In Table 6.5.6, we present the results for $n = 20$ customers. As the drone speed is fixed to 25 miles per hour on these instances, the table highlights performance w. r. t. the depot location. Generally, we observe that *MILP-BC-VI* also performs better than *MILP-A* on these instances: we observe improved MIP gaps, runtimes, and a larger share of instances solved to optimality. In particular, we observe a strong impact of the depot location and endurance on the difficulty of the problem. Generally, going from $\mathcal{E} = 20$ to $\mathcal{E} = 40$ makes the problem more difficult. Moreover, instances where the depot is located along the center of an edge appear to be the most difficult.

Table 6.5.6: Average MIP gap, average runtime (in seconds), the share of optimal solutions (opt.), and the average relative root relaxation $\%_r$ depending on the depot location and endurance $\mathcal{E}$ on the Murray and Chu (2015) instance set (40 instances per entry). For *MILP-BC-VI*, the average number of constraints (6.38) and (6.39) added as cuts is shown. Moreover, this table presents the makespan composition and the share of customers served by the drone through multi-legs (based on the results of *MILP-BC-VI*).

Instance		*MILP-A*				*MILP-BC-VI*					makespan			
Depot	$\mathcal{E}$	gap	time	opt.	$\%_r$	gap	time	opt.	$\%_r$	cuts	move	wait	overhead	multi-legs
Center	20	12.4%	3,115.3	22.5%	49.9%	6.7%	2,730.6	30.0%	50.6%	256.2	93.5%	1.2%	5.3%	21.0%
	40	18.3%	3,600.0	0.0%	49.9%	13.4%	3,585.1	2.5%	50.2%	705.7	92.1%	1.7%	6.1%	22.8%
Edge	20	18.4%	3,600.0	0.0%	47.2%	12.9%	3,536.6	5.0%	47.6%	347.2	92.1%	1.9%	5.9%	17.1%
	40	17.7%	3,600.0	0.0%	48.9%	14.7%	3,600.0	0.0%	49.3%	674.0	93.5%	0.4%	6.1%	16.9%
Origin	20	7.9%	3,158.8	22.5%	51.8%	3.5%	2,528.2	45.0%	52.4%	204.6	94.0%	1.6%	4.5%	18.0%
	40	17.6%	3,600.0	0.0%	50.8%	12.7%	3,520.2	5.0%	51.1%	1,852.9	93.1%	2.0%	4.9%	18.1%
Avg. ($n = 20$)		15.4%	3,445.7	7.5%	0.5	10.6%	3,250.1	14.6%	50.2%	673.4	93.1%	1.5%	5.5%	19.0%

Finally, it is worth highlighting that the share of customers served by the drone (through multi-legs) is generally smaller in both tables (i. e., Tables 6.5.5 and 6.5.6) than what we observed in previous sections. The reason might be related to the presence of overhead times. Even though their value is relatively small, summed up over all operations, they can account for up to 12.7% of the total makespan (refer to Table 6.5.5).

6.6 Conclusion

In this paper, we investigated the TSPD. More precisely, in Section 6.2, we reviewed existing MILP formulations and exact methods that are used for solving variants of the problem. Given the current state of research, only small-sized instances can be solved to proven optimality – even for the special case of serving every customer exactly once that is the most common one studied in the literature. Motivated by this fact, we proposed two new MILP formulations in Section 6.3. As an alternative, attempting to leverage the respective strengths of existing MILP and ILP approaches in the literature and motivated by our VIEQs, in Section 6.4, we proposed a B&C algorithm for solving the TSPD. Afterwards, in Section 6.5, we conducted an extensive computational study in order to analyze the quality of the different formulations in finding optimal TSPD solutions on various benchmark instances from the literature. As our detailed numerical study reveals, the performance of our formulations in solving the different benchmark instances varies. However, we have confirmed that the B&C approach *MILP-BC-VI* showed strong performance throughout the different instance sets. Indeed, using this approach, we can solve small instances with 9 or 10 customers in just a few seconds and several larger instances with 19 or 20 customers within an hour. Thus, this B&C algorithm challenges the existing

state-of-the-art in solving the TSPD.

From a modeling perspective, naturally, we believe that our formulation might also be used to study variants that consider, e. g., multiple trucks or drones (refer to (Poikonen et al., 2017; X. Wang et al., 2017)). Moreover, the additional variables in Section 6.4, that model the waiting time of the truck and drone, could be helpful in cases where these times might be constrained or associated with some cost (as it is the case in, e. g., (Ha et al., 2018)). We also believe that this formulation might be adjusted in a way to achieve the possibility of allowing each vertex to be visited more than once by the truck (refer to (Agatz et al., 2018)).

From a computational perspective, using formulation *MILP-BC-VI* and a customized B&C scheme lead to the best performance overall. However, we also showed that performance varied over the different benchmark sets. Therefore, it might be worth to further investigate how the different assumptions affect the difficulty of the problem. Finally, in order to obtain a further boost, it might be worthwhile to investigate if the cutting scheme proposed in this work can be enhanced by merging it with other type of cuts proposed in, e. g., (Dell'Amico et al., 2019; Dell'Amico et al., 2021b). For example, one might integrate a separation procedure into the B&C framework in order to cut off infeasible subtours as required instead of using explicit MTZ constraints.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments and suggestions that helped us in improving the quality of this paper.

6.7 References

Agatz, Niels, Paul Bouman, and Marie Schmidt (2018). "Optimization Approaches for the Traveling Salesman Problem with Drone". In: *Transportation Science* 52.4, pp. 965–981.

Aurambout, Jean-Philippe, Konstantinos Gkoumas, and Biagio Ciuffo (2019). "Last mile delivery by drones: an estimation of viable market potential and access to citizens across European cities". In: *European Transport Research Review* 11, pp. 1–21.

Bouman, Paul, Niels Agatz, and Marie Schmidt (2018a). "Dynamic programming approaches for the traveling salesman problem with drone". In: *Networks* 72.4, pp. 528–542.

Bouman, Paul, Niels Agatz, and Marie Schmidt (2018b). *Instances For The Tsp With Drone (And Some Solutions).*

Dell'Amico, Mauro, Roberto Montemanni, and Stefano Novellani (2019). "Models and algorithms for the Flying Sidekick Traveling Salesman Problem". In: *arXiv:1910.02559 [math]*.

Dell'Amico, Mauro, Roberto Montemanni, and Stefano Novellani (2021b). "Drone-assisted deliveries: new formulations for the flying sidekick traveling salesman problem". In: *Optimization Letters* 15.5, pp. 1617–1648.

Dorling, Kevin et al. (2017). "Vehicle Routing Problems for Drone Delivery". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 47.1, pp. 70–85.

Drexl, Michael (2012b). "Synchronization in Vehicle Routing—A Survey of VRPs with Multiple Synchronization Constraints". In: *Transportation Science* 46.3, pp. 297–316.

Es Yurek, Emine and H. Cenk Ozmutlu (2018). "A decomposition-based iterative optimization algorithm for traveling salesman problem with drone". In: *Transportation Research Part C: Emerging Technologies* 91, pp. 249–262.

Fischetti, Matteo, Ivana Ljubić, and Markus Sinnl (2016). "Benders decomposition without separability: A computational study for capacitated facility location problems". In: *European Journal of Operational Research* 253.3, pp. 557–569.

Fischetti, Matteo, Ivana Ljubić, and Markus Sinnl (2017). "Redesigning Benders Decomposition for Large-Scale Facility Location". In: *Management Science* 63.7, pp. 2146–2162.

Freitas, Júlia Cária de and Puca Huachi Vaz Penna (2018). "A Randomized Variable Neighborhood Descent Heuristic to Solve the Flying Sidekick Traveling Salesman Problem". In: *Electronic Notes in Discrete Mathematics* 66, pp. 95–102.

Freitas, Júlia Cária de and Puca Huachi Vaz Penna (2020). "A variable neighborhood search for flying sidekick traveling salesman problem". In: *International Transactions in Operational Research* 27.1, pp. 267–290.

Gurobi Optimization, LLC (2023). *Gurobi Optimizer Reference Manual*.

Ha, Quang Minh et al. (2018). "On the min-cost Traveling Salesman Problem with Drone". In: *Transportation Research Part C: Emerging Technologies* 86, pp. 597–621.

Ha, Quang Minh et al. (2020). "A hybrid genetic algorithm for the traveling salesman problem with drone". In: *Journal of Heuristics* 26, pp. 219–247.

Kim, Sungwoo and Ilkyeong Moon (2018). "Traveling Salesman Problem With a Drone Station". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 49.1, pp. 42–52.

Marinelli, Mario et al. (2018). "En route truck–drone parcel delivery for optimal vehicle routing strategies". In: *IET Intelligent Transport Systems* 12.4, pp. 253–261.

Miller, C. E., A. W. Tucker, and R. A. Zemlin (1960). "Integer Programming Formulation of Traveling Salesman Problems". In: *Journal of the ACM* 7.4, pp. 326–329.

Morandi, Nicola et al. (2023). "The Traveling Salesman Problem with Drones: The Benefits of Retraversing the Arcs". In: *Transportation Science* 57.5, pp. 1340–1358.

Murray, Chase C. and Amanda G. Chu (2015). "The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery". In: *Transportation Research Part C: Emerging Technologies* 54, pp. 86–109.

Otto, Alena et al. (2018). "Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey". In: *Networks* 72.4, pp. 411–458.

Poikonen, Stefan, Bruce Golden, and Edward A. Wasil (2019). "A Branch-and-Bound Approach to the Traveling Salesman Problem with a Drone". In: *INFORMS Journal on Computing* 31.2, pp. 335–346.

Poikonen, Stefan, Xingyin Wang, and Bruce Golden (2017). "The vehicle routing problem with drones: Extended models and connections". In: *Networks* 70.1, pp. 34–43.

Ponza, Andrea (2016). "Optimization of drone-assisted parcel delivery". MA thesis. Università Degli Studi di Padova.

Rojas Viloria, Daniela et al. (2020). "Unmanned aerial vehicles/drones in vehicle routing problems: a literature review". In: *International Transactions in Operational Research* 28.4, pp. 1626–1657.

Sacramento, David, David Pisinger, and Stefan Ropke (2019). "An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones". In: *Transportation Research Part C: Emerging Technologies* 102, pp. 289–315.

Schermer, Daniel, Mahdi Moeini, and Oliver Wendt (2019a). "A hybrid VNS/Tabu search algorithm for solving the vehicle routing problem with drones and en route operations". In: *Computers & Operations Research* 109, pp. 134–158.

Schermer, Daniel, Mahdi Moeini, and Oliver Wendt (2019b). "A matheuristic for the vehicle routing problem with drones and its variants". In: *Transportation Research Part C: Emerging Technologies* 106, pp. 166–204.

Schermer, Daniel, Mahdi Moeini, and Oliver Wendt (2020b). "The Drone-Assisted Traveling Salesman Problem with Robot Stations". In: *Proceedings of the Hawaii International Conference on System Sciences*, pp. 1308–1317.

Wang, Xingyin, Stefan Poikonen, and Bruce Golden (2017). "The vehicle routing problem with drones: several worst-case results". In: *Optimization Letters* 11.4, pp. 679–697.

7 Leveraging Machine Learning for the Per-Instance Configuration of MIP Solvers: An Algorithm Selection Approach

Abstract

Mixed-Integer Programming (MIP) solvers have undergone remarkable advancements, consistently integrating and refining algorithmic techniques developed over several decades of dedicated research. Given the complexity and feature-richness of these solvers, in principle, most of them allow the execution of the inherent *Branch-and-Cut* (B&C) framework to be configured in various ways that influence the optimization. Despite the potential for performance gains, it is non-trivial to make such adjustments sensibly a priori. Consequently, it is common practice to rely on a default configuration, which is generally assumed to be expertly-tuned, enabling it to work well for a wide range of applications.

In this paper, we challenge the conventional approach of relying on the default configuration. Instead, we propose a novel perspective through the *Algorithm Selection Problem* (ASP). The data-driven methodology leverages *Random Forest* (RF) regression as a *Machine Learning* (ML) model to approximate solver performance. This ML model is used to derive the components of a prescriptive optimization problem, seeking the optimal configuration for a given instance. It is demonstrated that this problem can be efficiently solved by a tailored *Branch-and-Bound* (B&B) procedure that exploits the structure of the underlying RF ML model. All things considered, this methodology empowers us to select the (predictably) optimal configuration on a per-instance basis with minimal computational overhead. We evaluate our approach by using a state-of-the-art solver and the well-established MIPLIB. The extensive numerical results demonstrate that the performance gap between the default and virtual-best solver can be closed by about half with respect to the average runtime and MIP gap.

7.1 Introduction

MIP is a fundamental and widely used tool in *Operations Research* (OR) (see, e. g., (Petropoulos et al., 2023) and references therein). Both academic and commercial MIP solvers have made remarkable advancements, consistently integrating and refining algorithmic techniques developed over several decades of dedicated research (Koch et al., 2022). Given the complexity and feature-richness of these solvers, in principle, most of them al-

low the execution of the inherent B&C framework to be configured in various ways that influence the optimization. Despite the potential for performance gains, it is non-trivial to make such adjustments sensibly a priori. Consequently, it is common practice to rely on a default configuration, which is generally assumed to be expertly-tuned, enabling it to work well for a wide range of applications.

In this paper, we challenge this assumption by exploring the problem of configuring a MIP solver on a per-instance basis, adopting the perspective of the ASP (Rice, 1976). Given a set of algorithms and a set of instances, the ASP consists of finding a mapping from instances to algorithms. This must be done in such a way that the performance of the selected algorithm is optimal according to a given performance criterion. The established approach to the ASP involves learning a regressor for each algorithm (Kerschke et al., 2019). However, the scalability of such an approach is clearly limited, and therefore it is only feasible for a handful of algorithms, as commonly studied in the context of the ASP.

To overcome this issue, we use a data-driven methodology that is based on *Empirical Model Learning* (EML) (Lombardi et al., 2017). We consider several thousand distinct configurations of a MIP solver as the set of algorithms and begin by learning an approximation of performance through a function dependent on universal MIP instance features and the algorithmic configuration of the solver. To describe this relationship, we employ RF regression as a ML model. This model is used to derive the components of a prescriptive optimization problem, seeking the optimal configuration of the solver for a given instance. All things considered, this approach empowers us to determine the (predictably) optimal configuration on a per-instance basis with minimal computational overhead. We evaluate our methodology by using a state-of-the-art solver and the well-known MIPLIB (Gleixner et al., 2021), which attempts to reflect the richness of challenges in various OR domains. In our paper, we make the following contributions:

1. We demonstrate the utility of RFs as a ML model to systematically derive the components of a prescriptive optimization problem that cannot intuitively be established in closed form. This problem enables us to determine the (predictably) optimal configuration of a solver on a per-instance basis.

2. For efficiently solving the prescriptive optimization problem, a tailored B&B approach is proposed that exploits the structure of the underlying RF ML model.

3. As a proof-of-concept, an extensive numerical study is conducted, demonstrating the effectiveness of the methodology on a benchmark of heterogenous problem instances. The approach successfully closes the performance gap between the default and virtual-best solver by about half with respect to the average runtime and MIP

gap.

4. Finally, we believe that this paper is a strong testament to the continued coalescence of predictive and prescriptive analytics on a methodological level.

The remainder of this paper unfolds as follows. Section 7.2 presents a brief overview of the related literature. As the problem of finding the optimal configuration for a MIP solver on a per-instance basis is addressed from the perspective of the ASP, Section 7.3 provides a concise definition of that problem. Section 7.4 outlines the proposed methodology in detail. This concerns the use of RFs as a ML model and the subsequent extraction of the components that enable us to formulate the prescriptive optimization problem. Emphasis is placed on how this optimization problem can be efficiently solved by a tailored B&B approach. The computational experiments and their numerical results follow in Section 7.5, where additional insights into the dataset and results obtained through the proposed methodology are provided. Lastly, Section 7.6 concludes our work with some final remarks.

7.2 Related Literature

In recent years, there has been growing interest in the intersection of MIP and ML (Bengio et al., 2021). This section provides an overview of existing methodological approaches that focus on improving the performance of MIP solvers by adjusting their default algorithmic behavior. We begin in Sections 7.2.1 and 7.2.2 with an overview of the *Algorithm Configuration Problem* (ACP) and ASP, respectively. Despite their seemingly close relation, these problems are generally treated differently from a methodological standpoint (Kerschke et al., 2019). Given the extensive literature in those fields and the existence of excellent literature reviews (see, e. g., (Hoos, 2011; Kerschke and Trautmann, 2019; Kotthoff, 2016; Schede et al., 2022)), our aim is to provide only a general introduction that emphasizes key concepts. We restrict ourselves to the application of MIP; although the underlying methods have also found success in related domains (see, e. g., (Kotthoff et al., 2015; Xu et al., 2008)). Both the ACP and ASP involve static adjustments, modifying the configuration of a solver or selecting from a set of solvers. In contrast, in Section 7.2.3 recent approaches to changing the behavior of MIP solvers dynamically, at the iteration level, are highlighted.

7.2.1 Algorithm Configuration in the context of MIP

The ACP can be briefly stated as follows: Given a configurable algorithm $A(\theta)$ (with possible parameter settings $\Theta = \Theta_1 \times \ldots \times \Theta_n$), a set of problem instances Π, and a

performance metric m, find a feasible configuration $\theta \in \Theta$ such that $A(\theta)$ performs optimally on Π according to m (Hoos, 2011; Kerschke and Trautmann, 2019). The underlying assumption is that the instances in Π are relatively homogeneous, falling within a narrow distribution of a specific problem class.

The ACP has some roots in the global optimization of black-box functions (see, e. g., (Jones, 2001; Jones et al., 1998; Mockus et al., 1978)). However, there are crucial differences, which make methods from that domain not directly applicable (Hutter et al., 2011): In black-box optimization, the focus is generally on single, deterministic, and noise-free functions that are queryable (i. e., evaluable at negligible cost). In contrast, in the ACP a query involves executing a typically randomized or stochastic algorithm with a configuration, which is generally time-intensive, especially in MIP scenarios. Birattari (2009) highlights the parallels between the ACP and challenges in ML. Particularly, the ACP is concerned with a large set of instances and even though these instances may be homogeneous, performance can still vary considerably across instances, making a statistical blocking design necessary (Hinkelmann and Kempthorne, 1994). The very fact that performance is to be optimized on a large set of instances is also what differentiates the ACP from hyperparameter optimization in ML (Kerschke et al., 2019).

The ACP is generally treated heuristically, and effective methods combine three distinct steps. These methods begin by sampling configurations, testing them on a subset of instances to assess their performance, and apply local search to identify neighboring configurations (Hutter et al., 2009). This iterative process continues until a given computational budget is exhausted. Afterwards the performance of the final configuration is evaluated against the (generally known) default configuration on a set of test instances (López-Ibáñez et al., 2016). Approaches to the ACP vary in how they identify new configurations and attempt to discard underwhelming ones (e. g., by racing (Maron and Moore, 1997)).

In essence, these approaches can be classified into model-free and model-based methods. Model-free methods (see, e. g., (Adenso-Díaz and Laguna, 2006; Ansótegui et al., 2009; Birattari, 2009; Birattari et al., 2010; Hutter et al., 2007; Hutter et al., 2009; López-Ibáñez et al., 2016)) iterate between generating configurations and subsequently altering them. Such alterations include intensification mechanisms and the exploration of neighborhoods. Model-based approaches (see, e. g., (Ansótegui et al., 2015; Bartz-Beielstein et al., 2021; Hutter et al., 2011)) are computationally more expensive, as they use a statistical model to build an approximation of algorithmic performance (Leyton-Brown et al., 2002). However, such approaches allow for the explicit consideration of problem features, facilitating the systematic identification of new configurations that are expected to perform well, in contrast to a purely randomized exploration (Hutter et al., 2011).

As discussed in Section 7.1, in the realm of MIP, a general assumption might be that the default configuration of any MIP solver is already an optimum of the ACP for a large-scale, heterogenous class of problems. However, it is not surprising that the ACP has been intensively studied in this context. Such studies have shown that this can lead to superior configurations of MIP solvers on homogeneous problem instances: e. g., Hutter et al. (2010), Hutter et al. (2011), Hutter et al. (2007), and Hutter et al. (2009) demonstrate that speedups of up to two orders of magnitude are possible. Although, it must be stressed that these speedups are highly solver and benchmark dependent. Moreover, a known caveat is that such problem-specific configurations do not generalize well, i. e., they can perform significantly worse than the default configuration when used outside their special scenarios (Hutter et al., 2011). It is for this reason that we do not pursue the ACP any further in the context of this work. As discussed during Section 7.1, instead of a one-size-fits-all approach, we aim to determine a promising configuration on a per-instance basis.

7.2.2 Algorithm Selection in the context of MIP

Based on the ideas of Rice (1976), the ASP can be briefly stated as follows: Given a set of algorithms $\mathcal{A}$, a set of problem instances Π, and a performance metric m, determine for each problem $\pi \in \Pi$ which algorithm $A \in \mathcal{A}$ performs optimally according to m (for more details refer to Section 7.3). In contrast to the ACP (see Section 7.2.1), the instances Π may be assumed to be heterogenous. Therefore, it may not be appropriate to use a single algorithm, but instead worth attempting to leverage performance complementarities.

The ASP is generally approached in two phases (Kotthoff, 2016). First, a ML model is learnt with sufficient historical data. Then, in the second phase, this model is used for algorithm selection. While it may seem compelling to learn a classifier, directly predicting which algorithm to use for a given instance, the predominant approach involves the use of regressors (see (Kerschke et al., 2019) and references therein). This may be explained by the fact that a pure classification approach lacks information concerning the magnitude of the performance difference between algorithms. In scenarios involving only two algorithms, a single regressor can be used to compute (predict) the relative performance between them (see, e. g., (Berthold et al., 2022)). And as long as the number of algorithms is small (typically less than a dozen), the conventional approach is to learn a regressor for each algorithm (see, e. g., (Kerschke et al., 2018; Leyton-Brown et al., 2003)). Still, a major drawback is that such approaches are not suited to scale well in presence of numerous algorithms, as both offline training and, particularly, online use may become too time-consuming or resource-intensive.

A key requirement of the ASP is that the set of algorithms $\mathcal{A}$ must be fixed a priory. Typically, an educated guess or preliminary experiments are made to determine which parameters are most likely to influence the performance of a single, configurable algorithm (see, e. g., (Hutter et al., 2006)). Alternatively, a portfolio of different algorithms can be considered, each using their default configuration or, if computational resources allow, finding an alternative configuration for each of them by means of the ACP (see, e. g., (Kerschke et al., 2018)). Another approach to integrate the ACP in the ASP involves clustering (Kadioglu et al., 2010). Assume that a single, configurable algorithm A is given and any unsupervised ML algorithm is used to split the instances Π into k clusters $\Pi_1, \ldots, \Pi_k$. Then, any approach to the ACP can be used to determine an optimal configuration $A(k)$ for each cluster k. For an unseen instance, an attempt is made to associate it with the closest cluster and select the corresponding algorithm. Kadioglu et al. (2010) show that this approach works on average better than an instance-oblivious ACP approach. Nevertheless, a pure clustering approach may be agnostic to instance hardness, as structurally similar problems may still vary drastically in difficulty. Xu et al. (2011) address this concern by training a set of RFs to classify, for each pair of algorithms, which algorithm performs better. Following that, a majority rule can be applied by using all RFs to determine the winner. They show that this approach yields superior results compared to relying on a pure ACP approach.

The ASP has been scarcely investigated in the context of MIP solvers. Two notable works focus on the binary selection of a single algorithmic component by means of the ASP. These works are of particular interest, because they are said to have already been integrated in the commercial MIP solver FICO Xpress (Fair Isaac Corporation, 2023). One of these works explores the decision of choosing between standard scaling and Curtis-Reid scaling (Berthold and Hendel, 2021). Generally, the latter is more appropriate for numerically challenging problems; however, always using it would be computationally detrimental (for details refer to (Berthold and Hendel, 2021) and references therein). The authors propose to use the coefficient ranges, that result from applying either scaling technique, as features and the attention level, which is a proxy for the likelihood of numerical problems, as a label. A linear model and RF are trained as regressors and, by defining appropriate threshold values, these models are used classify which scaling method to apply for a given instance. Their results show that even the simple linear model can improve the numerical stability through autoscaling. In another work, Berthold et al. (2022) address whether cuts should be generated globally (i.e., at the root node of the B&C tree) or also locally. Beyond static features (e. g., size- and sparsity), they use dynamic features that capture the results of applying a scaling and presolving routine, as well as the effect of global cuts at the root node. They generate a dataset that captures the effects on instance-based

runtime depending on whether local cuts are used or not, and train a RF regressor that learns to predict the speedup factor when using local cuts. When comparing the two base strategies, i.e., always or never generating local cuts, the former dominates. However, by using their RF to decide which strategy to use, an improvement w. r. t. runtime is achieved.

