ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

in cotutela con Università di Passau

DOTTORATO DI RICERCA IN
Ingegneria Biomedica, Elettrica e dei Sistemi

Ciclo 38

Settore Concorsuale: Ricerca operativa (01/A6)

Settore Scientifico Disciplinare: Ricerca Operativa (MAT/09)

STUDIES ON OPTIMIZATION PROBLEMS WITH DYNAMICALLY ARRIVING INFORMATION

Presentata da: ***Catherine Lorenz***

Coordinatore Dottorato
Prof. Dr. Michele Monaci

Supervisori
Prof. Dr. Daniele Vigo
Prof. Dr. Alena Otto

Esame finale anno 2024

Acknowledgements

For the present dissertation, I would like to express my gratitude for the support I have received from many wonderful people.

First, I would like to express my warmest thanks to my supervisor, mentor, and co-author, Prof. Alena Otto, for her excellent guidance and relentless support. I could always count on her for a listening ear, constructive feedback, and guidance in both research and beyond. I consider myself very fortunate for the research freedom, the unwavering trust in my abilities, and the invaluable opportunities for the academic development I received from her side. I sincerely hope we can maintain our friendship and productive, enjoyable collaboration in the future.

I also wish to thank my second supervisor and co-author, Prof. Daniele Vigo, for sharing ideas, advice, and potential research paths with me in our truthfully enjoyable research discussions. I am immensely grateful for the opportunity to pursue a dual doctorate under his supervision, and I highly appreciate the bureaucratic efforts he undertook to make this possible. Thanks for his generosity in sharing his office with me and the warm welcome he extended to me at the University of Bologna.

Furthermore, a big thank you goes to Prof. Stefan Irnich for taking the time to read and review my dissertation.

I extend my gratitude to all my co-authors: Prof. Michel Gendreau, Prof. Bruce Golden, Dr. Yuchen Luo, Dr. Nicola Mimmo, Prof. Erwin Pesch, and Luis Rocha. Working with them has been a pleasure. A special thanks goes to Prof. Michel Gendreau for inviting me to CIRRELT in Montreal, where I had the privilege of working under his mentorship for four months. Besides gaining valuable insights on routing research and the scientific community, I always greatly enjoyed our friendly, casual chats.

A heartfelt thank you goes to my colleagues and fellow PhD candidates at the Chair of Management Science/Operations and Supply Chain Management of the University of Passau: Amir Hosseini, Ayse Karabair, Maya Marten, Andrei Newskii, and Luis Rocha, for being with me on this journey. They always had my back when I felt overwhelmed with organizational- or teaching tasks, and consistently offered their help and support. Their valuable tips and feedback elevated all my conference presentations and boosted my confidence for my talks. I would like to specially thank Susanne Fritsch who is a spectacular secretary and the backbone of our chair. She consistently guided me through the bureaucratic maze, took work of my shoulders, and with her foresight and organizational skills ensured I never missed any important deadline.

I thank the Operations Research group of the Department of Electrical, Electronic, and Information Engineering at the University of Bologna; especially Prof. Valentina Cacchiani, Prof. Enrico Malaguri, Prof. Michele Monaci, Prof. Paolo Paronuzzi, Dr. Antonio Punzo, and Federico Michelotto, for taking me out for lunch every single day and making me feel very welcome at the University of Bologna. Through you, I have come closer to the Italian culture and language.

I am grateful to the Women's Advancement Office of the University of Passau and the Bayerische Forschungsallianz (BayFOR) for funding the research trips that have helped me to develop as a scientist.

Finally, I am so thankful for my wonderful dear friends and family. Even through distance, your warmth is always with me, and I know I can always count on you. Our meetings and phone calls have always helped me overcome work stress and remind me of what truly matters in life. I am deeply indebted to my parents, Milena and Marco Lorenz, and my aunt, Michele Lorenz, for their mental and financial support since the first day of my education and their unwavering belief in my success. Without them, none of this could have happened. Last but not least, my deepest thanks go to my best friend, greatest supporter, and life partner, Valentin Damjantschitsch, who has been my strongest pillar in the past years. Thank you for supporting me through moments of anxiety, for your patience through all my chaos, for sharing all the happy times with me, and for always being there no matter what. I love you.

Contents

List of Figures

List of Tables

List of Abbreviations

ALNS	**A**daptive **L**arge **N**eighborhood **S**earch
Avg	**Av**era**g**e
B&P	**B**ranch-**a**nd-**P**rice
BS	**B**alas-**S**imonetti
BS-R	**B**alas-**S**imonetti neighborhood with **R**eplenishment
CCTP	**C**overing **C**anadian **T**raveler **P**roblem
CD	**C**onservative **D**elivery
CIOPT	**C**omplete **I**nformation **O**ptimum
CIOS	**C**omplete **I**nformation **O**ptimal **P**olicy **S**olution
CPS	**C**ircular **P**ath with **S**pikes
CR	**C**ompetitive **R**atio
CRediT	**C**ontributor **R**oles **T**axonomy
CTP	**C**anadian **T**raveler **P**roblem
DRL	**D**eep **R**einforcement **L**earning
DRP-E	**D**rone **R**outing **P**roblem with **E**nergy replenishment
DD	**D**ynamic **D**eterministic
DP	**D**ynamic **P**rogramming
DV	**D**elivery **V**ehicle
DS	**D**ynamic **S**tochastic
FIFO	**F**irst-**I**n-**F**irst-**O**ut
ILP	**I**nteger **L**inear **P**rogram
KWT	**K**ruskal-**W**allis **T**est
Max	**Max**imum
MIP	**M**ixed-**I**nteger **P**rogram
MPA	**M**ultiple **P**lan **A**pproach
MRS	**M**obile **R**eplenishment **S**tation
MSA	**M**ultiple **S**cenario **A**pproach
MTZ	**M**iller-**T**ucker-**Z**emlin
OBP	**O**rder **B**atching **P**roblem
OBSRP-R	**O**rder **B**atching, **S**equencing, and Picker **R**outing **P**roblem with **R**elease Times
OBRP	**O**rder **B**atching and Picker **R**outing **P**roblem
OOBSRP	**O**nline **O**rder **B**atching, **S**equencing, and Picker **R**outing **P**roblem
OOBP	**O**nline **O**rder **B**atching **P**roblem
OTSP	**O**nline **T**raveling **S**alesman **P**roblem
PAH	**P**lan **A**t **H**ome
PDF	**P**robability **D**istribution **F**unction
PS	**P**olicy **S**olution
Reopt	**Reopt**imization
RL	**R**eplenishment **L**ocation
SA	**S**imulated **A**nnealing
SF	**S**urveillance-**F**irst
SKU	**S**tock **K**eeping **U**nit

SoD	**S**ource **o**f **D**ynamism and uncertainty
SP	**S**ub**p**roblem
TSP	**T**raveling **S**alesman **P**roblem
TTRP	**T**ruck and **T**railer **R**outing **P**roblem
VND	**V**ariable **N**eighborhood **D**escent
VRP	**V**ehicle **R**outing **P**roblem
VTWB	**V**ariable **T**ime **W**indow **B**atching
VTW	**V**ariable **T**ime **W**indow strategy

Dedicated to my parents,
Marco and Milena

Chapter 1

Introduction

In the fast-paced, connected world of the 21st century, our financial systems, industrial processes, communication- and transportation networks, wholesale and retail, are all essentially unfolding on an *online*- or even *real-time* scale. Advanced IT infrastructures, like cloud-computing and web sockets, allow the share of current stock prices with traders and small investors at seconds- or even milliseconds intervals (Google Cloud Services, 2023). Satellite imagery feeds traffic control centers and drivers with immediate warnings on road conditions and traffic. In smart manufacturing plants, sensors and machinery capture and share detailed information on the production's progress with the control center in real-time, through the Internet of Things (Cañas et al., 2021; Phuyal et al., 2020). Services and goods are ordered comfortably and spontaneously through mobile applications and e-commerce platforms with just a few clicks. As quickly as these messages are sent, the recipients are expected to process the information – and *react* accordingly.