In the context of the ACP there has been some evidence that the configuration landscape of an algorithm may be more benign than expected (see (Pushak and Hoos, 2018)). Although, treating the selection of individual components of an algorithm separately might fail to capture the interdependence of various algorithmic components, especially when dealing with heterogenous instances. In that regard, Iommazzo (2021), Iommazzo et al. (2020a), and Iommazzo et al. (2020b) study the ASP in the context of a unit commitment problem. They consider the ASP to be the problem of configuring multiple algorithmic components of a MIP solver (thus forming distinct algorithms) on a per-instance basis. A dataset is constructed by sampling each algorithm on each instance and features specific to unit commitment problems are used. *Support Vector Regression* (SVR) with linear or radial-basis function kernels or a *Decision Tree* (DT) are employed to build an approximation. This approximation relies not only on problem features, but also on the algorithm's parameters (see also (Hutter et al., 2011)). The authors demonstrate how an optimal algorithm can be extracted in the presence of an instance using an optimization model. With this methodology (see also (Lombardi et al., 2017)), they can, on average, improve over the default solver w. r. t. the MIP gap within a runtime limit; although, this comes at the cost of a notably smaller share of feasible solutions after the runtime limit. One general issue that the authors raise concerns scalability. On the one hand, the optimization problems resulting from the DT and linear SVR can be solved relatively quickly. On the other hand, SVR in conjunction with a radial-basis function implies a difficult *Mixed-Integer Non-Linear Programming* (MINLP) problem, taking several seconds to be solved by a state-of-the-art MINLP solver. As discussed in more detail in Section 7.3, a near-instantaneous selection of an algorithm is crucial requirement for any effective ASP approach.

7.2.3 Dynamic Configuration of MIP solvers

Statically adjusting (or selecting) an algorithm, as discussed in Sections 7.2.1 and 7.2.2, can be seen as a high-level decision. However, there have also been efforts to enhance MIP solvers at the iteration level. As such works are only loosely related to this paper, we discuss them with utmost brevity.

Dynamic decisions in MIP solving by B&C concerns, e. g., issues such as node selection, branching decisions, or when to apply primal heuristics (see, e. g., (Lodi and Zarpellon,

2017) and references therein). In practice, these decisions are generally made heuristically (Lodi, 2013) and such problems have the typical characteristics of a Reinforcement Learning (compare to, e. g., (Mitchell, 1997; Russell et al., 2010; Sutton and Barto, 2018)) or Multi-Armed Bandit Problem (see, e. g., (Auer et al., 2002; Gittins, 1979)): i. e., given a state (of the B&B tree), possible actions (e. g., on which variable to branch in a given node), and reward signals (e. g., the change in the integrality gap), the objective is to find an optimal policy.

Nevertheless, learning to branch is commonly approached from the perspective of supervised ML, e. g., by creating a ranking of possible branching variables (Gupta et al., 2020; Khalil et al., 2016; Zarpellon et al., 2021) in each B&B node. Similar methodologies have been applied to dynamically run components of a solver, e. g., primal heuristics (Chmiela et al., 2021; Khalil et al., 2017). These methods have demonstrated promising results in terms of noticeable speedups.

7.3 Preliminaries & Problem Definition

In this study, we want to approach the problem of selecting a configuration for a MIP solver on a per-instance basis, adopting the perspective of the ASP. Akin to Rice (1976) (and, e. g., Hoos et al. (2021), Kerschke et al. (2019), and Kotthoff (2016)), we define the ASP as follows. Given a set of instances Π that belong to a problem space $\mathcal{P}$, we have a set of algorithms $\mathcal{A}$ that are applicable to $\mathcal{P}$. The performance of $A \in \mathcal{A}$ on $\pi \in \Pi$ is specified by a mapping $p : \Pi \times \mathcal{A} \to \mathbb{R}^n$; and the n components represent different performance measures that are associated with executing A on π (e. g., runtime, memory consumption, etc.). As the trade-off between these components can be intricate and domain-specific, and may depend on user preferences, a norm $\|p\| : \mathbb{R}^n \to \mathbb{R}_0^+$ is commonly employed. Without loss of generality, we focus on minimizing $\|p\|$. In its most desirable form, the ASP involves constructing a selector (recommender) $S : \Pi \to \mathcal{A}$ such that the following condition holds:

$$\|p(\pi, S(\pi))\| \leq \|p(\pi, A)\| \quad \forall \pi \in \Pi, A \in \mathcal{A}. \tag{7.1}$$

Any selector S^* that satisfies (7.1) is commonly referred to as an *oracle* or *Virtual-Best Solver* (VBS), and it establishes a tight lower bound on the achievable performance for any viable selector (Kerschke et al., 2019; Rice, 1976). Closely related to the VBS is the concept of the *Single-Best Solver* (SBS), i. e., any single algorithm A^* with the best performance among all the $A \in \mathcal{A}$; consequently, A^* must satisfy the following condition

(Kerschke et al., 2019):

$$\sum_{\pi \in \Pi} \|p(\pi, A^*)\| \leq \sum_{\pi \in \Pi} \|p(\pi, A)\| \quad \forall A \in \mathcal{A}. \tag{7.2}$$

The SBS A^* qualifies as an upper bound that must be maintained by any competitive recommender S. An alternative is to derive an upper bound from the performance of a known *Default Solver* (DS) $A_0 \in \mathcal{A}$ (see, e. g., (Berthold et al., 2022; Iommazzo et al., 2020a)).

A seemingly trivial approach to a VBS would be to define $S(\pi) = \arg\min\{p(\pi, A) \mid A \in \mathcal{A}\}$, which may be achieved by exhaustively evaluating every algorithm in presence of an instance. Such a recommender would guarantee that condition (7.1) always holds and is therefore an oracle. However, this procedure is of limited practical use and illustrates that several further (soft) constraints may apply to the ASP. A universal constraint is that the (online) evaluation of $S(\pi)$ must be near-instantaneous, with negligible resource requirements (compared to executing the selected A), rendering the trivial approach impractical. In contrast, the (one-time offline) construction of $S(\pi)$ is generally allowed to be resource-intensive. This highlights the need for efficient and scalable methodologies for algorithm selection in practice.

As recognized early by Rice (1976), a key component in constructing a competitive selector is the use of features (see also (Kotthoff, 2016)). We assume that the feature space $\mathcal{F}$ has lower dimensionality than $\mathcal{P}$, and we define a feature mapping by $f : \mathcal{P} \to \mathcal{F}$. This allows us to reshape the selector as $S : \mathcal{F} \to \mathcal{A}$ and apply a function composition $S(f(\pi))$ instead. For features to be useful, they must be computationally inexpensive and sufficiently discriminative; and some argue that the design of high-quality features, custom-tailored to match $\mathcal{P}$, is a primary activity in the ASP (see (Kerschke et al., 2019) and references therein).

To summarize, given a set of problems Π and a set of applicable algorithms $\mathcal{A}$, the ASP can be stated as searching for a feature mapping $f : \mathcal{P} \to \mathcal{F}$ and a selector $S : \mathcal{F} \to \mathcal{A}$, where $S(f(\pi))$ can be evaluated near-instantaneously, that solves the following optimization problem:

$$\min \sum_{\pi \in \Pi} \|p(\pi, S(f(\pi)))\| \tag{7.3}$$

$$\text{s. t.} \quad \sum_{\pi \in \Pi} \|p(\pi, S^*(\pi))\| \leq \sum_{\pi \in \Pi} \|p(\pi, S(f(\pi)))\| \leq \sum_{\pi \in \Pi} \|p(\pi, A^*)\| \leq \sum_{\pi \in \Pi} \|p(\pi, A_0)\| \tag{7.4}$$

Proving that a selector is optimal for (7.3)–(7.4) is non-trivial. As a result, algorithm selection is generally treated as an empirical two-phase process (Kerschke et al., 2019;

Kotthoff, 2016). In the first phase, historical data is given or generated, features are engineered, and a selector – generally, a ML model – is learnt offline using training data. The selector is then used online, on a per-instance basis, to determine which algorithm to use on test data. To evaluate the performance, a comparison is drawn to the VBS and DS (or SBS) by means of ex-post control, with the primary objective of closing as much of the performance gap between the DS (or SBS) and VBS as possible. Kerschke et al. (2019) note that state-of-the-art algorithm selection procedures have been able to close such gaps by 25% to 96%. However, they emphasize the need to recognize that the performance gap between the SBS and VBS can be substantial, especially when numerous algorithms display strong performance complementarities on different instances.

7.4 Methodological Approach

As discussed in Sections 7.2–7.3, the key to the ASP lies in the construction of a selector S that utilizes instance-based features $f(\pi)$. Evaluating $S(f(\pi))$ can be computationally challenging, and in some cases, heuristic solutions might be necessary. Typically, the number of algorithms is small in the ASP, and the standard approach is to learn a regressor for each algorithm. An alternative could be to rely on pairwise classification and a majority vote. However, such approaches face scalability challenges when dealing with more than a few algorithms, as both offline training and, particularly, online use may become too time-consuming or resource-intensive.

The methodology proposed in this paper adopts the paradigm of EML (Lombardi et al., 2017), a process leveraging ML to derive components of a prescriptive optimization problem that cannot intuitively be established in closed form. In EML, the first step involves learning the relation between decisions and observations through a ML model. Afterwards, components of the learnt ML model are embedded into an optimization problem. In this paper, we design this problem in a way to search for an optimal algorithm $A \in \mathcal{A}$ in presence of an instance $\pi \in \Pi$. This work employs RFs as a ML model (see Section 7.4.1) and demonstrates how its components can be embedded within a combinatorial optimization problem (see Section 7.4.2). The proposed approach, distinct from related works that leverage tree-based learners (see, e. g., (Hutter et al., 2011; Hutter et al., 2006; Iommazzo, 2021)), introduces a novel and efficient way for solving the resulting optimization problem by using a tailored B&B procedure that exploits the structure of the underlying RF.

The assumption about the algorithm space $\mathcal{A} = \Theta_1 \times \ldots \times \Theta_m$ is made with the only requirement that each algorithm $A \in \mathcal{A}$ is characterized by a finite sequence of parameters, each of which has a continuous or integer domain. This assumption is common when dealing with configurable algorithms, where distinct algorithmic configurations (i. e., specific

instantiations) may be viewed as separate algorithms (Hutter et al., 2011; Kerschke et al., 2019; López-Ibáñez et al., 2016).

7.4.1 Random Forests as a Machine Learning Model

RFs are a well-established technique in supervised ML for regression (Breiman, 2001). At its core, a RF is an ensemble of rooted and directed binary DTs. Each DT within the RF contains two types of nodes: internal (decision) nodes, characterized by a splitting criterion acting on features, and external (leaf) nodes, which bear a numerical label (see also Figure 7.4.1). The construction of DTs typically involves generating axis-aligned splits through the greedy *Classification and Regression Trees* (CART) algorithm (Breiman et al., 1998; Pedregosa et al., 2011). This algorithm recursively partitions the data, aiming to group similar labels together. It is worth noting that while the construction of 'optimal' DTs (e. g., w. r. t. minimal prediction error, given a fixed complexity) has been a subject of research interest (see, e. g., (Bertsimas and Dunn, 2017; Firat et al., 2020)), the CART algorithm remains the commonly-used approach.

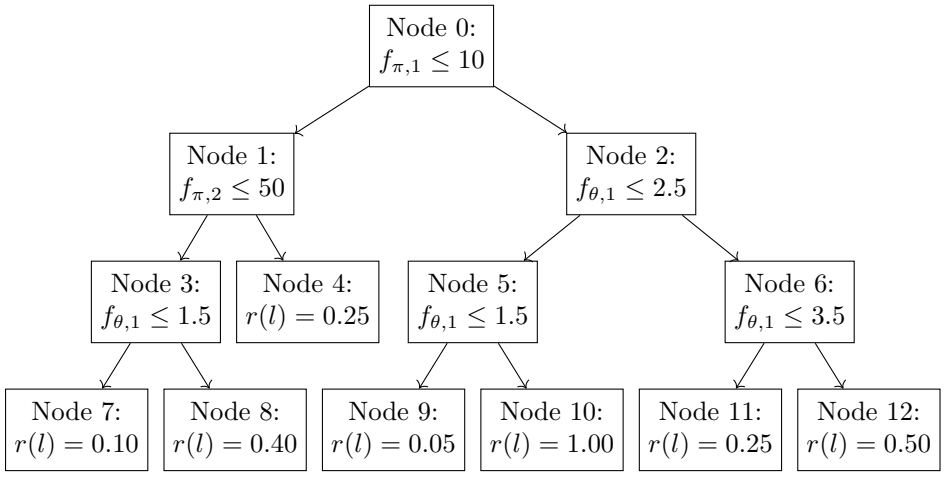

Figure 7.4.1: A single decision tree t that might be part of a forest $\mathcal{M}$. Each decision node $d \in \mathcal{D}_t$ is characterized by an (either algorithmic or instance) feature $f(d)$ along with a threshold value $v(d)$ and each leaf node $l \in L_t$ stores a learnt regression value $r(l)$.

Randomness is injected into a RF through two key mechanisms: First, each DT is constructed using a bootstrap sample (Breiman, 1996). Second, when constructing an internal node, only a random subset of features is considered to determine the locally optimal splitting criterion. The introduction of randomness through these mechanisms is advantageous because it effectively addresses the variance problem, which is commonly attributed to DTs (Breiman, 2001). By building an ensemble of trees with diverse training

sets and feature subsets, a RF tends to be more robust and less prone to overfitting.

Consider a learning sample $\mathcal{L} = \{(\boldsymbol{x_1}, y_1), \ldots, (\boldsymbol{x_k}, y_k)\}$ with k tuples, where each tuple i consists of a feature vector $\boldsymbol{x_i}$ and a numerical label y_i. The feature vector $\boldsymbol{x_i}$ is represented by an ordered list of $m+n$ features, denoted as $F = (f_{\theta,1}, \ldots, f_{\theta,m}, f_{\pi,1}, \ldots, f_{\pi,n})$, which can be divided into two disjoint subsets. $F_\pi = (f_{\pi,1}, \ldots, f_{\pi,n})$ corresponds to n instance specific features and $F_\theta = (f_{\theta,1}, \ldots, f_{\theta,m})$ to m features that characterize any algorithm $A \in \mathcal{A}$ (more details follow in Section 7.5). Given a RF $\mathcal{M}$ that was trained (e. g., by the CART algorithm) on $\mathcal{L}$, each DT $t \in \mathcal{M}$ has (internal) decision nodes D_t and (external) leaf nodes L_t. Each decision node $d \in D_t$ is defined by a (univariate) feature $f(d) \in F$ and a threshold value $v(d) \in \mathbb{R}$, which is a learnt constant. Additionally, each leaf node $l \in L_t$ stores a learnt regression value $r(l) \in \mathbb{R}$. The total set of nodes in each DT is $N_t = D_t \cup L_t, \forall t \in \mathcal{M}$.

For a given sample $\boldsymbol{x_i}$, the inference process begins by traversing each DT $t \in \mathcal{M}$ from the root decision node. At each decision node $d \in D_t$, it checks if the condition $f(d) \leq v(d)$ holds for $\boldsymbol{x_i}$. By definition, if this condition is fulfilled it selects the left-hand branch and the right-hand branch otherwise, and this process continues until a leaf $l \in L_t$ is reached. The computed regression value $r(l)$ at this leaf node represents the predicted value $r_t(\boldsymbol{x_i}) \in \mathbb{R}$ for $\boldsymbol{x_i}$ in that tree. As RFs are an ensemble method, the final regression value for a given sample $\boldsymbol{x_i}$ is the arithmetic mean over all DTs, i. e.:

$$r(\mathcal{M}, \boldsymbol{x_i}) = \frac{1}{|\mathcal{M}|} \sum_{t \in \mathcal{M}} r_t(\boldsymbol{x_i}). \tag{7.5}$$

7.4.2 Optimizing by means of Empirical Model Learning

In Section 7.4.2.1 we demonstrate how components of the RF ML model can be integrated into a combinatorial optimization problem. Specifically, we formulate a *Mixed-Integer Linear Program* (MILP) that enables us to determine the optimal algorithm for a given instance. Rather than solving this model by standard solver, in Section 7.4.2.2 we introduce a B&B approach that exploits the inherent structure of the underlying RF to deduce the logical implication for an optimal algorithm.

7.4.2.1 Embedding the Machine Learning Model into an Optimization Problem

Given a trained RF $\mathcal{M}$ and an instance π (characterized by F_π) the search for an optimal F_θ can be framed as a combinatorial optimization problem (see also (Bertsimas and Dunn, 2017; Iommazzo, 2021; Lombardi et al., 2017)). Let $y_n^t, \forall t \in \mathcal{M}, n \in N_t$ be the binary variables indicating if node n is reached in DT t. Additionally, use x_i $(i = 1, ..., m, m+$

$1, \ldots, m+n$) as variables that assign a value to feature i – n of which are going to be fixed by F_π and m of which are decidable, characterizing the optimal algorithm. The notation left(d) and right(d) denotes pointers to the left-hand and right-hand child of a node d in any DT t. With this notation, let us introduce the following optimization model (7.6)–(7.13), where (7.6) is the objective function and (7.7)–(7.13) forms the set of constraints.

$$\min \quad \frac{1}{|\mathcal{M}|} \sum_{t \in \mathcal{M}} \sum_{l \in L_t} r(l) y_l^t, \tag{7.6}$$

$$\text{subject to} \quad y^t_{\text{root}} = \sum_{l \in L_t} y_l^t = 1 \qquad \forall t \in \mathcal{M}, \tag{7.7}$$

$$y^t_{\text{left}(d)} + y^t_{\text{right}(d)} = y^t_d \qquad \forall t \in \mathcal{M}, d \in \mathcal{D}_t \tag{7.8}$$

$$x_{f(d)} \leq v(d) + M(1 - y^t_{\text{left}(d)}) \quad \forall t \in \mathcal{M}, d \in \mathcal{D}_t, \tag{7.9}$$

$$x_{f(d)} + M(1 - y^t_{\text{right}(d)}) > v(d) \qquad \forall t \in \mathcal{M}, d \in \mathcal{D}_t, \tag{7.10}$$

$$x_i \in \Theta_i \qquad \forall i = 1, \ldots, m, \tag{7.11}$$

$$x_{j+m} = f_{\pi,j} \qquad \forall j = 1, \ldots, n, \tag{7.12}$$

$$y_n^t \in \{0, 1\} \qquad \forall t \in \mathcal{M}, n \in N_t. \tag{7.13}$$

The objective function (7.6) aligns with (7.5), aiming to select leafs in a way that minimizes the joint prediction of the ensemble. Constraints (7.7) guarantee that the root node and exactly one leaf are visited in each DT t. To ensure connectivity from the root to the leaf, constraints (7.8) enforce that if we visit a decision node d in a given DT then exactly one child of that node must be visited. Constraints (7.9) and (7.10) represent the branching constraints in each decision node. Constraints (7.11) indicate that the features x_i, concerning the algorithmic configuration, must be constrained to match their feasible domain Θ_i – which may generally correspond to lower and upper bounds or integrality requirements, as per the assumption at the beginning of this section. Moreover, any variable x_{j+m} ($j = 1, \ldots, n$), representing the value of an instance-based feature, must be explicitly fixed according to the instance π, as stated by constraints (7.12). Finally, constraints (7.13) restrict the variables y_n^t to be binary.

The MILP (7.6)–(7.13) clearly reflects the structure of the learnt RF. However, solving this model directly might be challenging, particularly due to the potentially poor linear relaxation in light of the linearized disjunctive constraints (7.9) and (7.10). Moreover, the amount of variables y_n^t grows exponentially with the height of the learnt tree t. While alternative formulations or heuristic solutions could be explored (see, e. g., (Hutter et al.,

2011; Iommazzo, 2021)), we propose a different approach in Section 7.4.2.2 that addresses the problem through a B&B approach.

7.4.2.2 A Branch-and-Bound Approach for solving the Optimization Problem

The nature of DTs allows them to be evaluated very efficiently. Recall that we assumed that for every DT $t \in \mathcal{M}$, the internal (decision) nodes can be partitioned into two sets of nodes: $D_t(F_\theta)$ and $D_t(F_\pi)$; i. e., the sets of internal nodes that use algorithmic features F_θ or instance-based features F_π as a splitting criterion. Given an instance that is characterized by F_π, we begin by constructing an equivalent DT t' that depends only on features F_θ as follows (see also (Iommazzo, 2021)). We start with an empty DT t' and traverse DT t from the root, inferring from F_π until we reach the first internal node or leaf $u \in D_t(F_\theta) \cup L_t$. We copy this node as the root node of t' and if this node is a leaf node then we are done. Otherwise, we continue to both children of u in t and proceed with the process of inference (given F_π) until we reach the next decision nodes (or leafs) $u_l(u), u_r(u) \in D_t(F_\theta) \cup L_t$. We add these nodes to t' and also include the directed arcs (u, u_l) and (u, u_r) to t', where $u \in D_t(F_\theta)$, $u_l, u_r \in D_t(F_\theta) \cup L_t$. This process continues recursively for any node $u_l, u_r \notin L_t$. In practice, this can be achieved by traversing every DT $t \in \mathcal{M}$ in a depth-first, from left to right, fashion, as sketched in Algorithm 7.1. As an example, for the DT depicted in Figure 7.4.1 and given a problem characterized by features F_π, applying Algorithm 7.1 may yield one of the DTs depicted in Figure 7.4.2.

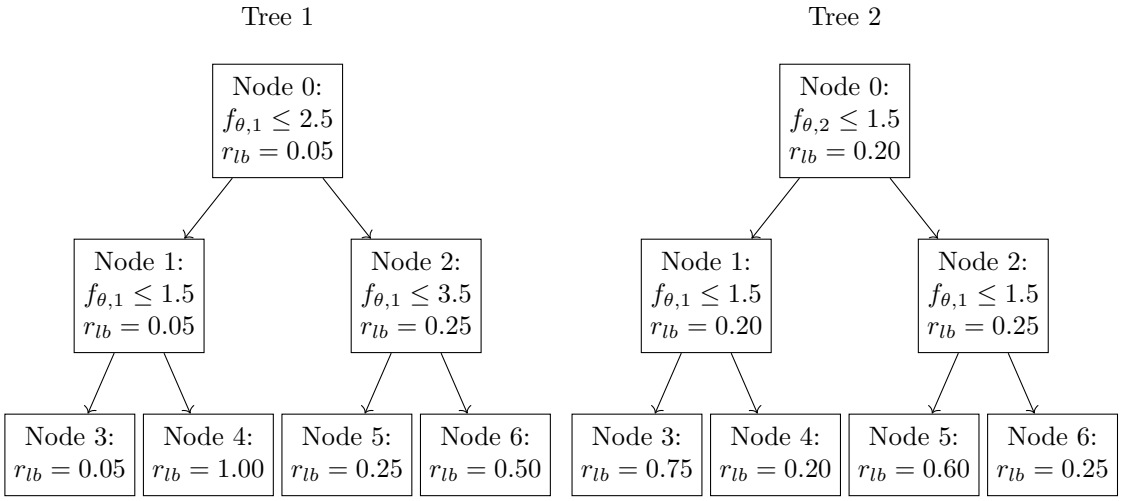

Figure 7.4.2: A collection of two trees t' that depend only on F_θ. The left-hand tree can be constructed by applying Algorithm 7.1 to the tree in Figure 7.4.1 given that, e. g., $f_{\pi,1} > 10$.

Now let us consider having a RF $\mathcal{M}'$, in which each tree $t' \in \mathcal{M}'$ has been processed

Algorithm 7.1 Depth-first search traversal of a tree t to obtain a tree t' that, given F_π, behaves identical and depends only on F_θ.

```
1: function TreeReduction(t, F_π)
2:     node ← getRoot(t)
3:     node ← traverseTree(t, node, F_π)
4:     t' ← initTreeByRoot(node)
5:     if ¬ leaf(node) then
6:         stack.push([rightChild(node), node])
7:         stack.push([leftChild(node), node])
8:     end if
9:     while stack is not empty do
10:        node, parent ← stack.pop()
11:        node ← traverseTree(t, node, F_π)
12:        if ¬ leaf(node) then
13:            stack.push([rightChild(node), node])
14:            stack.push([leftChild(node), node])
15:        end if
16:        t'.appendChild(parent, node)
17:    end while
18:    return t'
19: end function
```

according to Algorithm 7.1. In the trivial case, i. e., if $\mathcal{M}'$ is formed by only a single DT t', computing the (predictably) optimal algorithm corresponds to selecting the leaf $l' \in L_{t'}$ with (without loss of generality) minimal $r(l')$. Following that, the branching constraints that are encountered on the path from the root node of t' to l' form the logical implication for the optimal algorithm.

In the more general case where there is more than one DT in the RF $\mathcal{M}$, we propose a novel way to search for the optimal algorithm through a B&B procedure (Land and Doig, 1960). To facilitate that approach, we use $r_{lb}^{t'}(d')$ as a lower bound for the regression value that can be found in the subgraph of t', rooted at the decision node $d' \in \mathcal{D}_{t'}$. We initialize these values for all decision nodes to '∞'; and for the leaf nodes $l' \in L_{t'}$ it follows that $r_{lb}^{t'}(l') = r(l')$. Then, $\forall t' \in \mathcal{M}'$, starting at each leaf $l' \in L_{t'}$ we traverse the DT upwards, towards its root node. During this process, we overwrite the label $r_{lb}^{t'}(p)$ with $r_{lb}^{t'}(d)$, where p is the parent of d, iff $r_{lb}^{t'}(p) > r_{lb}^{t'}(d)$ holds (see also Figure 7.4.2).

The search for an optimal algorithm by means of B&B follows a straightforward process (Algorithm 7.2). We initialize a series of pointers that reference the root node of each DT $t' \in \mathcal{M}'$ and the initial bound corresponds to $\sum_{t' \in \mathcal{M}'} r_{lb}^{t'}(\text{root})$. We consider a node in the B&B search tree to be feasible, iff all pointers reference leaf nodes. In that case, the branching constraints encountered on each path from root to leaf in all trees $t' \in \mathcal{M}'$

Algorithm 7.2 B&B to determine the optimal algorithm, given a RF $\mathcal{M}'$, in which each tree was processed according to Algorithm 7.1.

```
function BranchAndBound(Reduced Random Forest M')
    M' ← propgatateBoundsFromLeafs(M')
    bestFeasible ← ∞
    bestState ← initializeRootState(M')
    if bestState.isFeasible(M') then
        return bestState
    else
        pq = priorityQueue()
        pq.push(bestState.branch(M'))
    end if
    while ¬ pq.isEmpty() do
        state = pq.pop()
        if state.getBound(M') ≥ bestFeasible then
            return bestState
        else if state.isFeasible(M') then
            if state.getValue(M') < bestFeasible then
                bestState ← state
                bestFeasible ← state.getValue(M')
            end if
        else
            pq.push(state.branch(M'))
        end if
    end while
end function
```

form the logical implication for the optimal algorithm. Following the scheme of best-first search, to add a branching constraint, we start by selecting the DT t' in which the pointer references a decision node d' with minimal value of $r_{lb}^{t'}(\cdot)$ compared to all other DTs. Based on that decision node d' – i. e., its splitting criterion characterized by feature $f(d')$ and threshold value $v(d')$ – we add two branches to our B&B search tree (with corresponding branching constraints $f(d') \leq v(d')$ and $f(d') > v(d')$, respectively). For each newly added branch, we move the pointer in DT t' accordingly and continue the process of inference in that tree (given F_π and existing branching constraints), until we reach a new decision node that splits on F_θ (or a leaf). In all other DTs, the newly introduced branching constraint may make further traversal now possible; therefore, we also attempt to move these pointers. Precisely, for any DT $t' \in \mathcal{M}'$, we check if the pointer currently references a decision node d' that acts on the same feature $f(d')$ as the most recent branching constraint. If either $f(d') \leq v(d')$ or $f(d') > v(d')$ is now already implied by an added branching constraint, we move the pointer accordingly. Then, we continue the process of inference (given F_π

and existing branching constraints), until we reach a new decision node d' that acts on a feature F_θ (or a leaf). After that, we calculate the updated bound for the current node in the B&B search tree. We continue processing our B&B search tree for as long as we have found no feasible solution and until no unexplored node remains in the B&B search tree with a bound that is less than the value of the incumbent solution. Figure 7.4.3 illustrates this procedure using the B&B search tree that results from applying Algorithm 7.2 to the two DTs in Figure 7.4.2.

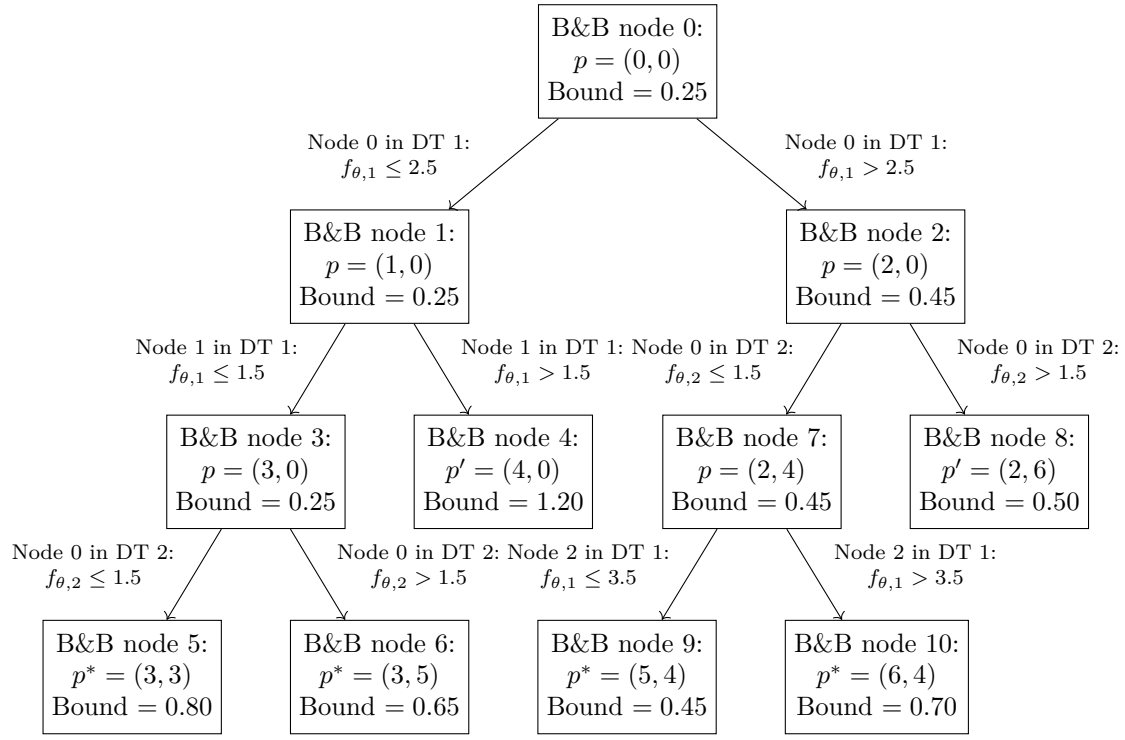

Figure 7.4.3: An illustration of the B&B procedure to determine the implication for the optimal algorithm. This figure results from applying Algorithm 7.2 to the DTs in Figure 7.4.2. Each B&B node is characterized by its identifier, the local pointer p, which shows the currently active nodes in trees 1 and 2 (in Figure 7.4.2), respectively, and the local bound. Feasible solutions are highlighted by a pointer p^* and corresponding B&B nodes pruned by feasibility. B&B nodes that can be pruned by bound (B&B nodes 4 and 8) are highlighted by a pointer p'. As no open node remains, the optimal solution corresponds to the B&B node 9 and the path from the root to that node forms the implication $2.5 < f_{\theta,1} \leq 3.5$ and $f_{\theta,2} \leq 1.5$ for the optimal algorithm.

7.5 Computational Experiments and their Numerical Results

For the remainder of this paper, we consider MILP as a problem space $\mathcal{P}$ and the solver Gurobi Optimizer 9.5.2 (Gurobi Optimization, LLC, 2023) as an algorithm space $\mathcal{A}$. A

brief overview of these concepts is provided in Section 7.5.1. In Section 7.5.2, the generation procedure of the dataset is described. This includes an exploratory analysis to see if there may be any merit to approaching this dataset from the perspective of the ASP. Finally, Section 7.5.3 discusses the numerical results of the computational experiments in detail.

7.5.1 Problem Space and Algorithms

Following Gleixner et al. (2021), a canonical form of a MILP optimization problem is the following. Let $\mathcal{V} \in \mathbb{N}$ be the number of variables (columns) and $\mathcal{C} \in \mathbb{N}$ be the number of constraints (rows). Variables are further categorized into continuous variables $\mathcal{V}_C$, general integer variables (excluding binary) $\mathcal{V}_I$, and binary variables $\mathcal{V}_B$, such that $\mathcal{V}_C + \mathcal{V}_I + \mathcal{V}_B = \mathcal{V}$. The coefficient matrix is denoted by $H \in \mathcal{Q}^{\mathcal{C}\times\mathcal{V}}$. The left-hand and right-hand side vectors, that place lower and upper bounds on each constraint, are represented by $l^H \in \mathbb{R}^{\mathcal{C}}$ and $u^H \in \mathbb{R}^{\mathcal{C}}$, respectively. Additionally, the lower and upper bounds of the decision variable vector x are specified by the vectors $l^x, u^x \in \mathbb{R}^{\mathcal{V}}$. Finally, the objective coefficient vector is denoted by $c \in \mathbb{R}^{\mathcal{V}}$. Using this notation, the canonical form of any MILP is the following:

$$\min \left\{ c^\top x : l^H \leq Hx \leq u^H, l^x \leq x \leq u^x, x \in \{0,1\}^{\mathcal{V}_B} \times \mathbb{Z}^{\mathcal{V}_I} \times \mathbb{R}^{\mathcal{V}_C} \right\}. \tag{7.14}$$

In our work, we use the commercial MIP solver Gurobi Optimizer 9.5.2 (Gurobi Optimization, LLC, 2023) as a configurable algorithm that is applicable to problems of type (7.14). Comparative studies have demonstrated that the default configuration of this solver is already highly optimized, making it an effective choice even when compared to other state-of-the-art solvers (Hutter et al., 2010): In fact, improvements in performance on homogeneous problems through parameter tuning by means of the ACP were relatively modest, indicating the solver's robust default settings.

Given the computational intractability of exhaustively sampling the entire parameter space for this solver, as in similar studies (Iommazzo et al., 2020a; Iommazzo et al., 2020b), we focus on a subset of parameters that impact the B&C algorithm's execution on a high level. Specifically, we consider the following parameters, along with their respective domains (refer to (Gurobi Optimization, LLC, 2023) for details):

1. `Cuts`: Determines global cut aggressiveness. Domain: $\{-1, 0, 1, 2, 3\}$

2. `Disconnected`: Indicates the effort spent by the solver in identifying independent sub-models within the problem. Domain: $\{-1, 0, 1, 2\}$.

3. `Heuristics`: Designates the time spent applying MIP heuristics. Domain: $\{0, 5, 10, 15\}\%$.
4. `(Node)Method`: Sets the method used for solving linear relaxations. Domain: $\{-1, 0, 1, 2\}$.
5. `Presolve`: Correlates with the time spent during presolve. Domain: $\{-1, 0, 1, 2\}$.
6. `Symmetry`: Detection level for identifying the symmetry. Domain: $\{-1, 0, 1, 2\}$.

This results in 5120 distinct configurations. The DS is obtained by leaving all parameters at -1, except for `Heuristics`, which has a default value of 5%. For the remainder of this paper, any instantiation of Gurobi Optimizer using one of these configurations is referred to as an algorithm (or solver).

7.5.2 Dataset

Our experiments make use of the MIPLIB, which originated from the work of Bixby et al. (1992) and is currently in its sixth iteration. This ongoing project provides a carefully curated library of problem instances, collected in a data-driven and scientifically rigorous manner (Gleixner et al., 2021). The MIPLIB aims to represent the diversity of challenges faced by both academic and industrial researchers in various domains, making it a widely accepted benchmark for comparing different solvers or versions thereof (Mittelmann, 2020). It is worth noting that this diversity contrasts sharply with most works discussed in Section 7.2 (except for (Berthold et al., 2022; Berthold and Hendel, 2021), who also relied on the MIPLIB). A common characteristic in these works involves focusing on (homogeneous) instances specific to a particular problem. This generally leads to the development of highly problem-specific features that may not readily be transferred to related problems within the same domain.

In the most recent iteration of the MIPLIB, Gleixner et al. (2021) tested eight different MIP solvers, each using its default configuration, on a set of 1065 benchmark instances. Notably, each of these eight solvers demonstrated superior performance on at least one of the 1065 instances. While performance variability is inherent to MIP (Lodi and Tramontani, 2014), this observation underscores the potential advantages of employing algorithm selection in this context.

7.5.2.1 Generation Procedure and Preprocessing

To facilitate data collection, we focus on a subset of 677 instances from the MIPLIB labeled as 'easy'. It is important to note that the label 'easy' should be approached

with caution, as the only other categories are 'hard' and 'open', comprising 144 and 244 instances, respectively. Additionally, we only considered problems that are inherently solvable, meaning instances that are neither infeasible nor unbounded. This filtering results in 629 instances. Further, we excluded an additional four instances due to their extensive main memory requirements, which would adversely affect the data generation phase of our study.[1]

We collected the performance data for each of the 5120 possible algorithms (see Section 7.5.1) on each of the 625 instances, by running the solver five times, utilizing five different random seeds. Consequently, we obtained a total of $5120 \times 625 \times 5 = 16$ million data points. This dataset was generated by running the solver with a single thread, a time limit of 60 seconds, and an allocation of sufficient (8 GB of) RAM on compute nodes equipped with Intel® Xeon® Gold 6126 Processors.

For a given algorithm, instance, and random seed, we collected the values of the primal objective bound (incumbent solution) z_p, the dual objective bound z_d, and the runtime. Gurobi Optimization, LLC (2023) defines the MIP gap as follows for minimization problems, which we also add to the dataset:

$$\text{MIP gap} \coloneqq \frac{|z_p - z_d|}{|z_p|}. \tag{7.15}$$

As Laporte and Toth (2022) stress, this definition must not be confused with optimality gaps, which measure the deviation from z_p to a (known) optimal objective value z^*. We use the MIP gap because, outside of benchmarks, z^* is generally not known in advance, making MIP gaps a well-suited surrogate to monitor how close a solver is to certifying optimality for a solution.

In situations where no primal solution z_p is found within the allotted runtime, MIP gaps according to (7.15) are, in principle, undefined. Initially, in these cases, the assigned MIP gap label was '∞'. To address this numerical issue, the following transformation was applied once all data had been collected. First, the 95$^{\text{th}}$-percentile MIP gap, denoted by $\text{gap}_{95\%}$, was identified – representing the value for which 95% of all recorded MIP gaps are smaller or equal to. Then, the MIP gaps that exceeded this threshold were set to 10 times $\text{gap}_{95\%}$, inspired by the commonly used PAR-10 scoring (see, e. g., (Bischl et al., 2016)).

Overall, we spent approximately 20.53 CPU-core-years to generate the dataset. At several points, we performed analysis and consistency checks on our dataset. These inspections revealed that the solver could behave erraticly or numerically unstable in presence of a few algorithms. This had caused the solver to classify some problems as, e. g., unbounded

[1] Namely, NEOS-4332801-SERET, NEOS-4332810-SESIA, T11NONREG, and SUPPORTCASE11.

or infeasible, or to return implausible solutions. As pointed out by Berthold and Hendel (2021), Eggensperger et al. (2019), and Tang et al. (2020), care must be taken when adjusting any default algorithm, as this may lead to unforeseen effects, e. g., exceptions or the return of implausible results. In the extreme case, e. g., when purely optimizing runtime, the 'best-performing' algorithm may – mistakenly – be the one that causes the algorithm to terminate prematurely, e. g., by returning an incorrect result. Still, it is important to acknowledge that these misclassifications may actually, i. e., within numerical tolerance, be considered to be correct. Nevertheless, in order to have a clean dataset, we removed six further instances.[2] Therefore, the final dataset consists of $5120 \times 619 \times 5 = 15\,846\,400$ entries.

7.5.2.2 Exploratory Analysis

In this section, we provide a first glance at the dataset that was generated according to Section 7.5.2.1. Note that, for the remainder of this paper, we abbreviate an instance-seed-pair by an instance. In the context of our study, when comparing two different algorithms A_1 and A_2 on a single instance purely based on runtimes and MIP gaps, it is straightforward to establish a dominance relation: $A_1 \succ A_2$ if $t_1 < t_2 \leq 60$ seconds (i.e., $g_1 \approx 0\%$) or $t_1 = t_2 = 60$ seconds and $g_1 < g_2$. When taking multiple instances into account, establishing an overall dominance relation becomes non-trivial. As a thought experiment, consider Table 7.5.1 where each algorithm dominates the others on one instance. In particular, the algorithm that yields the shortest runtime on average has the worst gaps and vice versa. Clearly, an optimal (ex-post) policy would be to select algorithm i on instance i, $\forall i \in \{1, 2, 3\}$. This example illustrates that there may be an inherent trade-off in MIP between short runtimes and small gaps (given a runtime limit). This will be further explored in the experiments (refer to Section 7.5.3).

Table 7.5.1: An example of three algorithms on three instances where each algorithm dominates the other ones on exactly one instance.

	Instance 1		Instance 2		Instance 3		Average	
	time	gap	time	gap	time	gap	time	gap
Algorithm 1	15s	0%	60s	75%	60s	75%	45s	50%
Algorithm 2	20s	0%	10s	0%	60s	300%	30s	100%
Algorithm 3	60s	10%	60s	10%	60s	10%	60s	10%

In any case, for the remainder of this section, we are going to investigate our dataset as

[2] Namely, BLEY_XS2, CONTROL20-5-10-5, DALE-CTA, NEOS-1223462, NEOS-3083784-NIVE, and TRANSPORTMOMENT.

follows:

- For each of the 5120 algorithms, average runtime and MIP gap are calculated for solving the 3095 instances. This allows us to identify the SBS (see (7.2) in Section 7.3) w. r. t. runtime and MIP gap.
- For each of the 3095 instances, the algorithms are sorted lexicographically by dominance, i. e., first by MIP gap and then by runtime. For each instance, we assign a rank to every algorithm, proceeding in ascending order. This way, the first rank on a particular instance corresponds to the solver that dominated all others. For each rank 1 to 5120, we aggregate the runtime and MIP gaps across all 3095 instances. Consequently, rank 1 (rank 5120) corresponds to the VBS (virtual-worst solver) (see (7.1) in Section 7.3).

The analysis aims to address key questions to evaluate the viability of applying the methodology proposed in Section 7.4. These questions include:

1. What is the performance gap between the DS and SBS? A marginal gap might suggest little benefit in deviating from the DS when using a single algorithm for solving all instances.
2. How significant is the performance gap between the SBS and VBS? If the gap is marginal, it could be reasonable to stick with the SBS instead of employing algorithm selection.
3. On average, how much does a competitive selector need to close the performance gap between the DS and VBS to match or surpass the SBS?
4. Does the VBS consist of a large and diverse set of algorithms, or is it composed of only a small portfolio? If the latter is true, considering per-set algorithm selection might be more effective (see (Kerschke et al., 2019) and references therein).

To address these questions, Figure 7.5.1 provides an initial insight by showcasing the average runtime of the solvers across the complete dataset. The DS exhibits an average runtime of approximately 35.47 seconds, making it the 501^{st}-single-best and placing it into the top ten percentile of all solvers. The SBS records an average runtime of around 32.81 seconds, surpassing the DS by 2.66 seconds, constituting a 7.5% decrease. In contrast, the VBS achieves an average runtime of 25.10 seconds, outperforming the SBS by 7.71 seconds, representing a 23.5% decrease. This implies that the SBS manages to close only around 25.7% of the performance gap between the DS and VBS. The figure also highlights

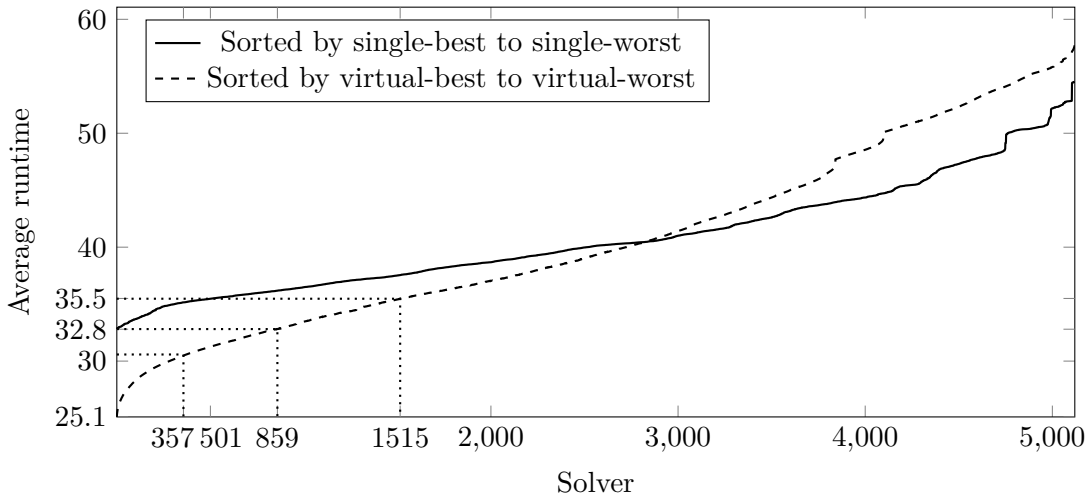

Figure 7.5.1: Average runtime for all solvers.

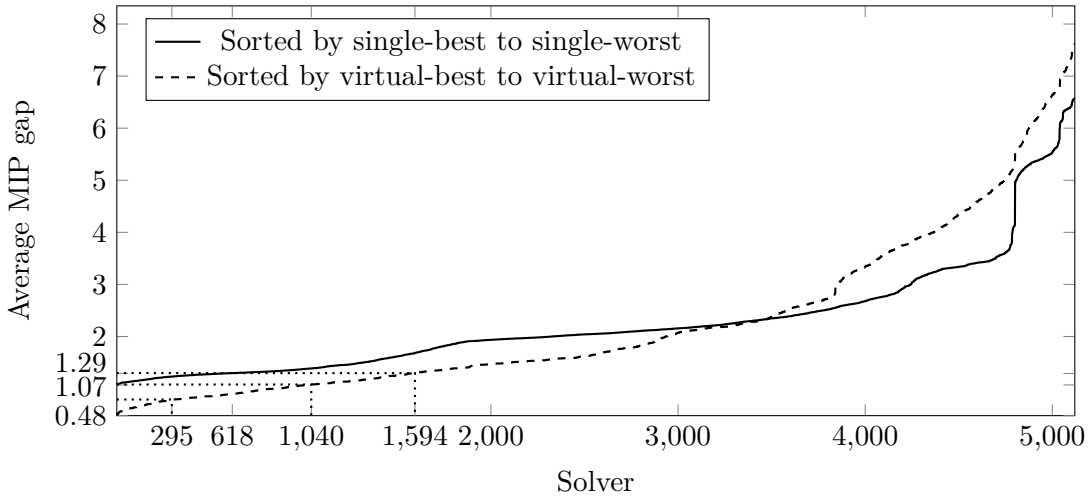

Figure 7.5.2: Average MIP gap for all solvers.

the potential of algorithm selection, as the DS is rivalled by the virtual-1515th solver, and a virtual-859th solver would yield an average runtime nearly identical to that of the SBS.

Figure 7.5.2 reveals analogous observations regarding the average MIP gap. The DS performs as the 618^{th} single-best with an average gap of 1.29, closely resembling the virtual-1594^{th} solver. In comparison, the SBS achieves an average gap of 1.07, almost mirroring the virtual-1040^{th} solver. Notably, the VBS attains an average gap of only 0.48, indicating that the SBS reduces about 27.2% of the performance gap between the DS and VBS.

Analyzing the VBS reveals its utilization of 1264 distinct algorithms across the 3095

instances. Notably, this suggests that different solvers might outperform others on the same base instance when using different random seeds, underscoring the inherent performance variability of MIP solvers (Lodi and Tramontani, 2014). Among these 1264 distinct solvers, the most frequently used occurs 46 times, the second most frequent 25 times, and the third most frequent 23 times. In contrast, 668 solvers are only applied to a single instance by the VBS. Surprisingly, the most frequently used solver exhibits comparatively poor average performance: 39.79 seconds runtime and a 2.02 MIP gap (compare with Figures 7.5.1 and 7.5.2). In contrast, the SBSs w. r. t. runtime and MIP gap are only used by the VBS in 0 and 4 cases, respectively, and the DS is absent altogether. This result is surprising for two reasons. First, it could be interpreted as confirming that algorithms with poor average performance may still play a vital role in a competitive selector. Second, addressing the ASP with a portfolio of algorithms determined through the ACP (refer to Section 7.2.2) is unlikely to ever include such algorithms.

Lastly, Figure 7.5.3 offers a brief glimpse at the tunability (Probst et al., 2019). The tunability concerns the ceteris paribus effect of altering a single parameter of the DS.

In summary, the insights derived from the analysis of Figures 7.5.1 and 7.5.2 are encouraging. Undeniably, there is a significant potential for performance degradation compared to even the DS; however, a more optimistic perspective is that even a virtual-859^{th} (or virtual-1040^{th}) solver can outperform the SBS in terms of average runtime (or average MIP gap). Hence, by adhering to the methodology proposed in Section 7.4, achieving a selector that consistently opts for a solver within the top 16.8% of all solvers would already surpass the performance of the SBS.