From an optimization perspective, the decision problems that arise in these environments, are classified as *dynamic - or online optimization problems*. More formally, those are *sequential decision problems*, for which the implementation of decisions may start, before all the relevant input data of the problem is available. Some- or all of the input data arrives dynamically as the solution evolves, and the decisions may be altered accordingly. An *online algorithm* defines a sequence of decisions for these problems, that prescribe a tentative solution given the *available* input at each point when new information is received, without the explicit knowledge of future input. In some contexts, we also use the alternative notion of *online policy*, in the sense of a *strategy* for reacting to dynamic information, based on the current status of the problem.

Online algorithms have first been studied in the field of computer science, for instance, to describe resource management tasks in operating systems, or, the maintenance of data structures to serve dynamic access requests to its elements at low cost, introduced by Sleator and Tarjan (1985) as the *paging problem* and the *list accessing problem*, respectively (see also Albers, 2003; Borodin & El-Yaniv, 1998, for a comprehensive overview in this field). In the following years, the Operations Research community joined the research stream, leading to the introduction of several online variants to the most fundamental optimization problems, such as the *Online Traveling Salesman - and Repairman Problem*, distinct *Online Job Scheduling Problems, Online Bin Packing, Online Matching*, and some of their most natural extensions (see, e.g. Ausiello et al., 2001; Feuerstein & Stougie, 2001; Galambos & Woeginger, 1993; Karp et al., 1990; Lee & Lee, 1985; G. Zhang et al., 2001). These classical problems have the advantage that, despite their static variant possibly being computationally challenging, they have a quite simple structure, that can be described mathematically with just a few types of constraints. As a result, although with considerable mathematical effort, it has been possible to study them *analytically*, providing simple-to-implement - and sometimes even polynomial-time - online algorithms with *performance guarantees*. In some cases, even *best-possible* online policies have been identified under specific criteria. However, the difficulty of the analytical study increases with the addition of every practical problem feature.

Decision problems in realistic real-time environments, as described at the beginning of this chapter, often involve a *substantial* amount of constraints and assumptions. For instance, solution algorithms may have to synchronize the tasks of several - often heterogeneous servers, vehicles, or machines; adhere to capacity-, resource-, and time window constraints; or, deal with complex objective functions. Furthermore, in our digitized world, the pervasive collection of data inherently provides these algorithms with an additional input – an enormous amount of historical data. It is expected that these algorithms will leverage this wealth of information to their advantage. Therefore, the current stream of literature on dynamic optimization problems has largely shifted its focus to the development of *fast* online algorithms for these complex problem structures, which can propose adjusted feasible solutions at the reception of each new input rapidly enough to fulfill the real-time criterion (cf., e.g. Annear et al., 2023; Didden et al., 2024; Schilde et al., 2014; Ulmer et al., 2021). Additionally, new ways to use stochastic assumptions and available data within these approaches are presented. However, due to the lack of analytical performance guarantees and the unavailability of *optimal* algorithms, most studies resort to comparative computational simulations to assess the performance of their methods against simpler benchmark online policies to validate their effectiveness, typically using case- and problem-specific datasets. This approach leaves us with uncertainty about the generalizability of these findings and the potential for further improvement in the proposed algorithms.

This dissertation focuses on two *realistic* and *contemporary* online optimization problems, namely the *Online Order Batching, Sequencing, and Picker Routing Problem (OOBSRP)*, which is of high practical relevance in e-commerce warehouses, and the *Traveling Salesman Problem with a Truck and a Drone under Incomplete Information (TSP-DI)*, which describes a crucial operation in modern disaster relief. Our studies distinguish themselves from the prevailing literature, as we will recommend well-performing online policies based on both *classical- and innovative analytical performance measures and mechanisms*. Our suggestions will be supported by extensive computational studies. Real-time planning often goes hand-in-hand with automation and robotics. Therefore, robots will be considered an aspect of both studied online problems in this dissertation. For instance, in TSP-DI, our analysis will point out the benefits of embedding drone technology in a previously underestimated role as a surveillance tool for knowledge acquisition. The new type of recommended policies leveraging the power of robots are often associated with some *practical* challenges, for example, stemming from their limited battery capacity. This particular challenge is addressed in a dedicated chapter with an algorithmic focus, which is also of independent significance.