7.5.3 Numerical Results for Offline Learning and Online Inference

Following best-practices, k-fold cross-validation is used, as commonly recommended in the literature (Bischl et al., 2016; Eggensperger et al., 2019; Kerschke et al., 2018). This means that the 3095 instances are randomly shuffled, before dividing them into $k = 5$ folds (akin to (Berthold et al., 2022; Berthold and Hendel, 2021)). This implies that the same base instance may be present in the training set multiple times (with different random seeds) or not at all. For each iteration of cross-validation, we use $(k - 1)$ folds with the associated performance data of the 5120 algorithms to train the RF (see Section 7.5.3.1). The remaining fold is reserved for testing. In more detail, for each instance in the test set we leverage the trained RF and apply the B&B methodology described in Section 7.4.2 to decide which algorithm to use. This allows us to compare the performance of the proposed approach (refer to Section 7.4) against the DS, SBS, and VBS (see Section 7.5.3.2). As illustrated in Figure 7.5.4, we repeated this process k times, such that each of the k subsets

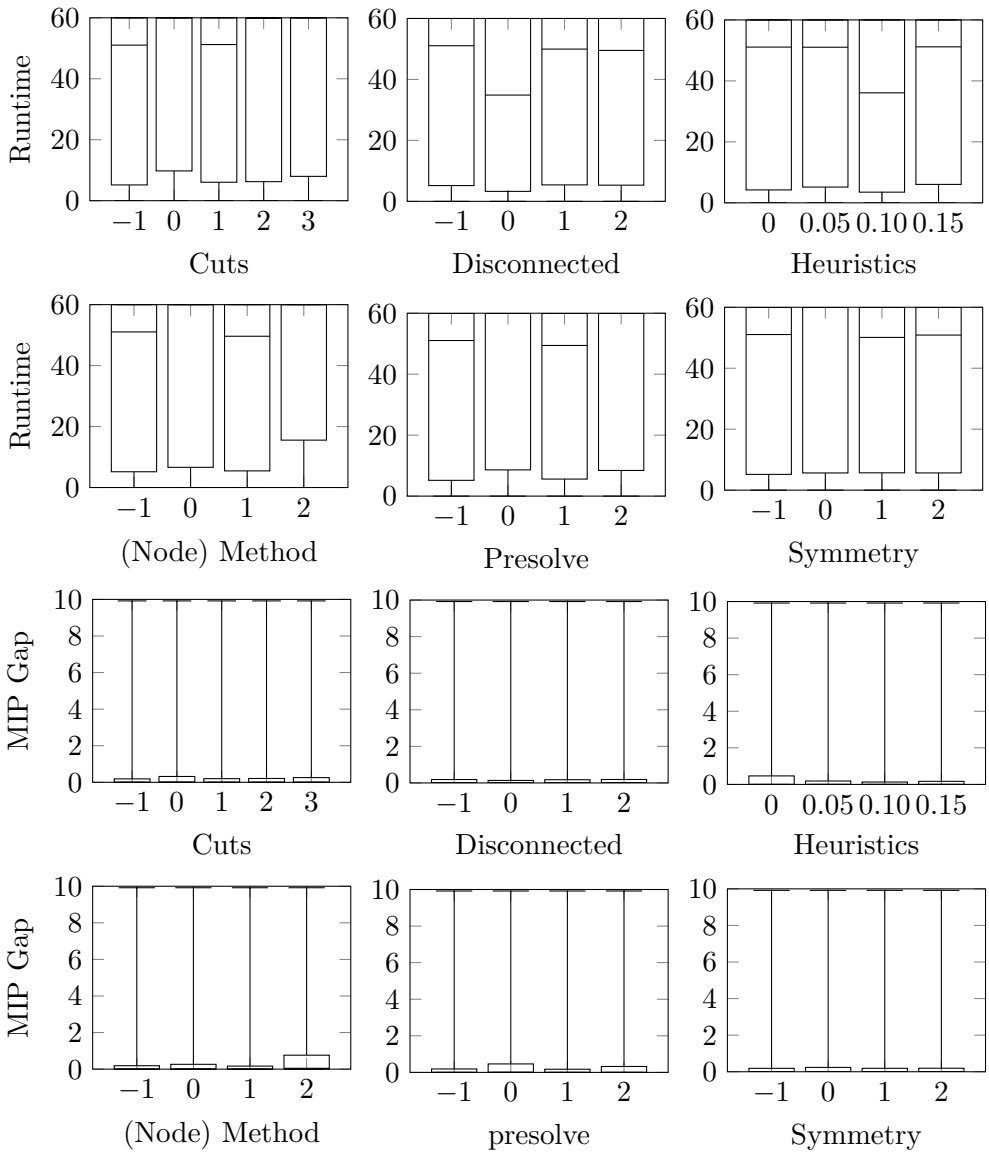

Figure 7.5.3: This figure illustrates the tunability, i. e., the ceteris paribus effect of adjusting only a single parameter of the DS. The box plots highlight the mean, maximum, and minimum values along with the 25^{th} and 75^{th} percentile.

is used for testing exactly once.

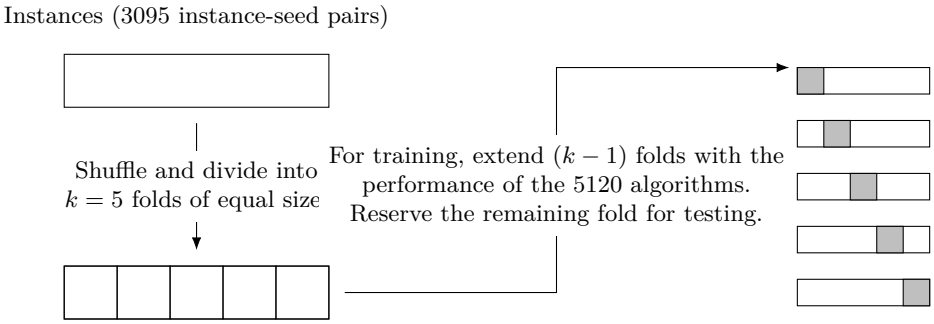

Figure 7.5.4: The process of k-fold cross validation illustrated for $k = 5$.

7.5.3.1 Offline Learning

Given our interest in using heterogenous benchmark instances, we must abstain from using highly problem-specific features, which are generally used in the context of the ASP (see (Kerschke and Trautmann, 2019) and references therein). Moreover, due to the large scope of algorithms under consideration, probing for dynamic information is not a feasible option. Instead, we are curious if high-level features of MILPs, which are generally applicable across a wide range of problems, can already be sufficiently informative for a successful ASP approach. It is important to note that there are limited approaches in the literature (see, e. g., (Loreggia et al., 2016)) that acknowledge the challenging nature of feature engineering and the fact that the resulting features are typically confined to very specific problems. In that regard, we use two types of generic MILP features to characterize problem instances: *Size Related Properties* (SRP) and *Constraint Related Properties* (CRP) that were introduced by Gleixner et al. (2021). The SRP are the number of variables $\mathcal{V}$, the number of constraints $\mathcal{C}$, as well as the non-zero density of the matrix. These properties also include the proportion of continuous ($\mathcal{V}_C$), binary ($\mathcal{V}_B$), and integer variables ($\mathcal{V}_I$) w. r. t. the total number of variables $\mathcal{V}$. Finally, the SRP include the proportion of implicit integers, indicating variables for which integrality is implied by the model. The CRP is discussed in-depth by Gleixner et al. (2021) and for completeness, we include their categorization of linear constraints in Table 7.5.2. The CRP measure the proportion with which these categories occur. A given constraint may fall into multiple categories (e. g., set packing and knapsack). In such cases, the authors suggest using the most specific (i. e., topmost in Table 7.5.2) type. Apart from the classification of linear constraints in Table 7.5.2, the CRP also includes the share of (fundamentally non-linear) Indicator Constraints. Such constraints may be handled internally by the solver in

a more advantageous way, rather than relying on the (in)famous big-M linearization.

Table 7.5.2: Classification of linear constraints as illustrated by Gleixner et al. (2021) (sorted from most specific to most general). In this table, $a, b, c \in \mathbb{R}$ denote coefficients whereas x, y, and z represent decision variables.

Type	Linear constraints…
Free	with no finite side
Singleton	with a single variable
Aggregation	of the form $ax + by = c$
Precedence	of the form $ax - ay \leq b$
Variable Bound	of the form $ax + by \leq c, x \in \{0, 1\}$
Set Partitioning	of the form $\sum x_i = 1, x_i \in \{0, 1\} \forall i$
Set Packing	of the form $\sum x_i \leq 1, x_i \in \{0, 1\} \forall i$
Set Covering	of the form $\sum x_i \geq 1, x_i \in \{0, 1\} \forall i$
Cardinality	of the form $\sum x_i = k, x_i \in \{0, 1\} \forall i, k \geq 2$
Invariant Knapsack	of the form $\sum x_i \leq b, x_i \in \{0, 1\} \forall i, b \in \mathbb{N}, b \geq 2$
Equation Knapsack	of the form $\sum a_i x_i = b, x_i \in \{0, 1\} \forall i, b \in \mathbb{N}, b \geq 2$
Binpacking	of the form $\sum a_i x_i + ax \leq a, x, x_i \in \{0, 1\} \forall i, a, a_i \in \mathbb{N} \forall i, a \geq 2$
Knapsack	of the form $\sum a_i x_i \leq b, x_i \in \{0, 1\} \forall i, b \in \mathbb{N}, b \geq 2$
Integer Knapsack	of the form $\sum a_i x_i \leq b, x_i \in \mathbb{Z} \forall i, b \in \mathbb{N}$
Mixed Binary	of the form $\sum a_i x_i + \sum b_j z_j \{\leq, =\} c, x_i \in \mathbb{Z} \forall i, z_j \in \mathbb{R} \forall j$
General Linear	with no special structure

The SRP and CRP may be considered to be appropriate features due to their alignment with the ideal properties outlined by Kerschke et al. (2019): they are informative, interpretable, cheaply computable, and in particular broadly applicable to the domain of MILP. Importantly, for a given problem these features are invariant to mathematically equivalent transformations (e. g., shuffling the order of rows or columns).

We implemented the RF using `scikit-learn` version 1.0.0 (Pedregosa et al., 2011). Hyperparameters were left at their default values (the RF consists of 100 DTs). However, to account for regularization and randomness, we tuned two hyperparameters through nested cross-validation (Stone, 1974) (with four internal folds) by grid-search. The first hyperparameter is `min_samples_split` (i. e., the minimal number of samples required in each node to consider splitting further) for which we considered 0.01%, 0.1%, and 1% of the training set. The second hyperparameter is `max_features` (i. e., the number of random features considered in each decision node for splitting) for which we considered 5 (i. e., approximately $\sqrt{|F|}$) or 10 features. Given that tree-based learners are invariant to scaling, no special feature engineering is required; in particular, this also concerns the encoding of the algorithm's parameters (Hastie et al., 2009). We repeated our experiments thrice, to study the effects of using one of the following three training labels:

1. Runtime, scaled to a maximum value of 1.

2. MIP gap, scaled to a maximum value of 1.

3. The equally weighted sum of both of the above, divided by a factor of 2.

Table 7.5.3 presents the mean coefficients of determination (R^2) and mean absolute errors of the models, after the training procedure. Additionally, these metrics are included for the test sets (for that purpose, by associating each instance with the performance data of the 5120 algorithms). While these values suggest an acceptable model fit, it is essential to note that standard ML metrics might have limited utility in the context of combinatorial optimization pipelines (Accorsi et al., 2022). Our primary focus remains on assessing whether these learnt models facilitate effective decision-making (refer to Section 7.5.3.2).

Table 7.5.3: Average coefficient of determination (R^2) and mean absolute errors of the trained models for the three considered labels (averaged across all learnt models).

	Runtime		MIP Gap		Weighted Sum	
	Training Set	Test Set	Training Set	Test Set	Training Set	Test Set
R^2	0.873	0.858	0.841	0.823	0.875	0.860
MAE	0.078	0.084	0.068	0.073	0.071	0.076

It is worth noting that preliminary experiments have indicated that the effects of (sequential forwards or backwards) feature selection have a negligible impact on these scores; therefore, we did not pursue this any further. Figure 7.5.5 shows the impurity-based relative feature importance (Breiman, 2001) when using either the runtime, MIP gap, or the weighted sum as training labels. Unsurprisingly, the number of variables, constraints, and non-zero density appear to have the largest feature importance. And considering that the cardinality of most algorithm parameters is relatively small (refer to Section 7.5.1), which might put them at a disadvantage when it comes to impurity-based feature importance, their values also stand out. Noticeably, the importance of the parameters `Disconnected` and `Symmetry` seems to be almost negligible, which we find surprising in light of Figure 7.5.3.

7.5.3.2 Online Inference

After training a RF according to Section 7.5.3.1, it remains to leverage the model to determine the (predictably) optimal algorithm for a given instance. Let it be known that directly solving the model (7.6)–(7.13) presented in Section 7.4.2.1 results in a computational overhead far exceeding the potential performance gains through algorithm selection (see also Figure 7.5.1). To address this, the B&B procedure suggested in Section 7.4.2.2 was implemented. Using this approach, the optimal algorithm can be determined in a

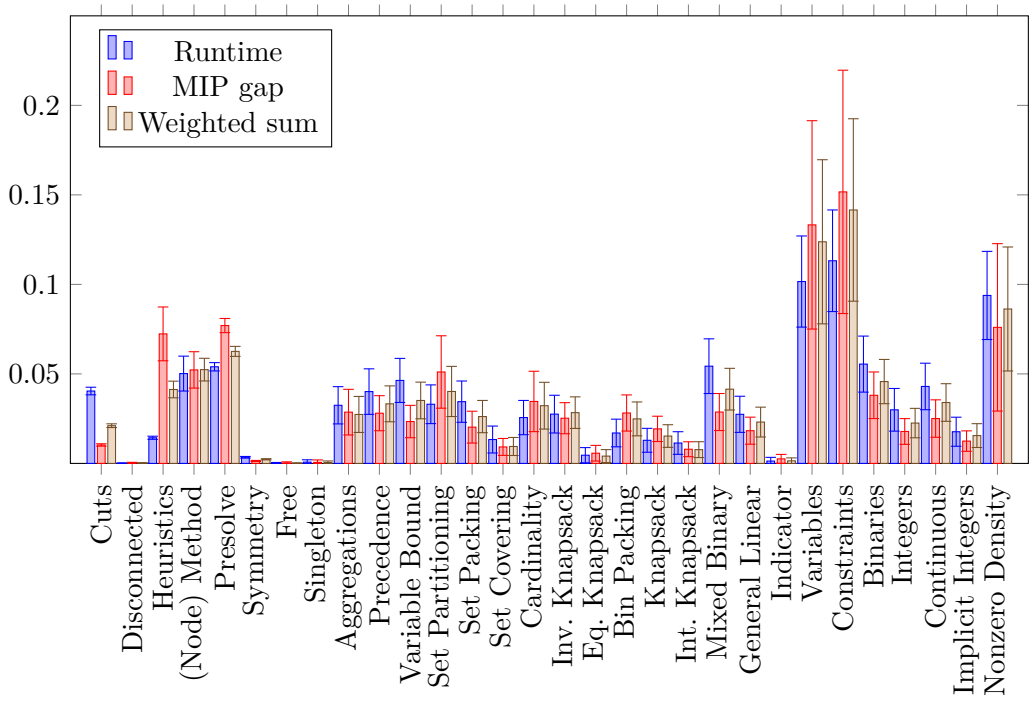

Figure 7.5.5: Gini importance (normalized to $[0, 1]$) and the corresponding standard deviation. The values are averaged (for each training label) over all models learnt during the five rounds of cross-validation.

matter of a few milliseconds, primarily attributed to the time spent building the reduced trees (see Algorithm 7.1). The subsequent B&B process (see Algorithm 7.2) takes only a fraction of this time. This gives the proposed methodology the desired property of near-instantaneous online selection, as deemed necessary in Section 7.3.

The performance of the *Solver found by means of Algorithm Selection* (SAS) was evaluated against that of the DS, SBS, and VBS on each instance of the test set. As a reminder, the VBS acts as an oracle, fulfilling condition (7.1) (refer to Section 7.3). In contrast, for each test set, the SBS is defined as the algorithm A^* that fulfils condition (7.2) (refer to Section 7.3) on the corresponding training set w. r. t. the given training label.

Tables 7.5.4–7.5.6 illustrate the numerical results obtained during the five rounds of cross-validation when using runtime, MIP gap, or the weighted sum as training label, respectively. Each table shows the number of unique algorithms used to solve the 3095 instances (# algorithms), the number of instances that have been solved optimally (# optimal), and the number of instances for which at least a feasible solution was found within the runtime limit (# feasible). Moreover, the tables show the average (μ) runtime and MIP gap as well as their respective standard deviation (σ). All of these values are

shown for the SAS, DS, SBS, and VBS. The tables also include an instance-wise comparison, indicating the share of wins, ties, and losses with respect to runtime and MIP gap. A win (lose) w. r. t. runtime or MIP gap denotes a smaller (larger) runtime or smaller (larger) MIP gap after runtime on a particular instance against another solver. Ties occur when both approaches reach the 60-second runtime limit or when the MIP gap is (approximately) identical. The instance-wise comparison further includes the number of times each approach finds an optimal solution or primal feasible solution compared to another approach.

Figures 7.5.6–7.5.8 further illustrate the instance-wise comparison. These scatter plots highlight the runtime and MIP gap of the SAS and VBS compared to the DS and SBS on a per-instance basis. To interpret these figures effectively, referencing Tables 7.5.4–7.5.6 is recommended: e. g., when considering the runtime (MIP gap) of the VBS w. r. t. the DS or SBS in Figure 7.5.6, many points are located in the top right corner, implying that both runs timed out on several instances. While this is also signified by the enlarged marks, Table 7.5.4 provides a detailed answer: exactly 963 (138) data points are located in the top right corner of the runtime (MIP gap) plot. To enhance the clarity of the point distribution, a linear regression is included in each plot as the line with the best fit for the given data.

When it comes to runtime as the performance metric of interest (see Table 7.5.4 and Figure 7.5.6), the DS, SBS, and VBS exhibit mean runtimes of 35.47s, 32.93s, and 25.10s, respectively. When training with runtime as label, a mean runtime of 30.58s was achieved by the SAS. This closes the performance gap w. r. t. runtime between the DS (respectively, SBS) and VBS by approximately 47.2% (respectively, 30.0%). When we put these numbers into the perspective of Figure 7.5.1, an average runtime of 30.58s corresponds approximately to the virtual-373$^{\text{rd}}$ solver, which is in the top 7.3% of all virtual solvers. In other words, given that the DS corresponded approximately to the virtual-1515$^{\text{th}}$ solver, we close the gap between a virtual-solver that performs on average nearly identical to the DS and the VBS by approximately 75.4%.

Notably, both the VBS and SAS approach rely on many algorithms to tackle the 3095 instances (787 for the SAS and 1264 for the VBS). In particular, the SAS solves more instances to optimality than either the DS or SBS (1863 vs. 1626 or 1682). However, it must be noted that training with runtime as a label comes at a cost: the number of instance for which a feasible solution is found after runtime by the SAS is smaller than the same metric for the DS and the SBS (2514 vs. 2711 or 2762). Moreover, the average MIP gap is nearly 50% larger than that of the DS with a noticeable standard deviation. When considering the instance-wise comparison, the SAS beats the DS (SBS) in 47.9% (35.4%) and loses in only 13.0% (25.8%) of all cases w. r. t. runtime. However, the relatively weak

Table 7.5.4: Results obtained when using runtime as the training label.

		SAS		DS		SBS		VBS	
	# algorithms	787		1		2		1264	
	# feasible	2514		2711		2762		2957	
	# optimal	1863		1626		1682		2134	
μ	runtime & gap	30.58s	1.91	35.47s	1.30	32.93s	1.13	25.10s	0.48
σ		26.45s	3.85	26.10s	3.25	26.80s	3.06	26.23s	2.04
SAS	wins vs. w. r. t. runtime / gap	-	-	47.9%	16.4%	35.4%	12.6%	0.0%	0.0%
	ties vs. w. r. t. runtime / gap	-	-	39.1%	62.6%	38.8%	63.4%	31.3%	65.0%
	losses vs. w. r. t. runtime / gap	-	-	13.0%	20.9%	25.8%	24.0%	68.7%	35.0%
	# both optimal / primal feasible	-	-	1588	2426	1641	2451	1859	2514
	# only row optimal / primal feasible	-	-	271	88	218	63	0	0
	# only column optimal / primal feasible	-	-	25	285	38	311	273	443
	# neither optimal / primal feasible	-	-	1211	296	1198	270	963	138
DS	wins vs. w. r. t. runtime / gap	13.0%	20.9%	-	-	11.0%	9.6%	0.0%	0.0%
	ties vs. w. r. t. runtime / gap	39.1%	62.6%	-	-	44.4%	66.5%	31.1%	58.2%
	losses vs. w. r. t. runtime / gap	47.9%	16.4%	-	-	44.6%	23.9%	68.9%	41.8%
	# both optimal / primal feasible	1588	2426	-	-	1572	2684	1613	2711
	# only row optimal / primal feasible	25	285	-	-	41	27	0	0
	# only column optimal / primal feasible	271	88	-	-	107	78	519	246
	# neither optimal / primal feasible	1211	296	-	-	1375	306	963	138
SBS	wins vs. w. r. t. runtime / gap	25.8%	24.0%	44.6%	23.9%	-	-	0.0%	0.0%
	ties vs. w. r. t. runtime / gap	38.8%	63.4%	44.4%	66.5%	-	-	31.3%	60.5%
	losses vs. w. r. t. runtime / gap	35.4%	12.6%	11.0%	9.6%	-	-	68.7%	39.5%
	# both optimal / primal feasible	1641	2451	1572	2684	-	-	1679	2762
	# only row optimal / primal feasible	38	311	107	78	-	-	0	0
	# only column optimal / primal feasible	218	63	41	27	-	-	453	195
	# neither optimal / primal feasible	1198	270	1375	306	-	-	963	138
VBS	wins vs. w. r. t. runtime / gap	68.7%	35.0%	68.9%	41.8%	68.7%	39.5%	-	-
	ties vs. w. r. t. runtime / gap	31.3%	65.0%	31.1%	58.2%	31.3%	60.5%	-	-
	losses vs. w. r. t. runtime / gap	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	-	-
	# both optimal / primal feasible	1859	2514	1613	2711	1679	2762	-	-
	# only row optimal / primal feasible	273	443	519	246	453	195	-	-
	# only column optimal / primal feasible	0	0	0	0	0	0	-	-
	# neither optimal / primal feasible	963	138	963	138	963	138	-	-

performance w. r. t. MIP gap is also visible here: the SAS beats the DS (SBS) in only 16.4% (12.6%) and loses in 20.9% (24.0%) of all cases w. r. t. MIP gap.

When it comes to the MIP gap as the performance metric of interest (see Table 7.5.5 and Figure 7.5.7), the DS, SBS, and VBS exhibit mean MIP gaps of 1.30, 1.14, and 0.48, respectively. At first glance, the SBS w. r. t. MIP gap performs very similar to the SBS w. r. t. runtime (see also Table 7.5.4). However, when using the MIP gap as training label, the performance of the SAS is quite different with a mean MIP gap of 0.79 and mean runtime of 32.31s. This closes the performance gap w. r. t. MIP gap between the DS (respectively, SBS) and VBS by approximately 62.2% (respectively, 53.0%). Once

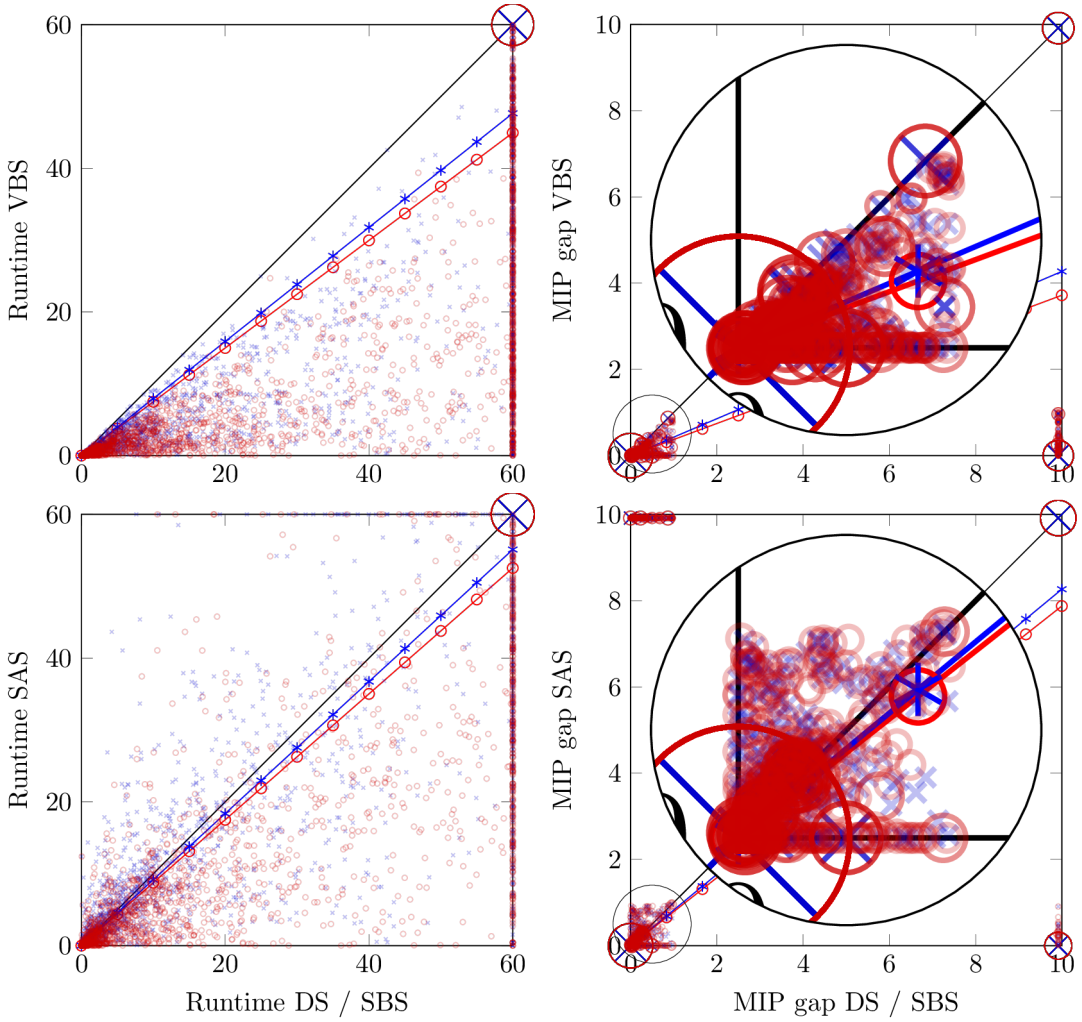

Figure 7.5.6: Instance-wise comparison when using runtime as the training labels. Red circles represent the default solver and blue crosses the single-best solver.

again, putting these numbers into the perspective of Figure 7.5.2, an average MIP gap of 0.79 corresponds approximately to the virtual-295$^{\text{th}}$ solver, which is in the top 5.8% of all virtual solvers. Therefore, given that the DS corresponded approximately to the virtual-1594$^{\text{th}}$ solver, we close the gap between a virtual solver that behaves on average nearly identical to the DS and the VBS by approximately 81.5%.

As was the case when using runtime as training label, the SAS relies on many algorithms to approach the 3095 instances. Similarly, the SAS also solves more instances to optimality than either the DS or SBS (1789 vs. 1626 or 1674). Unsurprisingly, when training for MIP gap, the number of feasible solutions found is also increased compared to the DS and SBS (2868 vs. 2711 and 2761). When considering the instance-wise comparison, the SAS

approach beats the DS (respectively, SBS) in 29.5% (respectively, 23.4%) and loses in only 9.2% (respectively, 14.4%) of all cases w. r. t. MIP gap. And in contrast to Table 7.5.4, Table 7.5.5 also shows that training for MIP gap does not cause the SAS to perform significantly worse than the DS or SBS w. r. t. runtime (32.31s vs. 35.47s and 33.46s). However, the losses w. r. t. runtime (21.5% and 32.3% vs. the DS and SBS) are more pronounced.

Table 7.5.5: Results obtained when using MIP gap as the training label.

		SAS		DS		SBS		VBS	
	# algorithms	815		1		5		1264	
	# feasible	2868		2711		2761		2957	
	# optimal	1789		1626		1674		2134	
μ	runtime & gap	32.31s	0.79	35.47s	1.30	33.46s	1.14	25.10s	0.48
σ		26.17s	2.57	26.10s	3.25	26.71s	3.06	26.23s	2.04
SAS	wins vs. w. r. t. runtime / gap	-	-	37.6%	29.5%	27.4%	23.4%	0.0%	0.0%
	ties vs. w. r. t. runtime / gap	-	-	41.0%	61.3%	40.3%	62.2%	31.3%	64.3%
	losses vs. w. r. t. runtime / gap	-	-	21.5%	9.2%	32.3%	14.4%	68.7%	35.7%
	# both optimal / primal feasible	-	-	1571	2698	1608	2743	1785	2868
	# only row optimal / primal feasible	-	-	214	170	177	125	0	0
	# only column optimal / primal feasible	-	-	42	13	66	18	347	89
	# neither optimal / primal feasible	-	-	1268	214	1244	209	963	138
DS	wins vs. w. r. t. runtime / gap	21.5%	9.2%	-	-	15.6%	12.2%	0.0%	0.0%
	ties vs. w. r. t. runtime / gap	41.0%	61.3%	-	-	44.0%	63.5%	31.1%	58.2%
	losses vs. w. r. t. runtime / gap	37.6%	29.5%	-	-	40.4%	24.3%	68.9%	41.8%
	# both optimal / primal feasible	1571	2698	-	-	1554	2672	1613	2711
	# only row optimal / primal feasible	42	13	-	-	59	39	0	0
	# only column optimal / primal feasible	214	170	-	-	120	89	519	246
	# neither optimal / primal feasible	1268	214	-	-	1362	295	963	138
SBS	wins vs. w. r. t. runtime / gap	32.3%	14.4%	40.4%	24.3%	-	-	0.0%	0.0%
	ties vs. w. r. t. runtime / gap	40.3%	62.2%	44.0%	63.5%	-	-	31.2%	60.0%
	losses vs. w. r. t. runtime / gap	27.4%	23.4%	15.6%	12.2%	-	-	68.8%	40.0%
	# both optimal / primal feasible	1608	2743	1554	2672	-	-	1674	2761
	# only row optimal / primal feasible	66	18	120	89	-	-	0	0
	# only column optimal / primal feasible	177	125	59	39	-	-	458	196
	# neither optimal / primal feasible	1244	209	1362	295	-	-	963	138
VBS	wins vs. w. r. t. runtime / gap	68.7%	35.7%	68.9%	41.8%	68.8%	40.0%	-	-
	ties vs. w. r. t. runtime / gap	31.3%	64.3%	31.1%	58.2%	31.2%	60.0%	-	-
	losses vs. w. r. t. runtime / gap	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	-	-
	# both optimal / primal feasible	1785	2868	1613	2711	1674	2761	-	-
	# only row optimal / primal feasible	347	89	519	246	458	196	-	-
	# only column optimal / primal feasible	0	0	0	0	0	0	-	-
	# neither optimal / primal feasible	963	138	963	138	963	138	-	-

The accumulated evidence strongly supports the effectiveness of the SAS in line with our expectations. Training for runtime (MIP gap) results in the SAS exhibiting a smaller mean runtime (MIP gap) compared to the DS or SBS, thereby closing the default-VBS

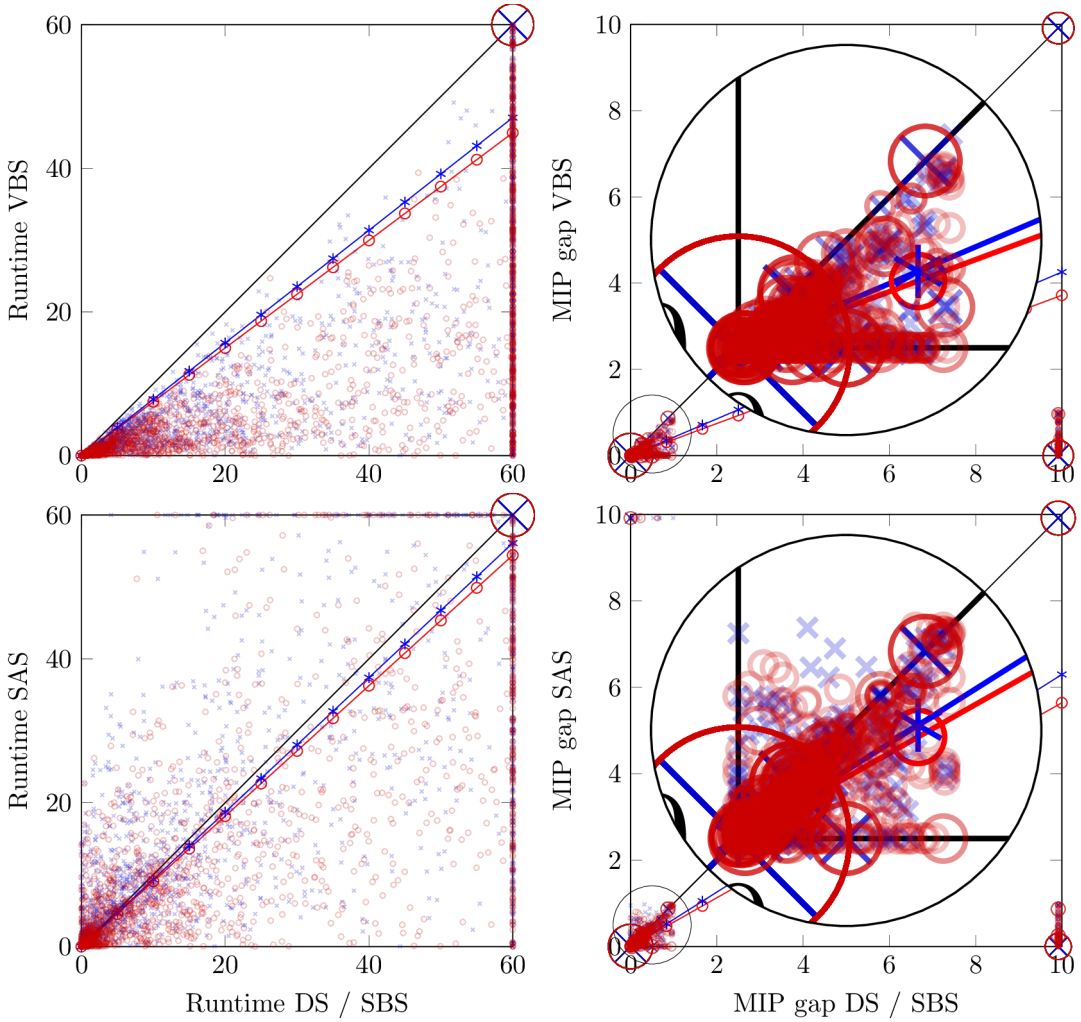

Figure 7.5.7: Instance-wise comparison when using MIP gap as the training label. Red circles represent the default solver and blue crosses the single-best solver.

performance gap by 47.2% (62.2%). Training for MIP gap not only led to the SAS performing favorably to the DS or SBS in terms of MIP gap but also in terms of runtime. However, the reverse did not hold when training for runtime. For practitioners who cannot express a clear preference for either runtime or MIP gap as the primary objective, our study suggests that using the MIP gap as a training label may be preferable. It is worth noting that this conclusion stands in contrast to the literature, where MIP is often treated as a decision problem using PAR-10 scoring on runtime, while completely neglecting the effects of the MIP gap after a runtime limit (see Section 7.2 and references therein).

The SAS, trained with a weighted sum as the label (Table 7.5.6 and Figure 7.5.8), closely mirrors mean runtimes and MIP gaps compared to explicit training for individual

objectives (30.6s, 0.77 vs. 30.58s, 0.79). Notably, it outperforms explicit training for MIP gap by finding more feasible solutions (2875 vs. 2868, Table 7.5.5) and matches the number of optimal solutions found in training for runtime (1862 vs. 1863, Table 7.5.4). Moreover, when compared to the DS (SBS), it only loses concerning runtime and MIP gap in 12.9% and 11.7% (25.6% and 15.5%) of all cases, respectively. In comparison to training with distinct labels (see Tables 7.5.4–7.5.5), the losses concerning the metric outside the used training label were nearly twice as large. In summary, this straightforward approach achieves the most balanced performance among the three tested training labels.

Table 7.5.6: Results obtained when using the weighted sum of runtime and MIP gap as the training label.

		SAS		DS		SBS		VBS	
	# algorithms	848		1		3		1264	
	# feasible	2875		2711		2767		2957	
	# optimal	1862		1626		1678		2134	
μ	runtime & gap	30.60s	0.77	35.47s	1.30	33.05s	1.12	25.10s	0.48
σ		26.48s	2.54	26.10s	3.25	26.79s	3.03	26.23s	2.04
SAS	wins vs. w. r. t. runtime / gap	-	-	47.9%	27.2%	35.7%	21.8%	0.0%	0.0%
	ties vs. w. r. t. runtime / gap	-	-	39.2%	61.2%	38.7%	62.6%	31.3%	66.1%
	losses vs. w. r. t. runtime / gap	-	-	12.9%	11.7%	25.6%	15.5%	68.7%	33.9%
	# both optimal / primal feasible	-	-	1592	2702	1639	2753	1860	2875
	# only row optimal / primal feasible	-	-	268	173	221	122	0	0
	# only column optimal / primal feasible	-	-	21	9	38	14	272	82
	# neither optimal / primal feasible	-	-	1214	211	1197	206	963	138
DS	wins vs. w. r. t. runtime / gap	12.9%	11.7%	-	-	11.4%	9.1%	0.0%	0.0%
	ties vs. w. r. t. runtime / gap	39.2%	61.2%	-	-	44.4%	66.6%	31.1%	58.2%
	losses vs. w. r. t. runtime / gap	47.9%	27.2%	-	-	44.2%	24.3%	68.9%	41.8%
	# both optimal / primal feasible	1592	2702	-	-	1569	2688	1613	2711
	# only row optimal / primal feasible	21	9	-	-	44	23	0	0
	# only column optimal / primal feasible	268	173	-	-	108	79	519	246
	# neither optimal / primal feasible	1214	211	-	-	1374	305	963	138
SBS	wins vs. w. r. t. runtime / gap	25.6%	15.5%	44.2%	24.3%	-	-	0.0%	0.0%
	ties vs. w. r. t. runtime / gap	38.7%	62.6%	44.4%	66.6%	-	-	31.4%	60.3%
	losses vs. w. r. t. runtime / gap	35.7%	21.8%	11.4%	9.1%	-	-	68.6%	39.7%
	# both optimal / primal feasible	1639	2753	1569	2688	-	-	1677	2767
	# only row optimal / primal feasible	38	14	108	79	-	-	0	0
	# only column optimal / primal feasible	221	122	44	23	-	-	455	190
	# neither optimal / primal feasible	1197	206	1374	305	-	-	963	138
VBS	wins vs. w. r. t. runtime / gap	68.7%	33.9%	68.9%	41.8%	68.6%	39.7%	-	-
	ties vs. w. r. t. runtime / gap	31.3%	66.1%	31.1%	58.2%	31.4%	60.3%	-	-
	losses vs. w. r. t. runtime / gap	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	-	-
	# both optimal / primal feasible	1860	2875	1613	2711	1677	2767	-	-
	# only row optimal / primal feasible	272	82	519	246	455	190	-	-
	# only column optimal / primal feasible	0	0	0	0	0	0	-	-
	# neither optimal / primal feasible	963	138	963	138	963	138	-	-

To underline the seemingly superior performance of using the weighted sum as a training

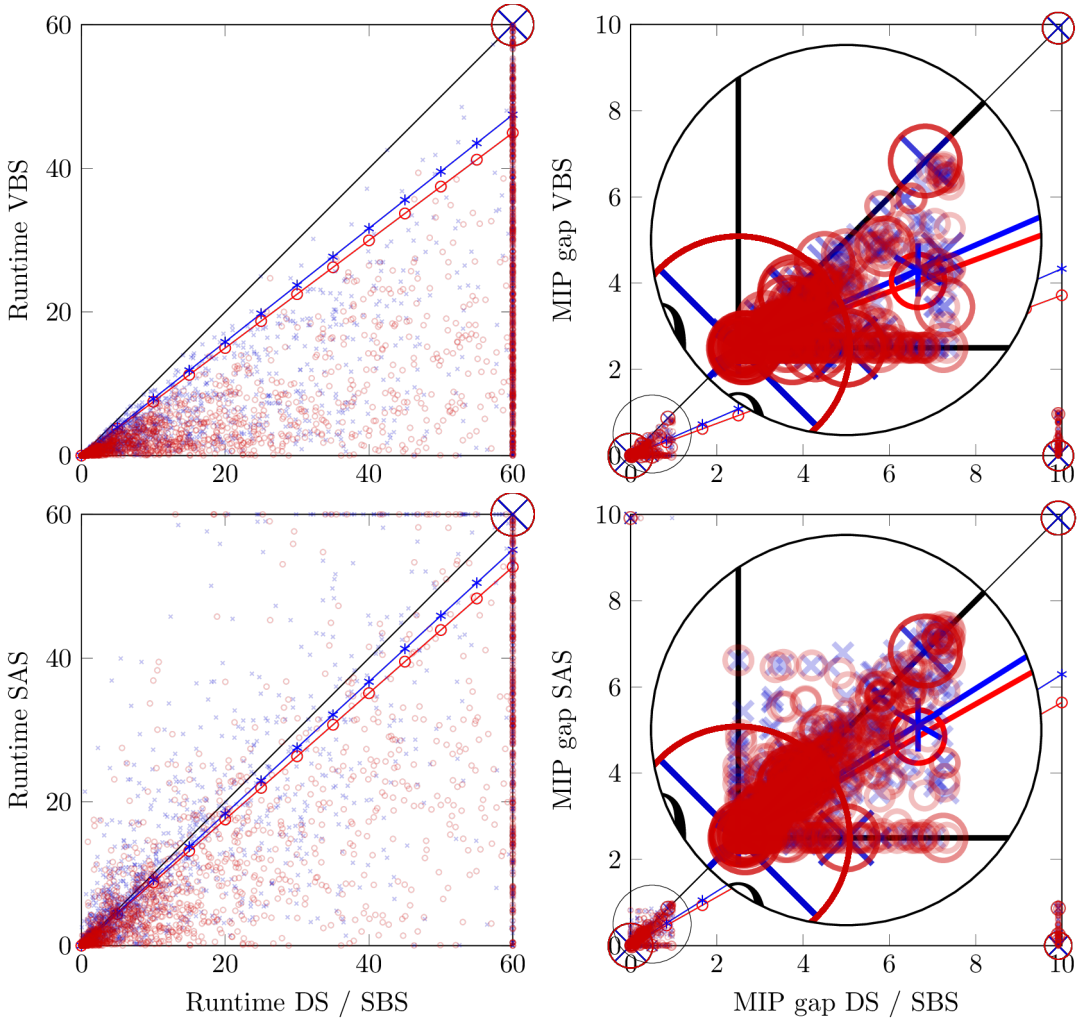

Figure 7.5.8: Instance-wise comparison when using the weighted sum of runtime and MIP gap as the training label. Red circles represent the default solver and blue crosses the single-best solver.

label, a series of one-sided Wilcoxon signed-rank tests were performed to determine the statistical significance of our results (Wilcoxon, 1945). The following hypotheses were used:

- $H_{0,t}(x)$: The distribution underlying the pairwise instance runtime differences between the x solver minus the SAS is stochastically less or equal to a distribution symmetric about zero.
- $H_{1,t}(x)$: The distribution underlying the pairwise instance runtime difference between the x solver minus the SAS is stochastically greater than a distribution symmetric about zero.

The corresponding hypotheses $H_{0,g}(x)$ and $H_{1,g}(x)$ were used concerning MIP gap instead of runtime. Given a 5% confidence level the following holds for both the DS and SBS (as arguments x). When using the runtime as training label, we can reject $H_{0,t}$ but cannot reject $H_{0,g}$. Conversely, when using the MIP gap as training label, we can reject $H_{0,g}$ but fail to reject $H_{0,t}$. Only when training for the weighted sum of runtime and gap are we able to reject both $H_{0,t}$ and $H_{0,g}$.

7.6 Conclusion

In this paper, the problem of selecting a configuration for a MIP solver on a per-instance basis was studied from the perspective of the ASP. The study began in Section 7.2 with a literature overview that discussed existing approaches for enhancing the performance of MIP solvers. This concerned the ACP and ASP as static approaches and further, dynamic approaches. In Section 7.3, a concise problem definition of the ASP was provided, following the foundational work of Rice (1976). The standard approach in the literature is to learn a regressor for each algorithm or to perform a binary classification for all pairs of algorithms (Kerschke et al., 2019); however, this approach becomes impractical with a large number of algorithms. Therefore, in Section 7.4 a methodology that follows the general idea of EML (Lombardi et al., 2017) was proposed. In a nutshell, EML employs ML to derive the components of a prescriptive optimization problem that cannot intuitively be established otherwise. In more detail, given historical performance data, a RF regressor was learnt (offline) as an approximation of a solver's performance that depends on its algorithmic configuration and instance features. The learnt RF model was integrated into a prescriptive optimization model, and a tailored B&B procedure allowed for the efficient computation of the (predictably) optimal configuration of a solver for a given instance. The methodology was evaluated in Section 7.5, using the MIPLIB benchmark and a state-of-the-art solver. Noteworthy, the MIPLIB attempts to reflect the richness of challenges in various OR domains. By using heterogenous instances and considering thousands of algorithms, the study differs from related works in the context of the ASP. The experiments provided strong evidence that the proposed methodology can significantly narrow the performance gap between the default solver and the virtual-best solver in terms of average runtime and MIP gap. To be more precise, we were able to close the performance gap between the default and virtual-best solver by about half w. r. t. average runtime and MIP gap; and this corresponds approximately to the performance of a virtual solver that is located in the top ten percentile of all solvers. Concerns were raised regarding the use of runtime as a training label, as commonly done in the literature, particularly due to the impact on the number of feasible solutions and MIP gap. The experiments suggested that

a weighted training label, considering both runtime and MIP gap, is preferable.

We believe that this paper provides not only a compelling argument for the benefits of studying the ASP in the context of MIP, but also to the more general, continued coalescence of predictive and prescriptive analytics on a methodological level. Despite the success demonstrated in this proof-of-concept, there remain open research questions that extend beyond the scope of this paper. One avenue for further exploration is the search for alternative features that are broadly applicable to MIP. Moreover, the current framework only makes a static choice. Exploring a more dynamic variant that incorporates feedback during the algorithm's execution could lead to superior performance. Finally, we believe that a larger community effort is required when it comes to nurturing and curating an even larger and more diverse dataset. We hope to explore some of these options in the future.

7.7 References

Accorsi, Luca, Andrea Lodi, and Daniele Vigo (2022). "Guidelines for the computational testing of machine learning approaches to vehicle routing problems". In: *Operations Research Letters* 50.2, pp. 229–234.

Adenso-Díaz, Belarmino and Manuel Laguna (2006). "Fine-Tuning of Algorithms Using Fractional Experimental Designs and Local Search". In: *Operations Research* 54.1, pp. 99–114.

Ansótegui, Carlos, Meinolf Sellmann, and Kevin Tierney (2009). "A Gender-Based Genetic Algorithm for the Automatic Configuration of Algorithms". In: *Principles and Practice of Constraint Programming*. Lecture Notes in Computer Science 5732. Berlin, Heidelberg: Springer, pp. 142–157.

Ansótegui, Carlos et al. (2015). "Model-Based Genetic Algorithms for Algorithm Configuration". In: *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pp. 733–739.

Auer, Peter, Nicolò Cesa-Bianchi, and Paul Fischer (2002). "Finite-time Analysis of the Multiarmed Bandit Problem". In: *Machine Learning* 47, pp. 235–256.

Bartz-Beielstein, Thomas, Martin Zaefferer, and Frederik Rehbach (2021). "In a Nutshell – The Sequential Parameter Optimization Toolbox". In: *arXiv:1712.04076 [cs.MS]*.

Bengio, Yoshua, Andrea Lodi, and Antoine Prouvost (2021). "Machine learning for combinatorial optimization: A methodological tour d'horizon". In: *European Journal of Operational Research* 290.2, pp. 405–421.

Berthold, Timo, Matteo Francobaldi, and Gregor Hendel (2022). "Learning to Use Local Cuts". In: *arXiv:2206.11618 [math.OC]*.

Berthold, Timo and Gregor Hendel (2021). "Learning To Scale Mixed-Integer Programs". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.5, pp. 3661–3668.

Bertsimas, Dimitris and Jack Dunn (2017). "Optimal classification trees". In: *Machine Learning* 106.7, pp. 1039–1082.

Birattari, Mauro (2009). *Tuning metaheuristics: a machine learning perspective*. 2nd ed. Studies in Computational Intelligence Vol. 197. Berlin: Springer.

Birattari, Mauro et al. (2010). "F-Race and Iterated F-Race: An Overview". In: *Experimental Methods for the Analysis of Optimization Algorithms*. Ed. by Thomas Bartz-Beielstein et al. Berlin, Heidelberg: Springer, pp. 311–336.

Bischl, Bernd et al. (2016). "ASlib: A benchmark library for algorithm selection". In: *Artificial Intelligence* 237, pp. 41–58.

Bixby, Robert E., Andrew E. Boyd, and Ronni R. Indovina (1992). "MIPLIB: a test set of mixed integer programming problems". In: *SIAM News* 25.2, p. 16.

Breiman, Leo (1996). "Bagging predictors". In: *Machine Learning* 24.2, pp. 123–140.

Breiman, Leo (2001). "Random Forests". In: *Machine Learning* 45.1, pp. 5–32.

Breiman, Leo et al. (1998). *Classification and regression trees*. 1st ed. Boca Raton: Chapman & Hall/CRC.

Chmiela, Antonia et al. (2021). "Learning to Schedule Heuristics in Branch and Bound". In: *Advances in Neural Information Processing Systems*. Vol. 34, pp. 24235–24246.

Eggensperger, Katharina, Marius Lindauer, and Frank Hutter (2019). "Pitfalls and Best Practices in Algorithm Configuration". In: *Journal of Artificial Intelligence Research* 64, pp. 861–893.

Fair Isaac Corporation (2023). *FICO Xpress Optimizer Reference Manual*.

Firat, Murat et al. (2020). "Column generation based heuristic for learning classification trees". In: *Computers & Operations Research* 116, p. 104866.

Gittins, J. C. (1979). "Bandit Processes and Dynamic Allocation Indices". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 41.2, pp. 148–164.

Gleixner, Ambros et al. (2021). "MIPLIB 2017: data-driven compilation of the 6th mixed-integer programming library". In: *Mathematical Programming Computation* 13, pp. 443–490.

Gupta, Prateek et al. (2020). "Hybrid Models for Learning to Branch". In: *Advances in Neural Information Processing Systems*. Vol. 33, pp. 18087–18097.

Gurobi Optimization, LLC (2023). *Gurobi Optimizer Reference Manual*.

Hastie, Trevor, Robert Tibshirani, and J. H. Friedman (2009). *The elements of statistical learning: data mining, inference, and prediction*. 2nd ed. Springer series in statistics. New York: Springer.

Hinkelmann, Klaus and Oscar Kempthorne (1994). *Design and analysis of experiments*. Rev. ed. Wiley series in probability and mathematical statistics. New York: Wiley.