The rest of this chapter is organized as follows: Section 1.1 provides the reader with a short literature overview and some necessary background for embedding the problems and methods at the heart of this dissertation into the current landscape of online optimization. Section 1.2 proceeds with the list of papers that contribute the substance of this thesis, elaborating on the author's contribution to these papers and summarizing the main results.

1.1 A glimpse on online optimization

We first categorize online optimization problems in the context of the applications they occur and informally introduce the OOBSRP and the TSP-DI within these classifications (see Section 1.1.1). Then, Section 1.1.2 briefly summarizes some of the main algorithmic approaches found in dynamic optimization, and Section 1.1.3 elaborates on different ways of evaluating the performance of these approaches. Finally, Section 1.1.4 discusses the usage of robots in online problems.

1.1.1 Problem classification and applications

Application fields of dynamic optimization are extremely diverse, and dynamic information may affect planning decisions of different nature, such as routing-, -scheduling-, resource allocation- or packing decisions. Besides that, problems may differ concerning their *sources of dynamism and uncertainty (SoD)* (cf. Ouelhadj & Petrovic, 2009; Pillac et al., 2013; Psaraftis, 2016). This dissertation centers around two online problems of very distinct application areas and SoDs.

In the *Online Order Batching, Sequencing, and Picker Routing Problem (OOBSRP)* studied in Chapters 2 and 3, customer orders must be retrieved by a picker from the inventory of a warehouse, and be brought to the depot for packing. In the era of e-commerce, the orders are placed through the retailer's website and thus arrive dynamically over time. To increase efficiency and customer satisfaction, they should be accommodated *immediately* upon their arrival in the current picking plan, which classifies the OOBSRP's SoD as *dynamic requests*. The uncertainty relates to the orders' arrival times and the associated picking locations. As it is custom in practice, several orders may be picked *simultaneously* in one load of the picker's cart, forming a so-called *batch*. Thus the decisions regarding the *grouping* orders into batches, the *scheduling* of these batches, and the picker *routing* through the warehouse for the orders' retrieval, are altered dynamically in OOBSRP. In this sense, the problem is a non-trivial blend of the *Online Traveling Salesman Problem (OTSP)* (Ascheuer et al., 2000; Ausiello et al., 2001), where the cities to be visited are announced dynamically, and the *Online Batching Problem* (Tian et al., 2012; G. Zhang et al., 2001), where jobs are grouped and scheduled dynamically on a batch-processing machine.

Dynamic requests is a frequent SoD that occurs also in other real-life routing problems, for example in taxi services, delivery- or pickup operations like courier or food ordering, and medical transportation, where the requests reach the servers sequentially as they are in the field (cf. e.g., D. Bertsimas et al., 2019; Ulmer et al., 2021). Similarly, in machine scheduling problems, manufacturing jobs, including the information on their product-specific production requirements, may reveal themselves on the fly (cf. e.g., Hall et al., 1998; Weibo Ren & Guan, 2022). The same SoD appears in general variants of online knapsack or revenue management problems, with applications for instance in online booking, advertising, or cloud computing, where an online planner must decide on the acceptance of dynamically incoming requests characterized by their consumption of a limited resource and the associated reward (cf. e.g., Ball & Queyranne, 2009; Kleywegt & Papastavrou, 1998).

The second online problem of this dissertation, the *Traveling Salesman Problem with a Truck and a Drone under Incomplete Information (TSP-DI)* considered in Chapter 4, is a routing problem of distinct SoD. Specifically, it refers to the delivery of relief supplies to a given set of destinations in the aftermath of a disaster. The latter has caused unknown road damages, which are completely impassable by the delivery truck, and which are detected on the fly, as they are approached by one of the delivery vehicles. Thus, the SoD that defines the online problem is *dynamic network availability*. The deliveries are performed in tandem with a drone, which is launched- and retrieved by the truck at dedicated (optional) *Steiner* locations. In that sense, the TSP-DI is a non-trivial extension of the Steiner traveling salesman problem with online edge blockages proposed by J. Zhang et al. (2015).