Hoos, Holger H. (2011). "Automated Algorithm Configuration and Parameter Tuning". In: *Autonomous Search*. Ed. by Youssef Hamadi, Eric Monfroy, and Frédéric Saubion. Berlin, Heidelberg: Springer, pp. 37–71.

Hoos, Holger H., Frank Hutter, and Kevin Leyton-Brown (2021). "Automated Configuration and Selection of SAT Solvers". In: *Handbook of Satisfiability*. Ed. by Armin Biere et al. Frontiers in Artificial Intelligence and Applications 336. IOS Press, pp. 481–507.

Hutter, Frank, Holger H. Hoos, and Kevin Leyton-Brown (2010). "Automated Configuration of Mixed Integer Programming Solvers". In: *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. Lecture Notes in Computer Science 6140. Berlin, Heidelberg: Springer, pp. 186–202.

Hutter, Frank, Holger H. Hoos, and Kevin Leyton-Brown (2011). "Sequential Model-Based Optimization for General Algorithm Configuration". In: *Learning and Intelligent Optimization*. Lecture Notes in Computer Science 6683. Berlin, Heidelberg: Springer, pp. 507–523.

Hutter, Frank, Holger H. Hoos, and Thomas Stützle (2007). "Automatic Algorithm Configuration Based on Local Search". In: *Proceedings of the 22nd national conference on Artificial intelligence*. Vol. 2. AAAI Press, pp. 1152–1157.

Hutter, Frank et al. (2006). "Performance Prediction and Automated Tuning of Randomized and Parametric Algorithms". In: *Principles and Practice of Constraint Programming*. Lecture Notes in Computer Science 4204. Berlin, Heidelberg: Springer, pp. 213–228.

Hutter, Frank et al. (2009). "ParamILS: An Automatic Algorithm Configuration Framework". In: *Journal of Artificial Intelligence Research* 36, pp. 267–306.

Iommazzo, Gabriele (2021). "Algorithmic configuration by learning and optimization". Ph.D. thesis. Institut polytechnique de Paris.

Iommazzo, Gabriele et al. (2020a). "A Learning-Based Mathematical Programming Formulation for the Automatic Configuration of Optimization Solvers". In: *Machine Learning, Optimization, and Data Science*. Lecture Notes in Computer Science 12565. Cham: Springer, pp. 700–712.

Iommazzo, Gabriele et al. (2020b). "Learning to Configure Mathematical Programming Solvers by Mathematical Programming". In: *Learning and Intelligent Optimization*. Lecture Notes in Computer Science 12096. Cham: Springer, pp. 377–389.

Jones, Donald R. (2001). "A Taxonomy of Global Optimization Methods Based on Response Surfaces". In: *Journal of Global Optimization* 21.4, pp. 345–383.

Jones, Donald R., Matthias Schonlau, and William J. Welch (1998). "Efficient Global Optimization of Expensive Black-Box Functions". In: *Journal of Global Optimization* 13.4, pp. 455–493.

Kadioglu, Serdar et al. (2010). "ISAC – Instance-Specific Algorithm Configuration". In: *Proceedings of the 19th European Conference on Artificial Intelligence*. IOS Press, pp. 751–756.

Kerschke, Pascal and Heike Trautmann (2019). "Automated Algorithm Selection on Continuous Black-Box Problems by Combining Exploratory Landscape Analysis and Machine Learning". In: *Evolutionary Computation* 27.1, pp. 99–127.

Kerschke, Pascal et al. (2018). "Leveraging TSP Solver Complementarity through Machine Learning". In: *Evolutionary Computation* 26.4, pp. 597–620.

Kerschke, Pascal et al. (2019). "Automated Algorithm Selection: Survey and Perspectives". In: *Evolutionary Computation* 27.1, pp. 3–45.

Khalil, Elias B. et al. (2016). "Learning to branch in mixed integer programming". In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. Vol. 30. AAAI Press, pp. 724–731.

Khalil, Elias B. et al. (2017). "Learning to Run Heuristics in Tree Search". In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pp. 659–666.

Koch, Thorsten et al. (2022). "Progress in mathematical programming solvers from 2001 to 2020". In: *EURO Journal on Computational Optimization* 10, p. 100031.

Kotthoff, Lars (2016). "Algorithm Selection for Combinatorial Search Problems: A Survey". In: *Data Mining and Constraint Programming*. Lecture Notes in Computer Science 10101. Cham: Springer, pp. 149–190.

Kotthoff, Lars et al. (2015). "Improving the State of the Art in Inexact TSP Solving Using Per-Instance Algorithm Selection". In: *Learning and Intelligent Optimization*. Lecture Notes in Computer Science 8994. Cham: Springer, pp. 202–217.

Land, A. H. and A. G. Doig (1960). "An Automatic Method of Solving Discrete Programming Problems". In: *Econometrica* 28.3, pp. 497–520.

Laporte, Gilbert and Paolo Toth (2022). "A gap in scientific reporting". In: *4OR* 20, pp. 169–171.

Leyton-Brown, Kevin, Eugene Nudelman, and Yoav Shoham (2002). "Learning the Empirical Hardness of Optimization Problems: The Case of Combinatorial Auctions". In: *Principles and Practice of Constraint Programming*. Lecture Notes in Computer Science 2470. Berlin, Heidelberg: Springer, pp. 556–572.

Leyton-Brown, Kevin et al. (2003). "A Portfolio Approach to Algorithm Selection". In: *Proceedings of the 18th international joint conference on Artificial intelligence*. Morgan Kaufmann Publishers Inc., pp. 1542–1543.

Lodi, Andrea (2013). "The Heuristic (Dark) Side of MIP Solvers". In: *Hybrid Metaheuristics*. Ed. by El-Ghazali Talbi. Studies in Computational Intelligence 434. Berlin, Heidelberg: Springer, pp. 273–284.

Lodi, Andrea and Andrea Tramontani (2014). "Performance Variability in Mixed-Integer Programming". In: *Theory Driven by Influential Applications*. Ed. by J. Cole Smith. Theory Driven by Influential Applications, pp. 1–12.

Lodi, Andrea and Giulia Zarpellon (2017). "On learning and branching: a survey". In: *TOP* 25.2, pp. 207–236.

Lombardi, Michele, Michela Milano, and Andrea Bartolini (2017). "Empirical decision model learning". In: *Artificial Intelligence* 244, pp. 343–367.

López-Ibáñez, Manuel et al. (2016). "The irace package: Iterated racing for automatic algorithm configuration". In: *Operations Research Perspectives* 3, pp. 43–58.

Loreggia, Andrea et al. (2016). "Deep learning for algorithm portfolios". In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. Vol. 30. AAAI Press, pp. 1280–1286.

Maron, Oden and Andrew W. Moore (1997). "The Racing Algorithm: Model Selection for Lazy Learners". In: *Artificial Intelligence Review* 11.1, pp. 193–225.

Mitchell, Tom M. (1997). *Machine Learning*. McGraw-Hill series in computer science. New York: McGraw-Hill.

Mittelmann, Hans D. (2020). "Benchmarking Optimization Software - a (Hi)Story". In: *SN Operations Research Forum* 1.2, pp. 1–6.

Mockus, J., V. Tiesis, and A. Zilinskas (1978). "The application of Bayesian methods for seeking the extremum". In: *Towards Global Optimization* 2, pp. 117–129.

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Petropoulos, Fotios et al. (2023). "Operational Research: Methods and Applications". In: *arXiv:2303.14217 [math.OC]*.

Probst, Philipp, Anne-Laure Boulesteix, and Bernd Bischl (2019). "Tunability: Importance of Hyperparameters of Machine Learning Algorithms". In: *Journal of Machine Learning Research* 20.1, pp. 1934–1965.

Pushak, Yasha and Holger H. Hoos (2018). "Algorithm Configuration Landscapes: More Benign Than Expected?" In: *Parallel Problem Solving from Nature*. Lecture Notes in Computer Science 11102. Cham: Springer, pp. 271–283.

Rice, John R. (1976). "The Algorithm Selection Problem". In: *Advances in Computers*. Vol. 15. Elsevier, pp. 65–118.

Russell, Stuart J., Peter Norvig, and Ernest Davis (2010). *Artificial intelligence: a modern approach*. 3rd ed. Prentice Hall series in artificial intelligence. Upper Saddle River: Prentice Hall.

Schede, Elias et al. (2022). "A Survey of Methods for Automated Algorithm Configuration". In: *Journal of Artificial Intelligence Research* 75, pp. 425–487.

Stone, M. (1974). "Cross-Validatory Choice and Assessment of Statistical Predictions". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 36.2, pp. 111–147.

Sutton, Richard S. and Andrew G. Barto (2018). *Reinforcement learning: an introduction*. 2nd ed. Adaptive computation and machine learning series. Cambridge, The MIT Press.

Tang, Yunhao, Shipra Agrawal, and Yuri Faenza (2020). "Reinforcement Learning for Integer Programming: Learning to Cut". In: *Proceedings of the 37th International Conference on Machine Learning*, pp. 9367–9376.

Wilcoxon, Frank (1945). "Individual comparisons by ranking methods". In: *Biometrics Bulletin* 1.6, pp. 80–83.

Xu, Lin et al. (2008). "SATzilla: Portfolio-based Algorithm Selection for SAT". In: *Journal of Artificial Intelligence Research* 32, pp. 565–606.

Xu, Lin et al. (2011). "Hydra-MIP: Automated algorithm configuration and selection for mixed integer programming". In: *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*. AAAI Press, pp. 16–30.

Zarpellon, Giulia et al. (2021). "Parameterizing Branch-and-Bound Search Trees to Learn Branching Policies". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35(5). AAAI Press, pp. 3931–3939.

8 Summary, Conclusion, and Outlook

This thesis addresses several challenges associated with introducing autonomous auxiliary vehicles, e. g., drones or robots, into a logistics system from the perspective of *Operations Research* (OR). To this end, optimization models are formalized that enable the assessment of potential benefits of integrating such vehicles into last-mile delivery. As the resulting models are computationally challenging, the thesis continuously refines the formulations and develops appropriate algorithms that are capable of producing high-quality solutions with reasonable computational effort. This facilitates effective problem-solving and aids in shaping the design of future delivery fleets, laying the groundwork for future decision support systems. In OR, *Mixed-Integer Programming* (MIP) solvers play a pivotal role. As is evident by this thesis, this concerns solving *Mixed-Integer Linear Program* (MILP) formulations directly or integrating parts of them in matheuristic frameworks. This raises the fundamental question if the performance of such solvers can be enhanced in any meaningful way, by adjusting their default algorithmic behavior on a per-instance basis. We trace the roots of this problem to the *Algorithm Selection Problem* (ASP) and develop a novel methodology that leverages *Machine Learning* (ML) to formalize a prescriptive optimization problem. By using a tailored *Branch-and-Bound* (B&B) approach, this methodology enables us to effectively compute the (predictably) optimal configuration of a MIP solver on a per-instance basis with minimal computational overhead. The potential impact extends beyond specific problem domains, fostering a more comprehensive and synergistic approach to decision-making and optimization.

This chapter summarizes the contents of this thesis and lists its main contributions in Section 8.1. The conclusion of this thesis includes an outlook concerning possibilities for future research in Section 8.2.

8.1 Summary and Conclusion

In Chapter 2 we introduce the *Vehicle Routing Problem with Drones and En Route Operations* (VRPDERO) (Schermer et al., 2019a). This problem involves the collaborative operation of trucks and drones in last-mile delivery. The distinguishing feature of the VRPDERO is that it allows drones to be launched and retrieved by the truck not only at customer locations but also en route. We formulate the VRPDERO as a MILP; in

addition, several effective *Valid Inequalities* (VIEQs) are proposed. The computational experiments affirm the effectiveness of the VIEQ; however, using a commercial solver only some small-sized problems can be solved optimally within reasonable time. To address this limitation, a heuristic is introduced that leverages the concepts of Variable Neighborhood Search and Tabu Search. This heuristic incorporates a drone insertion procedure as a local search operator, which is embedded within a scalable framework. An extensive computational study is presented, investigating the impact of various drone design parameters on, e. g., the makespan and algorithmic runtime.

This work contributes by formally exploring en route launch and retrieval, an aspect scarcely investigated in the literature. The numerical results demonstrate the potential advantages of reducing the makespan and enhancing drone utilization. Particularly, en route operations prove most beneficial when either the drone's endurance is small, allowing for operations that might otherwise be infeasible, or when the drone's relative velocity is sufficiently high, enabling more efficient execution of multiple operations. Despite these benefits, it remains crucial to acknowledge that introducing en route operations substantially increases the computational difficulty of the problem, especially when using a commercial solver. Nonetheless, the proposed heuristic was able to provide high-quality solutions within short computation time.

In Chapter 3 our focus shifts to the *Vehicle Routing Problem with Drones* (VRPD) (Schermer et al., 2019b). The chapter begins with an overview of related problems that feature trucks and drones in last-mile delivery, highlighting their intricate synchronization requirements. Following that, we provide an improved MILP formulation of the VRPD and derive several VIEQs for that model. Noteworthy is the introduction of the matheuristic, designed to efficiently exploit the structure of the VRPD. At the core of the matheuristic lies the *Drone Assignment and Scheduling Problem* (DASP). Given a feasible routing, the DASP aims to determine an optimal assignment and schedule of drones such that the makespan is minimized. In this context, we provide two MILP formulations for the DASP, demonstrating that these formulations can be adjusted to match the most common assumptions found in the literature. The chapter concludes by presenting the outcomes and insights gained from extensive computational experiments.

The formal model introduced in this work refines the truck-drone interactions in several ways, e. g., by incorporating round trip operations. The numerical results underscore the substantial potential for makespan reduction, achievable by using drones. Moreover, the results demonstrate that comparable makespan reductions can be achieved with various drone designs. Despite employing a simple route generation procedure, in terms of computational efficiency, the matheuristic is fast and able to consistently provide optimal or near optimal solutions. This underscores the relevance of the DASP as heuristic component for

solving such problems. In comparison to *Best Known Solutions* (BKSs), the matheuristic performs favorably, reliably finding solutions that are within a 0.5% margin to the BKSs. Notably, the model and matheuristic also contributed to the discovery of several new and improved BKSs. Overall, the matheuristic can be seen as a natural evolution of the drone insertion procedure proposed in Chapter 2. As demonstrated, this approach is scalable and easily adaptable to the most common assumptions. Consequently, this work also serves as an attempt to unify the divergent streams in the fast-growing literature.

In Chapter 4, the focus is on introducing the *Traveling Salesman Drone Station Location Problem* (TSDSLP) (Schermer et al., 2020c). This chapter provides a different perspective on last-mile logistics where, instead of directly pairing trucks with auxiliary vehicles, the latter are located in micro-depots. Two MILP formulations for the problem are presented: one is compact and the other one features an exponential number of constraints. To address the latter, a dynamic separation procedure is proposed. An extensive numerical study is conducted, revealing that suitable auxiliary vehicles, strategically positioned in micro-depots, can yield a significant reduction in the delivery time.

In contrast to the computational challenges posed by the VRPD, the study demonstrates that the TSDSLP is less challenging, due to the reduced need for synchronization between different vehicles. Despite this, significant savings w. r. t. delivery time are possible, especially if the number of stationed robots is sufficiently large. Notably, this holds true even for slow-moving auxiliary vehicles (e. g., wheeled robots). This finding is highly relevant, as such vehicles are generally less beneficial in related problems such as the *Traveling Salesman Problem with Drone* (TSPD) or VRPD.

Chapter 5 introduces the *Drone-Assisted Traveling Salesman Problem with Robot Stations* (TSPDRS) (Schermer et al., 2020b), merging the previous concepts of the TSPD with the TSDSLP. The problem is formulated as a MILP that substantially improves upon previous formulations. The chapter presents the numerical results of our computational study, showcasing that a state-of-the-art solver yields acceptable results within reasonable runtime for small-sized instances. Moreover, the results indicate that integrating a truck-drone tandem with properly situated robot stations can bring significant reductions in the delivery time or operational costs.

The prevailing objective in the related scientific literature typically revolves around minimizing the makespan. The study demonstrates a crucial relationship where optimizing the makespan aligns closely with reduced costs. In contrast, optimizing the operational costs may increase the makespan substantially. Importantly, our experiments indicate that the makespan objective is far more challenging from a computational perspective. Furthermore, the findings highlight that the operational utilization of vehicles varies based on the chosen objective function. Specifically, when the goal is to minimize costs, drone

round trips prove to be a suitable mode of operation. However, it is important to note that such operations may lead to a significant increase in mission time. This emphasizes the trade-offs and considerations involved in selecting the appropriate objective function to align with specific logistics goals and constraints.

Chapter 6 is dedicated to investigating the TSPD (Schermer et al., 2020a). The chapter begins with a review of existing MILP formulations and other exact methods that are used for solving variants of the problem. Given the state of research, it becomes evident that only small-sized instances can be solved to proven optimality. In response to this challenge, we propose two new and improved MILP formulations. Still, these formulations have one major disadvantage: numerous big-M constraints are required to accurately model the complex interactions between the truck and drone. To address this limitation, motivated by this disadvantage and corroborated by the findings of previous chapters, a third formulation for the TSPD is introduced that features an exponential number of constraints. By taking into account the possible modes of operation, this formulation cleverly decomposes the temporal relationships of the TSPD into separate components. Consequently, the model eliminates the need for big-M constraints entirely. To manage the exponential number of constraints, a dynamic separation procedure through a *Branch-and-Cut* (B&C) algorithm is proposed. An extensive computational study is conducted, analyzing the quality of the different formulations in finding optimal TSPD solutions across various benchmark instances from the literature.

The study contributes by offering additional insights into how the formulations perform in solving various benchmark instances from the literature and understanding the factors that influence their performance. Notably, the B&C approach showed strong performance across the different instances. Indeed, using this approach, small instances with 9 or 10 customers can be solved in just a few seconds and several larger instances with 19 or 20 customers within an hour. Consequently, this B&C algorithm challenges the existing state-of-the-art in solving the TSPD and allowed for the identification of several further, previously unknown optimal solutions. This advancement stands in stark contrast to the earliest work of Murray and Chu (2015), where even small-sized instances with only 10 customers could not be solved optimally within 30 minutes. Thereby, this study underlines the potentials of refining existing formulations through adept problem modeling.

In Chapter 7 the problem of selecting a configuration for a MIP solver on a per-instance basis is studied from the perspective of the ASP (Schermer, 2023). This study begins with a literature overview that discusses existing approaches for enhancing the performance of MIP solvers. Following the foundational work of Rice (1976), a concise problem definition of the ASP is given. Generally, the standard approach in the literature is to learn a regressor for each algorithm or to perform a binary classification for all pairs of algorithms;

however, this approach is not scalable and is therefore only suitable in presence of a handful of algorithms. Therefore, a novel methodology is proposed that follows the general idea of *Empirical Model Learning* (EML) (Lombardi et al., 2017). In a nutshell, EML employs ML to derive the components of a prescriptive optimization problem that cannot easily be established otherwise. In our case, given historical performance data, a *Random Forest* (RF) regressor is learnt (offline) as an approximation of a solver's performance that depends on its algorithmic configuration and instance features. It is demonstrated that this allows us to derive a prescriptive optimization problem whose components result directly from the RF ML model. Given an instance, this problem can be efficiently solved by a tailored B&B procedure. All things considered, this empowers us to compute (online) the (predictably) optimal configuration of a solver on a per-instance basis with minimal computational overhead. By using the well known MIPLIB and a state-of-the-art solver, the proposed methodology is evaluated in an extensive numerical study.

This study is the first of its kind to explore the viability of high-level features of MILPs and several thousand distinct algorithms across heterogeneous instances in the context of the ASP, thereby making unique contributions to the field. Following the computational experiments, compelling evidence supports the viability of the proposed methodology. To be more precise, the effective integration of ML and OR concepts is demonstrated through a sophisticated mapping from instances to configurations of a state-of-the-art solver. This approach successfully narrows the performance gap between the default and virtual-best solver by about half w. r. t. the average runtime and MIP gap. We believe that this paper is a strong testament not only to the benefits of applying the ASP in the context of MIP, but also underscores the more general, continued coalescence of predictive and prescriptive analytics on a methodological level.

8.2 Outlook

In exploring the challenges addressed in this thesis, several promising future research avenues have emerged. With regard to the challenges and opportunities related to auxiliary vehicles, we have hinted at possibilities for extending the current state of research throughout this thesis. Indeed, there are three essential focal points. First, improving existing models and further advancing exact methods. Second, the refinement and design of heuristic methods. Third, the continuous evolution of more generalized models. In this section, these aspects are emphasized, while also providing a glance at the developments that occurred since the related papers have been published.

As hinted in Chapter 6, a straightforward improvement of the model in that chapter may be achieved by adjusting the subtour elimination constraints (Dantzig et al., 1954).

Therefore, it comes as no surprise that the research on improved models and exact algorithms has not halted since that paper has been published. This concerns the TSPD in particular and includes approaches that are based on B&B (Dell'Amico et al., 2021a), B&C (Cavani et al., 2021; Dell'Amico et al., 2022), or *Branch-and-Price* (B&P) (Roberti and Ruthmair, 2021). Currently, the most effective of these approaches is a B&P approach that can solve most instances with 19 customers in the order of a few minutes and several instances with 29 customers in nearly an hour (Roberti and Ruthmair, 2021). Research on improved models and exact algorithms for the VRPD has also continued. In this regard, it is well known that *Branch-Price-and-Cut* (BPC) is the leading exact method for many variants of *Vehicle Routing Problems* (VRPs) (Costa et al., 2019). Consequently, it is not surprising that new ideas based on B&C (Tamke and Buscher, 2021; Tiniç et al., 2023) and B&P or BPC (Schmidt et al., 2023; Zhen et al., 2023; Zhou et al., 2023) exist in the literature. Nevertheless, the VRPD also remains difficult to be solved optimally, as the best of these approaches can only tackle some instances with 29 customers in a matter of nearly an hour. Still, it is important to reflect on what has already been achieved in this thesis and subsequent works that followed: small-sized problems with only 9 customers, which could not be solved within 30 minutes nearly 10 years ago (Murray and Chu, 2015), no longer pose a major challenge. Nevertheless, taking into account these references, there is a large opportunity to advance B&C methods to address the TSPD or B&P methods to address the VRPD. Noteworthy, several of these papers (see, e. g., (Schmidt et al., 2023)) confirm the observation that we made in Chapter 5: i. e., from a computational point of view, the minimum makespan objective remains far more challenging than the minimum cost objective. This reinforces the need for further developments in optimization techniques, especially tailored for makespan-related objectives. The temporal decomposition showcased in Chapter 6 could be a starting point for future endeavors.

Based on the fact that both the TSPD and VRPD remain difficult to be solved optimally, our introductory call for the development of problem-specific heuristics remains relevant. Throughout this thesis, we have illustrated that the development of heuristics is complicated by the fact that problem assumptions in the literature often deviate slightly. As such, this can require the development of problem-specific local search operators that respect the intricate synchronization requirements, which may not readily be adjusted to match different assumptions (see, e. g., (Chen et al., 2021; Moeini et al., 2023; Sacramento et al., 2019)). This observation reaffirms our belief in the relevance of the matheuristic that was presented in Chapter 3. We demonstrated that this approach can easily be adjusted to match the most common problem assumptions. Although, the simplistic route generation procedure in that chapter allowed for very promising results, further refinement seems possible. A straightforward extension would be to integrate state-of-the-art

algorithms for VRPs (e. g., (Accorsi and Vigo, 2021; Vidal, 2022)) into the matheuristic framework. Moreover, the DASP may be improved by drawing from the recent advancements in modeling. Finally, there is an opportunity to directly apply the matheuristic idea to all other problems that were presented in Chapters 2–6 or related problems (see, e. g., (Kloster et al., 2023)).

Concerning the continued development of more general models, the models that were treated in this thesis are subject to certain simplifying assumptions. The term 'rich' VRP has been coined to denote problems that feature many or all aspects of real-word VRP applications (Drexl, 2012a). In this regard, the problems studied in the literature are progressively embracing richness, e. g., by incorporating drones that possess multiple compartments (Amine Masmoudi et al., 2022; Sonnenberg et al., 2019) or vehicles that travel on time-dependent networks (Y. Wang et al., 2022). In addition, there are more realistic state-of-charge considerations, which more adequately capture the load-dependency and its impact on the endurance of drones or robots (Cheng et al., 2020; Di Puglia Pugliese et al., 2021; Kirschstein, 2020; Zhang et al., 2021). Related to that point, several studies investigate the effects of adjustable speeds of drones in particular (Raj and Murray, 2020; Tamke and Buscher, 2023) and the consequential impacts on energy consumption. Despite these advancements, the actual routing and trajectory optimization of drones may yet remain more complex than any of the current models can account for (see (Coutinho et al., 2018) and references therein). On a related note, we find it difficult to find compelling arguments why the launch and retrieval location of drones should be prohibited from coinciding. Although this assumption simplifies the truck-drone-interaction and is common in the literature (see, e. g., (Ha et al., 2018; Murray and Chu, 2015; Sacramento et al., 2019)), more recent works have largely embraced the approach advocated in this thesis, allowing for such operations (see, e. g., (Roberti and Ruthmair, 2021; Tiniç et al., 2023)). This point also applies to the en route operations, illustrated in Chapter 2, which are gaining more traction in the literature (see, e. g., (Masone et al., 2022; Thomas et al., 2023)). Taking all of these considerations into account, there are several avenues for refining existing optimization models, to better reflect the intricacies of energy management and routing when using autonomous auxiliary vehicles.

All of these ideas and developments are a testament to the fact that drones and other emerging technologies still present unique challenges and research opportunities, which may be addressed from an OR perspective. However, with great prospects comes great uncertainty and bridging the proof-of-concept to production gap remains an elusive challenge for the moment. This requires further contributions, likely extending beyond what OR can provide on its own as a scientific discipline (Chung et al., 2020; Poikonen and Campbell, 2020): Thus, it may be important to reflect as a scientific community whether

we are accurately modeling the key capabilities of auxiliary vehicles correctly, if we are focusing on the right objective function (e. g., makespan vs. cost), and in which cases both of these aspects are embedded in the right model (e. g., one presented in Chapters 2–6). This introspection will be crucial in ensuring that research aligns with practical implementation challenges and contributes meaningfully to the evolution of last-mile logistics.

In essence, these overarching questions lead us back to Ackoff's "unsolved problems in problem solving" (Ackoff, 1962) and set the stage for a final outlook concerning Chapter 7. Extending the dataset used in that chapter, e. g., by including the MILP models from previous chapters or related works (e. g., (Caspar et al., 2023)) is a straightforward possibility. This raises the question of how new knowledge may be integrated into the RF, once it has been trained. The trivial approach involves retraining the entire ML model; although this process is fast, it may not be the most efficient (and certainly not the most elegant) way. However, the integration of new knowledge concerns not only new instances, but also different resource limitations than the ones which we considered, in particular different runtime limits that may be relevant to the end-user. A straightforward possibility could be to explicitly include such resource limits as features into a single model, which would make our proposed methodology directly applicable. This would allow determining the anytime-behavior of any configuration and could be a key component for allowing more dynamic control throughout the algorithmic process. While the B&B approach can already determine optimal configurations in a matter of split seconds, further exploration of different branching strategies than the one which we proposed or the search for improved bounds could be worthwhile. Exploring alternative, generally applicable MIP features or other ML models (see also (Lombardi et al., 2017)) also appears to be a viable option. Although, it must be stressed that the RF is extremely well-suited, because it lends itself particularly well to be efficiently exploited by our B&B approach. Nonetheless, this may also hold for other models, such as neural networks. In any case, even when using RFs as a ML model there are further opportunities. As is most common, the RFs were constructed by generating axis-aligned splits; however, testing the same methodology with RFs that are constructed by using oblique cuts may be worthwhile, as these are able to express more complex patterns (Murthy et al., 1994). In any case, a weakness of the demonstrated approach – and to some extent, ML in general – is the fact that a large dataset is required to effectively train the model. Investigating how performance varies when only sparse data is available is a crucial question to explore, as this will also prove to be a key question for scaling the methodology to an even larger set of algorithms and instances. These considerations set the stage for future research, emphasizing the need to revisit foundational questions in problem-solving (Ackoff, 1962) and adapting to the dynamic landscape of prescriptive and predictive optimization (Bengio et al., 2021).

We intend to close this thesis with a quote that was popularized by Donald E. Knuth (Hyde, 2009): "Premature optimization is the root of all evil". Based on the results of this thesis, we remain unconvinced that the opposite of that adage has any truth when it comes to OR. That is to say, we remain highly skeptical that the belated optimization of poorly designed algorithms and models is the ideal solution to any problem. Instead, we advocate for a future where the carefully executed development of prescriptive optimization algorithms and models, as demonstrated in Chapters 2–6, will continue to thrive in a mutualistic symbiosis with predictive approaches that leverage ML to bring them to their full potential, as demonstrated in Chapter 7. This symbiosis could hold great promise for further advancing parts of the discipline.

8.3 References

Accorsi, Luca and Daniele Vigo (2021). "A Fast and Scalable Heuristic for the Solution of Large-Scale Capacitated Vehicle Routing Problems". In: *Transportation Science* 55.4, pp. 832–856.

Ackoff, Russell L. (1962). "Some Unsolved Problems in Problem Solving". In: *Operational Research Quarterly* 13.1, pp. 1–11.

Amine Masmoudi, M. et al. (2022). "Vehicle routing problems with drones equipped with multi-package payload compartments". In: *Transportation Research Part E: Logistics and Transportation Review* 164, p. 102757.

Bengio, Yoshua, Andrea Lodi, and Antoine Prouvost (2021). "Machine learning for combinatorial optimization: A methodological tour d'horizon". In: *European Journal of Operational Research* 290.2, pp. 405–421.

Caspar, Marvin, Daniel Schermer, and Oliver Wendt (2023). "Formulations for the Split Delivery Capacitated Profitable Tour Problem". In: *Computational Science and Its Applications*. Lecture Notes in Computer Science 13956. Cham: Springer, pp. 82–98.

Cavani, Sara, Manuel Iori, and Roberto Roberti (2021). "Exact methods for the traveling salesman problem with multiple drones". In: *Transportation Research Part C: Emerging Technologies* 130, p. 103280.

Chen, Cheng, Emrah Demir, and Yuan Huang (2021). "An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and delivery robots". In: *European Journal of Operational Research* 294.3, pp. 1164–1180.

Cheng, Chun, Yossiri Adulyasak, and Louis-Martin Rousseau (2020). "Drone routing with energy function: Formulation and exact algorithm". In: *Transportation Research Part B: Methodological* 139, pp. 364–387.

Chung, Sung Hoon, Bhawesh Sah, and Jinkun Lee (2020). "Optimization for drone and drone-truck combined operations: A review of the state of the art and future directions". In: *Computers & Operations Research* 123, p. 105004.

Costa, Luciano, Claudio Contardo, and Guy Desaulniers (2019). "Exact Branch-Price-and-Cut Algorithms for Vehicle Routing". In: *Transportation Science* 53.4, pp. 946–985.

Coutinho, Walton Pereira, Maria Battarra, and Jörg Fliege (2018). "The unmanned aerial vehicle routing and trajectory optimisation problem, a taxonomic review". In: *Computers & Industrial Engineering* 120, pp. 116–128.

Dantzig, George B., 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.

Dell'Amico, Mauro, Roberto Montemanni, and Stefano Novellani (2022). "Exact models for the flying sidekick traveling salesman problem". In: *International Transactions in Operational Research* 29.3, pp. 1360–1393.

Dell'Amico, Mauro, Roberto Montemanni, and Stefano Novellani (2021a). "Algorithms based on branch and bound for the flying sidekick traveling salesman problem". In: *Omega* 104, p. 102493.

Di Puglia Pugliese, Luigi, Francesca Guerriero, and Maria Grazia Scutellá (2021). "The Last-Mile Delivery Process with Trucks and Drones Under Uncertain Energy Consumption". In: *Journal of Optimization Theory and Applications* 191.1, pp. 31–67.

Drexl, Michael (2012a). "Rich vehicle routing in theory and practice". In: *Logistics Research* 5.1, pp. 47–63.

Ha, Quang Minh et al. (2018). "On the min-cost Traveling Salesman Problem with Drone". In: *Transportation Research Part C: Emerging Technologies* 86, pp. 597–621.

Hyde, Randall (2009). "The Fallacy of Premature Optimization". In: *ACM Ubiquity* 10.3, pp. 1–5.

Kirschstein, Thomas (2020). "Comparison of energy demands of drone-based and ground-based parcel delivery services". In: *Transportation Research Part D: Transport and Environment* 78, p. 102209.

Kloster, Konstantin et al. (2023). "The multiple traveling salesman problem in presence of drone- and robot-supported packet stations". In: *European Journal of Operational Research* 305.2, pp. 630–643.

Lombardi, Michele, Michela Milano, and Andrea Bartolini (2017). "Empirical decision model learning". In: *Artificial Intelligence* 244, pp. 343–367.

Masone, Adriano, Stefan Poikonen, and Bruce L. Golden (2022). "The multivisit drone routing problem with edge launches: An iterative approach with discrete and continuous improvements". In: *Networks* 80.2, pp. 193–215.

Moeini, Mahdi, Oliver Wendt, and Marius Schummer (2023). "A Bi-objective Routing Problem with Trucks and Drones: Minimizing Mission Time and Energy Consumption". In: *Computational Science and Its Applications*. Lecture Notes in Computer Science 14106. Cham: Springer, pp. 291–308.

Murray, Chase C. and Amanda G. Chu (2015). "The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery". In: *Transportation Research Part C: Emerging Technologies* 54, pp. 86–109.

Murthy, S. K., S. Kasif, and S. Salzberg (1994). "A System for Induction of Oblique Decision Trees". In: *Journal of Artificial Intelligence Research* 2, pp. 1–32.

Poikonen, Stefan and James F. Campbell (2020). "Future directions in drone routing research". In: *Networks* 77.1, pp. 116–126.

Raj, Ritwik and Chase C. Murray (2020). "The multiple flying sidekicks traveling salesman problem with variable drone speeds". In: *Transportation Research Part C: Emerging Technologies* 120, p. 102813.

Rice, John R. (1976). "The Algorithm Selection Problem". In: *Advances in Computers*. Vol. 15. Elsevier, pp. 65–118.

Roberti, Roberto and Mario Ruthmair (2021). "Exact Methods for the Traveling Salesman Problem with Drone". In: *Transportation Science* 55.2, pp. 315–335.

Sacramento, David, David Pisinger, and Stefan Ropke (2019). "An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones". In: *Transportation Research Part C: Emerging Technologies* 102, pp. 289–315.

Schermer, Daniel (2023). "Leveraging Machine Learning for the Per-Instance Configuration of MIP Solvers: An Algorithm Selection Approach". Preprint. Chair of Business Information Systems & Operations Research, University of Kaiserslautern-Landau.

Schermer, Daniel, Mahdi Moeini, and Oliver Wendt (2019a). "A hybrid VNS/Tabu search algorithm for solving the vehicle routing problem with drones and en route operations". In: *Computers & Operations Research* 109, pp. 134–158.

Schermer, Daniel, Mahdi Moeini, and Oliver Wendt (2019b). "A matheuristic for the vehicle routing problem with drones and its variants". In: *Transportation Research Part C: Emerging Technologies* 106, pp. 166–204.

Schermer, Daniel, Mahdi Moeini, and Oliver Wendt (2020a). "A branch-and-cut approach and alternative formulations for the traveling salesman problem with drone". In: *Networks* 76.2, pp. 164–186.

Schermer, Daniel, Mahdi Moeini, and Oliver Wendt (2020b). "The Drone-Assisted Traveling Salesman Problem with Robot Stations". In: *Proceedings of the Hawaii International Conference on System Sciences*, pp. 1308–1317.

Schermer, Daniel, Mahdi Moeini, and Oliver Wendt (2020c). "The Traveling Salesman Drone Station Location Problem". In: *Optimization of Complex Systems: Theory, Models, Algorithms and Applications*. Advances in Intelligent Systems and Computing 991. Cham: Springer, pp. 1129–1138.

Schmidt, Jeanette, Christian Tilk, and Stefan Irnich (2023). *Exact Solution of the Vehicle Routing Problem With Drones*. Working Paper 2311. Gutenberg School of Management and Economics, Johannes Gutenberg-Universität Mainz.

Sonnenberg, Marc-Oliver et al. (2019). "Autonomous Unmanned Ground Vehicles for Urban Logistics: Optimization of Last Mile Delivery Operations". In: *Proceedings of the Hawaii International Conference on System Sciences*, pp. 1538–1547.

Tamke, Felix and Udo Buscher (2021). "A branch-and-cut algorithm for the vehicle routing problem with drones". In: *Transportation Research Part B: Methodological* 144, pp. 174–203.

Tamke, Felix and Udo Buscher (2023). "The vehicle routing problem with drones and drone speed selection". In: *Computers & Operations Research* 152, p. 106112.

Thomas, Teena, Sharan Srinivas, and Chandrasekharan Rajendran (2023). "Collaborative truck multi-drone delivery system considering drone scheduling and en route operations". In: *Annals of Operations Research*, pp. 1–47.

Tiniç, Gizem Ozbaygin et al. (2023). "Exact solution approaches for the minimum total cost traveling salesman problem with multiple drones". In: *Transportation Research Part B: Methodological* 168, pp. 81–123.

Vidal, Thibaut (2022). "Hybrid genetic search for the CVRP: Open-source implementation and SWAP* neighborhood". In: *Computers & Operations Research* 140, p. 105643.

Wang, Yong et al. (2022). "Truck–drone hybrid routing problem with time-dependent road travel time". In: *Transportation Research Part C: Emerging Technologies* 144, p. 103901.

Zhang, Juan et al. (2021). "Energy consumption models for delivery drones: A comparison and assessment". In: *Transportation Research Part D: Transport and Environment* 90, p. 102668.

Zhen, Lu et al. (2023). "Branch-price-and-cut for trucks and drones cooperative delivery". In: *IISE Transactions* 55.3, pp. 271–287.

Zhou, Hang et al. (2023). "An exact algorithm for the two-echelon vehicle routing problem with drones". In: *Transportation Research Part B: Methodological* 168, pp. 124–150.

9 Bibliography

Accorsi, Luca, Andrea Lodi, and Daniele Vigo (2022). "Guidelines for the computational testing of machine learning approaches to vehicle routing problems". In: *Operations Research Letters* 50.2, pp. 229–234.

Accorsi, Luca and Daniele Vigo (2021). "A Fast and Scalable Heuristic for the Solution of Large-Scale Capacitated Vehicle Routing Problems". In: *Transportation Science* 55.4, pp. 832–856.

Achterberg, Tobias and Roland Wunderling (2013). "Mixed Integer Programming: Analyzing 12 Years of Progress". In: *Facets of Combinatorial Optimization*. Ed. by Michael Jünger and Gerhard Reinelt. Berlin, Heidelberg: Springer, pp. 449–481.

Ackoff, Russell L. (1962). "Some Unsolved Problems in Problem Solving". In: *Operational Research Quarterly* 13.1, pp. 1–11.

Adam, Stavros P. et al. (2019). "No Free Lunch Theorem: A Review". In: *Approximation and Optimization: Algorithms, Complexity and Applications*. Springer Optimization and Its Applications, 145. Cham: Springer, pp. 57–82.

Adenso-Díaz, Belarmino and Manuel Laguna (2006). "Fine-Tuning of Algorithms Using Fractional Experimental Designs and Local Search". In: *Operations Research* 54.1, pp. 99–114.

Agatz, Niels, Paul Bouman, and Marie Schmidt (2018). "Optimization Approaches for the Traveling Salesman Problem with Drone". In: *Transportation Science* 52.4, pp. 965–981.

Amine Masmoudi, M. et al. (2022). "Vehicle routing problems with drones equipped with multi-package payload compartments". In: *Transportation Research Part E: Logistics and Transportation Review* 164, p. 102757.

Ansótegui, Carlos, Meinolf Sellmann, and Kevin Tierney (2009). "A Gender-Based Genetic Algorithm for the Automatic Configuration of Algorithms". In: *Principles and Practice of Constraint Programming*. Lecture Notes in Computer Science 5732. Berlin, Heidelberg: Springer, pp. 142–157.

Ansótegui, Carlos et al. (2015). "Model-Based Genetic Algorithms for Algorithm Configuration". In: *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pp. 733–739.

Applegate, David L. et al. (2007). *The Traveling Salesman Problem: A Computational Study*. Princeton University Press.

Archetti, Claudia and M. Grazia Speranza (2014). "A survey on matheuristics for routing problems". In: *EURO Journal on Computational Optimization* 2.4, pp. 223–246.

Auer, Peter, Nicolò Cesa-Bianchi, and Paul Fischer (2002). "Finite-time Analysis of the Multiarmed Bandit Problem". In: *Machine Learning* 47, pp. 235–256.

Aurambout, Jean-Philippe, Konstantinos Gkoumas, and Biagio Ciuffo (2019). "Last mile delivery by drones: an estimation of viable market potential and access to citizens across European cities". In: *European Transport Research Review* 11, pp. 1–21.

Azadeh, Kaveh, René De Koster, and Debjit Roy (2019). "Robotized and Automated Warehouse Systems: Review and Recent Developments". In: *Transportation Science* 53.4, pp. 917–945.

Bartz-Beielstein, Thomas, Martin Zaefferer, and Frederik Rehbach (2021). "In a Nutshell – The Sequential Parameter Optimization Toolbox". In: *arXiv:1712.04076 [cs.MS]*.

Bengio, Yoshua, Andrea Lodi, and Antoine Prouvost (2021). "Machine learning for combinatorial optimization: A methodological tour d'horizon". In: *European Journal of Operational Research* 290.2, pp. 405–421.

Berthold, Timo, Matteo Francobaldi, and Gregor Hendel (2022). "Learning to Use Local Cuts". In: *arXiv:2206.11618 [math.OC]*.

Berthold, Timo and Gregor Hendel (2021). "Learning To Scale Mixed-Integer Programs". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.5, pp. 3661–3668.

Bertsimas, Dimitris and Jack Dunn (2017). "Optimal classification trees". In: *Machine Learning* 106.7, pp. 1039–1082.

Bertsimas, Dimitris and Nathan Kallus (2020). "From Predictive to Prescriptive Analytics". In: *Management Science* 66.3, pp. 1025–1044.

Birattari, Mauro (2009). *Tuning metaheuristics: a machine learning perspective*. 2nd ed. Studies in Computational Intelligence Vol. 197. Berlin: Springer.

Birattari, Mauro et al. (2010). "F-Race and Iterated F-Race: An Overview". In: *Experimental Methods for the Analysis of Optimization Algorithms*. Ed. by Thomas Bartz-Beielstein et al. Berlin, Heidelberg: Springer, pp. 311–336.

Bischl, Bernd et al. (2016). "ASlib: A benchmark library for algorithm selection". In: *Artificial Intelligence* 237, pp. 41–58.

Bixby, Robert E., Andrew E. Boyd, and Ronni R. Indovina (1992). "MIPLIB: a test set of mixed integer programming problems". In: *SIAM News* 25.2, p. 16.

Bouman, Paul, Niels Agatz, and Marie Schmidt (2018a). "Dynamic programming approaches for the traveling salesman problem with drone". In: *Networks* 72.4, pp. 528–542.

Bouman, Paul, Niels Agatz, and Marie Schmidt (2018b). *Instances For The Tsp With Drone (And Some Solutions)*.

Boysen, Nils, Stefan Fedtke, and Stefan Schwerdfeger (2021). "Last-mile delivery concepts: a survey from an operational research perspective". In: *OR Spectrum* 43.1, pp. 1–58.

Boysen, Nils, Stefan Schwerdfeger, and Felix Weidinger (2018a). "Scheduling last-mile deliveries with truck-based autonomous robots". In: *European Journal of Operational Research* 271.3, pp. 1085–1099.

Boysen, Nils et al. (2018b). "Drone delivery from trucks: Drone scheduling for given truck routes". In: *Networks* 72.4, pp. 506–527.

Breedam, Alex van (1994). "An Analysis of the Behavior of Heuristics for the Vehicle Routing Problem for a selection of problems with Vehicle-related, Customer-related, and Time-related Constraints Contents". Ph.D. thesis. University of Antwerp.

Breiman, Leo (1996). "Bagging predictors". In: *Machine Learning* 24.2, pp. 123–140.

Breiman, Leo (2001). "Random Forests". In: *Machine Learning* 45.1, pp. 5–32.

Breiman, Leo et al. (1998). *Classification and regression trees*. 1st ed. Boca Raton: Chapman & Hall/CRC.

Camm, Jeffrey D., Amitabh S. Raturi, and Shigeru Tsubakitani (1990). "Cutting Big M down to Size". In: *Interfaces* 20.5, pp. 61–66.

Carlsson, John Gunnar and Siyuan Song (2018). "Coordinated Logistics with a Truck and a Drone". In: *Management Science* 64.9, pp. 3971–4470.

Caspar, Marvin, Daniel Schermer, and Oliver Wendt (2023). "Formulations for the Split Delivery Capacitated Profitable Tour Problem". In: *Computational Science and Its Applications*. Lecture Notes in Computer Science 13956. Cham: Springer, pp. 82–98.

Cavani, Sara, Manuel Iori, and Roberto Roberti (2021). "Exact methods for the traveling salesman problem with multiple drones". In: *Transportation Research Part C: Emerging Technologies* 130, p. 103280.

Chauhan, Darshan, Avinash Unnikrishnan, and Miguel Figliozzi (2019). "Maximum coverage capacitated facility location problem with range constrained drones". In: *Transportation Research Part C: Emerging Technologies* 99, pp. 1–18.

Chen, Cheng, Emrah Demir, and Yuan Huang (2021). "An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and delivery robots". In: *European Journal of Operational Research* 294.3, pp. 1164–1180.

Cheng, Chun, Yossiri Adulyasak, and Louis-Martin Rousseau (2020). "Drone routing with energy function: Formulation and exact algorithm". In: *Transportation Research Part B: Methodological* 139, pp. 364–387.

Chmiela, Antonia et al. (2021). "Learning to Schedule Heuristics in Branch and Bound". In: *Advances in Neural Information Processing Systems*. Vol. 34, pp. 24235–24246.

Chung, Sung Hoon, Bhawesh Sah, and Jinkun Lee (2020). "Optimization for drone and drone-truck combined operations: A review of the state of the art and future directions". In: *Computers & Operations Research* 123, p. 105004.

Clarke, Geoff and John W. Wright (1964). "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points". In: *Operations Research* 12.4, pp. 568–581.

Cormen, Thomas H. et al. (2014). *Introduction to Algorithms*. 3rd ed. Cambridge: MIT Press.

Costa, Luciano, Claudio Contardo, and Guy Desaulniers (2019). "Exact Branch-Price-and-Cut Algorithms for Vehicle Routing". In: *Transportation Science* 53.4, pp. 946–985.

Coutinho, Walton Pereira, Maria Battarra, and Jörg Fliege (2018). "The unmanned aerial vehicle routing and trajectory optimisation problem, a taxonomic review". In: *Computers & Industrial Engineering* 120, pp. 116–128.

Crainic, Teodor Gabriel, Nicoletta Ricciardi, and Giovanni Storchi (2009). "Models for Evaluating and Planning City Logistics Systems". In: *Transportation Science* 43.4, pp. 432–454.

Dantzig, George B., 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.

Dantzig, George B. and John. H. Ramser (1959). "The Truck Dispatching Problem". In: *Management Science* 6.1, pp. 80–91.

Dell'Amico, Mauro, Roberto Montemanni, and Stefano Novellani (2019). "Models and algorithms for the Flying Sidekick Traveling Salesman Problem". In: *arXiv:1910.02559 [math]*.

Dell'Amico, Mauro, Roberto Montemanni, and Stefano Novellani (2022). "Exact models for the flying sidekick traveling salesman problem". In: *International Transactions in Operational Research* 29.3, pp. 1360–1393.

Dell'Amico, Mauro, Roberto Montemanni, and Stefano Novellani (2021a). "Algorithms based on branch and bound for the flying sidekick traveling salesman problem". In: *Omega* 104, p. 102493.

Dell'Amico, Mauro, Roberto Montemanni, and Stefano Novellani (2021b). "Drone-assisted deliveries: new formulations for the flying sidekick traveling salesman problem". In: *Optimization Letters* 15.5, pp. 1617–1648.

Di Puglia Pugliese, Luigi and Francesca Guerriero (2017). "Last-Mile Deliveries by Using Drones and Classical Vehicles". In: *Optimization and Decision Science: Methodologies and Applications*. Springer Proceedings in Mathematics & Statistics 217. Cham: Springer, pp. 557–565.

Di Puglia Pugliese, Luigi, Francesca Guerriero, and Maria Grazia Scutellá (2021). "The Last-Mile Delivery Process with Trucks and Drones Under Uncertain Energy Consumption". In: *Journal of Optimization Theory and Applications* 191.1, pp. 31–67.

Dorling, Kevin et al. (2017). "Vehicle Routing Problems for Drone Delivery". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 47.1, pp. 70–85.

Drexl, Michael (2012a). "Rich vehicle routing in theory and practice". In: *Logistics Research* 5.1, pp. 47–63.

Drexl, Michael (2012b). "Synchronization in Vehicle Routing—A Survey of VRPs with Multiple Synchronization Constraints". In: *Transportation Science* 46.3, pp. 297–316.

Eggensperger, Katharina, Marius Lindauer, and Frank Hutter (2019). "Pitfalls and Best Practices in Algorithm Configuration". In: *Journal of Artificial Intelligence Research* 64, pp. 861–893.

Es Yurek, Emine and H. Cenk Ozmutlu (2018). "A decomposition-based iterative optimization algorithm for traveling salesman problem with drone". In: *Transportation Research Part C: Emerging Technologies* 91, pp. 249–262.

Fair Isaac Corporation (2023). *FICO Xpress Optimizer Reference Manual*.

Firat, Murat et al. (2020). "Column generation based heuristic for learning classification trees". In: *Computers & Operations Research* 116, p. 104866.

Fischetti, Matteo, Ivana Ljubić, and Markus Sinnl (2016). "Benders decomposition without separability: A computational study for capacitated facility location problems". In: *European Journal of Operational Research* 253.3, pp. 557–569.

Fischetti, Matteo, Ivana Ljubić, and Markus Sinnl (2017). "Redesigning Benders Decomposition for Large-Scale Facility Location". In: *Management Science* 63.7, pp. 2146–2162.

Fragapane, Giuseppe et al. (2021). "Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda". In: *European Journal of Operational Research* 294.2, pp. 405–426.