Dynamic network availability is closely related to the SoD of *dynamic travel times*, which frequently occurs in traffic management- and vehicle routing problems. There, the traversing times of arcs are uncertain, for instance, due to traffic congestion, or, again, road damages, and are revealed only at their traversal (see, e.g., Mills et al., 2018; Schilde et al., 2014; Vodopivec & Miller-Hooks, 2017). Other widespread SoDs are *dynamic service times*, *-resource demand*, *- resource capacity or server availability* (cf. e.g., Didden et al., 2024; Gökalp et al., 2023; Goodson et al., 2016; J.-Q. Li et al., 2009). As we will see in the following, the SoD has a profound impact on both the solution and analysis of an online problem. Considering for example a dynamic traveling salesman problem, introducing a different or additional SoD can drastically change the criteria for what constitutes a good online algorithm: a policy that is near-optimal with one SoD may become highly suboptimal when faced with another SoD.

Another important characterization that appears in the taxonomy of dynamic problems is, whether they are *dynamic deterministic* or *dynamic stochastic* (cf. Pillac et al., 2013). In the first case, there is no probabilistic information on future input, because either historical data is unavailable, or, it cannot be assumed that the input follows a specific stochastic structure. Those are reasonable assumptions when a novel real-time operation is studied in a new environment, or in the unpredictable effects of a natural disaster, see Chapter 4. In the second case, planners have exploitable information either in the form of probability distributions for dynamic events or through extensive historical data at their disposal. In reality, this applies to OOBSRP in the case of an established e-commerce provider. In Chapter 2 we analyze the performance of an online policy that does *not* make use of the available data. Then, in Chapter 3, we qualify the potentials of using available stochastic information within so-called *anticipatory* online policies, introduced in the next Section 1.1.2.

1.1.2 Policy design and classical algorithmic approaches

In the following, we will briefly review some general *algorithmic schemes* commonly found in the online optimization literature, with particular emphasis on the *reoptimization* policy, which will be revisited repeatedly throughout the subsequent chapters. We aim to distinguish some of its variants, understand its limitations, and introduce some derivatives and alternative schemes designed to overcome these limitations.

In the case of dynamic deterministic (DD) problems, the most straightforward policies are *myopic*. This means that they are simply *reactive* to dynamic input, with *no foresight on* possible *future events*. Especially in the cases where dynamic decisions are irrevocable, not intricate, or *explicitly* apply to only one aspect of the solution, for instance when deciding on the acceptance or the allocation of resources to a request immediately at its arrival (see e.g., Jalota et al., 2023; Kalyanasundaram & Pruhs, 1993), myopic decision making is referred to as the *Greedy* policy. In other more thorough cases, we talk about the *reoptimization (Reopt)* policy, which repeatedly resolves the *(static) offline problem* that results from the current status of the problem, with the currently known information (cf. Chapters 2 and 4). Even this simple concept involves important considerations, some of which are *when*, *what* and *how* to reoptimize. The three questions are implicitely addressed for *Reopt* in OOBSRP in Chapter 3.

In some problems, like the TSP-DI, *Reopt* may be inherently *event-driven*, as the encounter of a blocked road necessitates the computation of an alternate route (see Chapter 4). In other areas, there is an ongoing discussion on *when* to reoptimize, whether it should be *periodically* or *event-driven* (e.g., at the reception of each new information) (see, e.g., Baykasoğlu et al., 2020; Ninikas & Minis, 2014). In the context of OOBSRP, some authors recommend applying state-of-the-art solution approaches for the *offline* problem in *waves* to orders that have accumulated over time (Löffler et al., 2022; Schiffer et al., 2022), and others have studied analytically the best-possible *time windows* to trigger periodical reoptimization in some simplified systems (Le-Duc & De Koster, 2007; Van Nieuwenhuyse & De Koster, 2009). In Chapter 3, we will discuss the possible loss of potential by periodical- compared to event-driven *Reopt* in OOBSRP.