Freitas, Júlia Cária de and Puca Huachi Vaz Penna (2018). "A Randomized Variable Neighborhood Descent Heuristic to Solve the Flying Sidekick Traveling Salesman Problem". In: *Electronic Notes in Discrete Mathematics* 66, pp. 95–102.

Freitas, Júlia Cária de and Puca Huachi Vaz Penna (2020). "A variable neighborhood search for flying sidekick traveling salesman problem". In: *International Transactions in Operational Research* 27.1, pp. 267–290.

Gendreau, Michel and Jean-Yves Potvin (2010). *Handbook of Metaheuristics*. 2nd ed. Cham: Springer.

Gentry, Nicholas Kristofer, Raphael Hsieh, and Luan Khai Nguyen (2016). "Multi-use UAV docking station systems and methods". US9387928B1 (United States Patent).

Ghiani, Gianpaolo, Gilbert Laporte, and Roberto Musmanno (2013). *Introduction to logistics systems management*. 2nd ed. Wiley series in operations research and management science. Chichester: John Wiley & Sons, Ltd.

Gittins, J. C. (1979). "Bandit Processes and Dynamic Allocation Indices". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 41.2, pp. 148–164.

Gleixner, Ambros et al. (2021). "MIPLIB 2017: data-driven compilation of the 6th mixed-integer programming library". In: *Mathematical Programming Computation* 13, pp. 443–490.

Glover, Fred (1986). "Future paths for integer programming and links to artificial intelligence". In: *Computers & Operations Research* 13.5, pp. 533–549.

Goeke, Dominik (2018). "Emerging trends in logistics". Ph.D. thesis. Technische Universität Kaiserslautern.

Gomory, Ralph E (1963). "An algorithm for integer solutions to linear programs". In: *Recent advances in mathematical programming* 64.14, pp. 260–302.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep learning*. Adaptive computation and machine learning. Cambridge: The MIT Press.

Guastaroba, G., M. G. Speranza, and D. Vigo (2016). "Intermediate Facilities in Freight Transportation Planning: A Survey". In: *Transportation Science* 50.3, pp. 763–789.

Gupta, Prateek et al. (2020). "Hybrid Models for Learning to Branch". In: *Advances in Neural Information Processing Systems*. Vol. 33, pp. 18087–18097.

Gurobi Optimization, LLC (2023). *Gurobi Optimizer Reference Manual*.

Ha, Quang Minh et al. (2018). "On the min-cost Traveling Salesman Problem with Drone". In: *Transportation Research Part C: Emerging Technologies* 86, pp. 597–621.

Ha, Quang Minh et al. (2020). "A hybrid genetic algorithm for the traveling salesman problem with drone". In: *Journal of Heuristics* 26, pp. 219–247.

Hansen, Pierre et al. (2017). "Variable neighborhood search: basics and variants". In: *EURO Journal on Computational Optimization* 5.3, pp. 423–454.

Hastie, Trevor, Robert Tibshirani, and J. H. Friedman (2009). *The elements of statistical learning: data mining, inference, and prediction*. 2nd ed. Springer series in statistics. New York: Springer.

Hinkelmann, Klaus and Oscar Kempthorne (1994). *Design and analysis of experiments*. Rev. ed. Wiley series in probability and mathematical statistics. New York: Wiley.

Hof, Julian (2019). "Metaheuristics for Vehicle-Routing Problems Arising in Sustainable Logistics". Ph.D. thesis. Technische Universität Kaiserslautern.

Hoos, Holger H. (2011). "Automated Algorithm Configuration and Parameter Tuning". In: *Autonomous Search*. Ed. by Youssef Hamadi, Eric Monfroy, and Frédéric Saubion. Berlin, Heidelberg: Springer, pp. 37–71.

Hoos, Holger H., Frank Hutter, and Kevin Leyton-Brown (2021). "Automated Configuration and Selection of SAT Solvers". In: *Handbook of Satisfiability*. Ed. by Armin Biere et al. Frontiers in Artificial Intelligence and Applications 336. IOS Press, pp. 481–507.

Hutter, Frank, Holger H. Hoos, and Kevin Leyton-Brown (2010). "Automated Configuration of Mixed Integer Programming Solvers". In: *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. Lecture Notes in Computer Science 6140. Berlin, Heidelberg: Springer, pp. 186–202.

Hutter, Frank, Holger H. Hoos, and Kevin Leyton-Brown (2011). "Sequential Model-Based Optimization for General Algorithm Configuration". In: *Learning and Intelligent Optimization*. Lecture Notes in Computer Science 6683. Berlin, Heidelberg: Springer, pp. 507–523.

Hutter, Frank, Holger H. Hoos, and Thomas Stützle (2007). "Automatic Algorithm Configuration Based on Local Search". In: *Proceedings of the 22nd national conference on Artificial intelligence*. Vol. 2. AAAI Press, pp. 1152–1157.

Hutter, Frank et al. (2006). "Performance Prediction and Automated Tuning of Randomized and Parametric Algorithms". In: *Principles and Practice of Constraint Programming*. Lecture Notes in Computer Science 4204. Berlin, Heidelberg: Springer, pp. 213–228.

Hutter, Frank et al. (2009). "ParamILS: An Automatic Algorithm Configuration Framework". In: *Journal of Artificial Intelligence Research* 36, pp. 267–306.

Hyde, Randall (2009). "The Fallacy of Premature Optimization". In: *ACM Ubiquity* 10.3, pp. 1–5.

Iommazzo, Gabriele (2021). "Algorithmic configuration by learning and optimization". Ph.D. thesis. Institut polytechnique de Paris.

Iommazzo, Gabriele et al. (2020a). "A Learning-Based Mathematical Programming Formulation for the Automatic Configuration of Optimization Solvers". In: *Machine Learning, Optimization, and Data Science*. Lecture Notes in Computer Science 12565. Cham: Springer, pp. 700–712.

Iommazzo, Gabriele et al. (2020b). "Learning to Configure Mathematical Programming Solvers by Mathematical Programming". In: *Learning and Intelligent Optimization*. Lecture Notes in Computer Science 12096. Cham: Springer, pp. 377–389.

Jones, Donald R. (2001). "A Taxonomy of Global Optimization Methods Based on Response Surfaces". In: *Journal of Global Optimization* 21.4, pp. 345–383.

Jones, Donald R., Matthias Schonlau, and William J. Welch (1998). "Efficient Global Optimization of Expensive Black-Box Functions". In: *Journal of Global Optimization* 13.4, pp. 455–493.

Jordan, Michael I. and Tom M. Mitchell (2015). “Machine learning: Trends, perspectives, and prospects”. In: *Science* 349.6245, pp. 255–260.

Jünger, Michael et al., eds. (2010). *50 Years of Integer Programming 1958-2008*. Berlin, Heidelberg: Springer.

Kadioglu, Serdar et al. (2010). “ISAC – Instance-Specific Algorithm Configuration”. In: *Proceedings of the 19th European Conference on Artificial Intelligence*. IOS Press, pp. 751–756.

Kerschke, Pascal and Heike Trautmann (2019). “Automated Algorithm Selection on Continuous Black-Box Problems by Combining Exploratory Landscape Analysis and Machine Learning”. In: *Evolutionary Computation* 27.1, pp. 99–127.

Kerschke, Pascal et al. (2018). “Leveraging TSP Solver Complementarity through Machine Learning”. In: *Evolutionary Computation* 26.4, pp. 597–620.

Kerschke, Pascal et al. (2019). “Automated Algorithm Selection: Survey and Perspectives”. In: *Evolutionary Computation* 27.1, pp. 3–45.

Khalil, Elias B. et al. (2016). “Learning to branch in mixed integer programming”. In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. Vol. 30. AAAI Press, pp. 724–731.

Khalil, Elias B. et al. (2017). “Learning to Run Heuristics in Tree Search”. In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pp. 659–666.

Kim, Sungwoo and Ilkyeong Moon (2018). “Traveling Salesman Problem With a Drone Station”. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 49.1, pp. 42–52.

Kirschstein, Thomas (2020). “Comparison of energy demands of drone-based and ground-based parcel delivery services”. In: *Transportation Research Part D: Transport and Environment* 78, p. 102209.

Kloster, Konstantin et al. (2023). “The multiple traveling salesman problem in presence of drone- and robot-supported packet stations”. In: *European Journal of Operational Research* 305.2, pp. 630–643.

Koch, Thorsten et al. (2022). “Progress in mathematical programming solvers from 2001 to 2020”. In: *EURO Journal on Computational Optimization* 10, p. 100031.

Kotthoff, Lars (2016). “Algorithm Selection for Combinatorial Search Problems: A Survey”. In: *Data Mining and Constraint Programming*. Lecture Notes in Computer Science 10101. Cham: Springer, pp. 149–190.

Kotthoff, Lars et al. (2015). “Improving the State of the Art in Inexact TSP Solving Using Per-Instance Algorithm Selection”. In: *Learning and Intelligent Optimization*. Lecture Notes in Computer Science 8994. Cham: Springer, pp. 202–217.

Kumar, Vipin and V. Nageshwara Rao (1987). "Parallel depth first search. Part II. Analysis". In: *International Journal of Parallel Programming* 16.6, pp. 501–519.

Land, A. H. and A. G. Doig (1960). "An Automatic Method of Solving Discrete Programming Problems". In: *Econometrica* 28.3, pp. 497–520.

Laporte, Gilbert and Paolo Toth (2022). "A gap in scientific reporting". In: *4OR* 20, pp. 169–171.

Leyton-Brown, Kevin, Eugene Nudelman, and Yoav Shoham (2002). "Learning the Empirical Hardness of Optimization Problems: The Case of Combinatorial Auctions". In: *Principles and Practice of Constraint Programming*. Lecture Notes in Computer Science 2470. Berlin, Heidelberg: Springer, pp. 556–572.

Leyton-Brown, Kevin et al. (2003). "A Portfolio Approach to Algorithm Selection". In: *Proceedings of the 18th international joint conference on Artificial intelligence*. Morgan Kaufmann Publishers Inc., pp. 1542–1543.

Lin, S. and B. W. Kernighan (1973). "An Effective Heuristic Algorithm for the Traveling-Salesman Problem". In: *Operations Research* 21.2, pp. 498–516.

Lodi, Andrea (2013). "The Heuristic (Dark) Side of MIP Solvers". In: *Hybrid Metaheuristics*. Ed. by El-Ghazali Talbi. Studies in Computational Intelligence 434. Berlin, Heidelberg: Springer, pp. 273–284.

Lodi, Andrea and Andrea Tramontani (2014). "Performance Variability in Mixed-Integer Programming". In: *Theory Driven by Influential Applications*. Ed. by J. Cole Smith. Theory Driven by Influential Applications, pp. 1–12.

Lodi, Andrea and Giulia Zarpellon (2017). "On learning and branching: a survey". In: *TOP* 25.2, pp. 207–236.

Lombardi, Michele, Michela Milano, and Andrea Bartolini (2017). "Empirical decision model learning". In: *Artificial Intelligence* 244, pp. 343–367.

López-Ibáñez, Manuel et al. (2016). "The irace package: Iterated racing for automatic algorithm configuration". In: *Operations Research Perspectives* 3, pp. 43–58.

Loreggia, Andrea et al. (2016). "Deep learning for algorithm portfolios". In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. Vol. 30. AAAI Press, pp. 1280–1286.

Macrina, Giusy et al. (2020). "Drone-aided routing: A literature review". In: *Transportation Research Part C: Emerging Technologies* 120, p. 102762.

Marinelli, Mario et al. (2018). "En route truck–drone parcel delivery for optimal vehicle routing strategies". In: *IET Intelligent Transport Systems* 12.4, pp. 253–261.

Maron, Oden and Andrew W. Moore (1997). "The Racing Algorithm: Model Selection for Lazy Learners". In: *Artificial Intelligence Review* 11.1, pp. 193–225.

Masone, Adriano, Stefan Poikonen, and Bruce L. Golden (2022). "The multivisit drone routing problem with edge launches: An iterative approach with discrete and continuous improvements". In: *Networks* 80.2, pp. 193–215.

Miller, C. E., A. W. Tucker, and R. A. Zemlin (1960). "Integer Programming Formulation of Traveling Salesman Problems". In: *Journal of the ACM* 7.4, pp. 326–329.

Min, Hokey, Vaidyanathan Jayaraman, and Rajesh Srivastava (1998). "Combined location-routing problems: A synthesis and future research directions". In: *European Journal of Operational Research* 108.1, pp. 1–15.

Mingers, John and Leroy White (2010). "A review of the recent contribution of systems thinking to operational research and management science". In: *European Journal of Operational Research* 207.3, pp. 1147–1161.

Mitchell, Tom M. (1997). *Machine Learning*. McGraw-Hill series in computer science. New York: McGraw-Hill.

Mittelmann, Hans D. (2020). "Benchmarking Optimization Software - a (Hi)Story". In: *SN Operations Research Forum* 1.2, pp. 1–6.

Mladenović, N and P Hansen (1997). "Variable Neighborhood Search". In: *Computers & Operations Research* 24.11, pp. 1097–1100.

Mockus, J., V. Tiesis, and A. Zilinskas (1978). "The application of Bayesian methods for seeking the extremum". In: *Towards Global Optimization* 2, pp. 117–129.

Moeini, Mahdi and Hagen Salewski (2020). "A Genetic Algorithm for Solving the Truck-Drone-ATV Routing Problem". In: *Optimization of Complex Systems: Theory, Models, Algorithms and Applications*. Advances in Intelligent Systems and Computing 991. Cham: Springer, pp. 1023–1032.

Moeini, Mahdi, Oliver Wendt, and Marius Schummer (2023). "A Bi-objective Routing Problem with Trucks and Drones: Minimizing Mission Time and Energy Consumption". In: *Computational Science and Its Applications*. Lecture Notes in Computer Science 14106. Cham: Springer, pp. 291–308.

Morandi, Nicola et al. (2023). "The Traveling Salesman Problem with Drones: The Benefits of Retraversing the Arcs". In: *Transportation Science* 57.5, pp. 1340–1358.

Mortenson, Michael J., Neil F. Doherty, and Stewart Robinson (2015). "Operational research from Taylorism to Terabytes: A research agenda for the analytics age". In: *European Journal of Operational Research* 241.3, pp. 583–595.

Murray, Chase C. and Amanda G. Chu (2015). "The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery". In: *Transportation Research Part C: Emerging Technologies* 54, pp. 86–109.

Murthy, S. K., S. Kasif, and S. Salzberg (1994). "A System for Induction of Oblique Decision Trees". In: *Journal of Artificial Intelligence Research* 2, pp. 1–32.

Nagy, Gábor and Saïd Salhi (2006). “Location-routing: Issues, models and methods”. In: *European Journal of Operational Research* 177.2, pp. 649–672.

Otto, Alena et al. (2018). “Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey”. In: *Networks* 72.4, pp. 411–458.

Padberg, Manfred and Giovanni Rinaldi (1991). “A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems”. In: *SIAM Review* 33.1, pp. 60–100.

Pedregosa, F. et al. (2011). “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Pelletier, Samuel, Ola Jabali, and Gilbert Laporte (2016). “50th Anniversary Invited Article—Goods Distribution with Electric Vehicles: Review and Research Perspectives”. In: *Transportation Science* 50.1, pp. 3–22.

Petropoulos, Fotios et al. (2023). “Operational Research: Methods and Applications”. In: *arXiv:2303.14217 [math.OC]*.

Poikonen, Stefan and James F. Campbell (2020). “Future directions in drone routing research”. In: *Networks* 77.1, pp. 116–126.

Poikonen, Stefan, Bruce Golden, and Edward A. Wasil (2019). “A Branch-and-Bound Approach to the Traveling Salesman Problem with a Drone”. In: *INFORMS Journal on Computing* 31.2, pp. 335–346.

Poikonen, Stefan, Xingyin Wang, and Bruce Golden (2017). “The vehicle routing problem with drones: Extended models and connections”. In: *Networks* 70.1, pp. 34–43.

Ponza, Andrea (2016). “Optimization of drone-assisted parcel delivery”. MA thesis. Università Degli Studi di Padova.

Probst, Philipp, Anne-Laure Boulesteix, and Bernd Bischl (2019). “Tunability: Importance of Hyperparameters of Machine Learning Algorithms”. In: *Journal of Machine Learning Research* 20.1, pp. 1934–1965.

Prodhon, Caroline and Christian Prins (2014). “A survey of recent research on location-routing problems”. In: *European Journal of Operational Research* 238.1, pp. 1–17.

Pushak, Yasha and Holger H. Hoos (2018). “Algorithm Configuration Landscapes: More Benign Than Expected?” In: *Parallel Problem Solving from Nature*. Lecture Notes in Computer Science 11102. Cham: Springer, pp. 271–283.

Raj, Ritwik and Chase C. Murray (2020). “The multiple flying sidekicks traveling salesman problem with variable drone speeds”. In: *Transportation Research Part C: Emerging Technologies* 120, p. 102813.

Rao, Nageshwara N. and Vipin Kumar (1987). “Parallel depth first search. Part I. Implementation”. In: *International Journal of Parallel Programming* 16.6, pp. 479–499.

Rice, John R. (1976). "The Algorithm Selection Problem". In: *Advances in Computers*. Vol. 15. Elsevier, pp. 65–118.

Roberti, Roberto and Mario Ruthmair (2021). "Exact Methods for the Traveling Salesman Problem with Drone". In: *Transportation Science* 55.2, pp. 315–335.

Rojas Viloria, Daniela et al. (2020). "Unmanned aerial vehicles/drones in vehicle routing problems: a literature review". In: *International Transactions in Operational Research* 28.4, pp. 1626–1657.

Russell, Stuart J., Peter Norvig, and Ernest Davis (2010). *Artificial intelligence: a modern approach*. 3rd ed. Prentice Hall series in artificial intelligence. Upper Saddle River: Prentice Hall.

Sacramento, David, David Pisinger, and Stefan Ropke (2019). "An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones". In: *Transportation Research Part C: Emerging Technologies* 102, pp. 289–315.

Savelsbergh, Martin and Tom Van Woensel (2016). "50th Anniversary Invited Article—City Logistics: Challenges and Opportunities". In: *Transportation Science* 50.2, pp. 579–590.

Schede, Elias et al. (2022). "A Survey of Methods for Automated Algorithm Configuration". In: *Journal of Artificial Intelligence Research* 75, pp. 425–487.

Schermer, Daniel (2019). "Integration of Drones in Last-Mile Delivery: The Vehicle Routing Problem with Drones". In: *Operations Research Proceedings 2018*. Operations Research Proceedings. Cham: Springer, pp. 17–22.

Schermer, Daniel (2023). "Leveraging Machine Learning for the Per-Instance Configuration of MIP Solvers: An Algorithm Selection Approach". Preprint. Chair of Business Information Systems & Operations Research, University of Kaiserslautern-Landau.

Schermer, Daniel, Mahdi Moeini, and Oliver Wendt (2018a). "A Variable Neighborhood Search Algorithm for Solving the Vehicle Routing Problem with Drones". Technical Report. Business Information Systems & Operations Research, Technische Universität Kaiserslautern.

Schermer, Daniel, Mahdi Moeini, and Oliver Wendt (2018b). "Algorithms for Solving the Vehicle Routing Problem with Drones". In: *Intelligent Information and Database Systems*. Lecture Notes in Computer Science 10751. Cham: Springer, pp. 352–361.

Schermer, Daniel, Mahdi Moeini, and Oliver Wendt (2019a). "A hybrid VNS/Tabu search algorithm for solving the vehicle routing problem with drones and en route operations". In: *Computers & Operations Research* 109, pp. 134–158.

Schermer, Daniel, Mahdi Moeini, and Oliver Wendt (2019b). "A matheuristic for the vehicle routing problem with drones and its variants". In: *Transportation Research Part C: Emerging Technologies* 106, pp. 166–204.

Schermer, Daniel, Mahdi Moeini, and Oliver Wendt (2020a). "A branch-and-cut approach and alternative formulations for the traveling salesman problem with drone". In: *Networks* 76.2, pp. 164–186.

Schermer, Daniel, Mahdi Moeini, and Oliver Wendt (2020b). "The Drone-Assisted Traveling Salesman Problem with Robot Stations". In: *Proceedings of the Hawaii International Conference on System Sciences*, pp. 1308–1317.

Schermer, Daniel, Mahdi Moeini, and Oliver Wendt (2020c). "The Traveling Salesman Drone Station Location Problem". In: *Optimization of Complex Systems: Theory, Models, Algorithms and Applications*. Advances in Intelligent Systems and Computing 991. Cham: Springer, pp. 1129–1138.

Schmidt, Jeanette, Christian Tilk, and Stefan Irnich (2023). *Exact Solution of the Vehicle Routing Problem With Drones*. Working Paper 2311. Gutenberg School of Management and Economics, Johannes Gutenberg-Universität Mainz.

Sluijk, Natasja et al. (2023). "Two-echelon vehicle routing problems: A literature review". In: *European Journal of Operational Research* 304.3, pp. 865–886.

Sonnenberg, Marc-Oliver et al. (2019). "Autonomous Unmanned Ground Vehicles for Urban Logistics: Optimization of Last Mile Delivery Operations". In: *Proceedings of the Hawaii International Conference on System Sciences*, pp. 1538–1547.

Stone, M. (1974). "Cross-Validatory Choice and Assessment of Statistical Predictions". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 36.2, pp. 111–147.

Sutton, Richard S. and Andrew G. Barto (2018). *Reinforcement learning: an introduction*. 2nd ed. Adaptive computation and machine learning series. Cambridge, The MIT Press.

Tamke, Felix and Udo Buscher (2021). "A branch-and-cut algorithm for the vehicle routing problem with drones". In: *Transportation Research Part B: Methodological* 144, pp. 174–203.

Tamke, Felix and Udo Buscher (2023). "The vehicle routing problem with drones and drone speed selection". In: *Computers & Operations Research* 152, p. 106112.

Tang, Yunhao, Shipra Agrawal, and Yuri Faenza (2020). "Reinforcement Learning for Integer Programming: Learning to Cut". In: *Proceedings of the 37th International Conference on Machine Learning*, pp. 9367–9376.

Thomas, Teena, Sharan Srinivas, and Chandrasekharan Rajendran (2023). "Collaborative truck multi-drone delivery system considering drone scheduling and en route operations". In: *Annals of Operations Research*, pp. 1–47.

Tiniç, Gizem Ozbaygin et al. (2023). "Exact solution approaches for the minimum total cost traveling salesman problem with multiple drones". In: *Transportation Research Part B: Methodological* 168, pp. 81–123.

Toth, Paolo and Daniele Vigo (2002). *The vehicle routing problem*. 1st ed. SIAM Series on Discrete Mathematics and Applications. Philadelphia: SIAM.

Toth, Paolo and Daniele Vigo, eds. (2014). *Vehicle routing: problems, methods, and applications*. 2nd ed. MOS-SIAM series on optimization. Philadelphia: SIAM.

Vidal, Thibaut (2022). "Hybrid genetic search for the CVRP: Open-source implementation and SWAP* neighborhood". In: *Computers & Operations Research* 140, p. 105643.

Wang, Xingyin, Stefan Poikonen, and Bruce Golden (2017). "The vehicle routing problem with drones: several worst-case results". In: *Optimization Letters* 11.4, pp. 679–697.

Wang, Yong et al. (2022). "Truck–drone hybrid routing problem with time-dependent road travel time". In: *Transportation Research Part C: Emerging Technologies* 144, p. 103901.

Wilcoxon, Frank (1945). "Individual comparisons by ranking methods". In: *Biometrics Bulletin* 1.6, pp. 80–83.

Wolpert, D.H. and W.G. Macready (1997). "No free lunch theorems for optimization". In: *IEEE Transactions on Evolutionary Computation* 1.1, pp. 67–82.

Xu, Lin et al. (2008). "SATzilla: Portfolio-based Algorithm Selection for SAT". In: *Journal of Artificial Intelligence Research* 32, pp. 565–606.

Xu, Lin et al. (2011). "Hydra-MIP: Automated algorithm configuration and selection for mixed integer programming". In: *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*. AAAI Press, pp. 16–30.

Zarpellon, Giulia et al. (2021). "Parameterizing Branch-and-Bound Search Trees to Learn Branching Policies". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35(5). AAAI Press, pp. 3931–3939.

Zhang, Juan et al. (2021). "Energy consumption models for delivery drones: A comparison and assessment". In: *Transportation Research Part D: Transport and Environment* 90, p. 102668.

Zhen, Lu et al. (2023). "Branch-price-and-cut for trucks and drones cooperative delivery". In: *IISE Transactions* 55.3, pp. 271–287.

Zhou, Hang et al. (2023). "An exact algorithm for the two-echelon vehicle routing problem with drones". In: *Transportation Research Part B: Methodological* 168, pp. 124–150.

Office: Gottlieb-Daimler-Straße – 67653 Kaiserslautern
Mail: daniel.schermer@rptu.de

Daniel Schermer

Curriculum Vitae

Scientific Career	
2018–Today	**Research Associate & PhD Student**
	Chair of Business Information Systems & Operations Research
	Prof. Dr. Oliver Wendt
	University of Kaiserslautern-Landau
2016–2018	**Master of Science**
	University of Kaiserslautern
	Business Management and Engineering
	in the Field of Electrical Engineering
2011–2016	**Bachelor of Science**
	University of Kaiserslautern
	Business Management and Engineering
	in the Field of Electrical Engineering