The question of *what* to reoptimize may refer to narrowing the solution space for *Reopt.* Depending on the system's scale, *local reoptimization* at each dynamic event may suffice. For instance, Didden et al. (2024) highlight that full rescheduling in large-scale manufacturing plants, with countless job-to-machine assignments, is inefficient. Instead, they propose *partial rescheduling* focused on selected areas of the shop floor, which performs well compared to complete rescheduling. A similar direction was taken by D. Bertsimas et al. (2019), who tackled a large-scale online taxi-routing problem in New York City by limiting *Reopt* to a few likely optimal solutions by prohibiting, for example, that a taxi drives empty to a far-away customer when demand is high, making their *Reopt* more efficient and tractable. Another interpretation of *what* to reoptimize may concern the subset of the decisions where the reoptimization takes effect. For instance, in the area of dynamic vehicle routing problems (VRPs) Ichoua et al. (2000) advocate for the benefits of allowing *Reopt* to *divert* a vehicle from its route *between* two customer locations, which is restricted in most other studies. In job scheduling, the discussion about the algorithmic benefits of *preemption* belongs into this category. Turning to OOBSRP, very similarly, most studies on online batching and picker routing do not allow modifications to the currently picked batch when a new order arrives. Only Giannikas et al. (2017) introduced this possibility under the name of *interventionist-* (reoptimization) policies, which distinguish themselves from *non-interventionist* policies who allow modifications only when the picker is at the depot. Interestingly, our research in Chapter 2 demonstrates that, from an *asymptotical* perspective, the interventionist *Reopt* has *no* advantage compared to the non-interventionist *Reopt* in OOBSRP. Later, in Chapter 3, we show that from a more practical point of view, on a finite horizon, the opposite is the case.

Finally, the question of *how* to reoptimize is strictly shaped by the complexity of the associated *offline* problem, for which finding an *optimal* solution in real-time is often not possible in practical problems. As a consequence, *heuristic reoptimization* is applied in practice and academia, which reaches from simple policies like priority *dispatching rules* in job scheduling (cf. Ouelhadj & Petrovic, 2009) or greedy *insertion* of requests (Bent & Van Hentenryck, 2004), to sophisticated *metaheuristics*, which are the most widespread in dynamic optimization literature (see e.g., surveys of Ojeda Rios et al., 2021; Ouelhadj & Petrovic, 2009; J. Zhang & Woensel, 2023, for an overview). Metaheuristics have the advantage that they enable *continuous reoptimization* (Gendreau et al., 1999). Furthermore, they are practical as the previously planned solution prior to a dynamic event might render straightforwardly a good initial solution for an improvement procedure incorporating the new information. Such a metaheuristic improvement procedure is for example developed in Chapter 5 of this thesis. In general, D. Bertsimas et al. (2019) and J. Yang et al. (2004) have demonstrated in the case of dynamic routing, that the efficiency of *Reopt* significantly relies on the quality of the embedded offline algorithm. In Chapter 3, we observe the same for OOBSRP, when comparing simple *S-shape* routing policies and *First-Come-First-Serve* batching policies to *optimal* batching and routing in *Reopt*.

Myopic policies such as *Reopt* were frequently *advised against* due to consequential mistakes or blamed of bringing instability and disruption to the dynamic system (Bent & Van Hentenryck, 2004; Ouelhadj & Petrovic, 2009; Xu et al., 2009). In many online environments, researchers have recommended policies which select solutions based on the *similarity* with other solutions instead. A typical approach to generate better predictive-reactive schedules in production is to re-plan at each dynamic event considering *simultaneously* shop efficiency (e.g. in terms of the makespan) and deviation from the original schedule (e.g., from original job starting times or the original production sequence) for more stability (see Ouelhadj & Petrovic, 2009, for an overview). For the dynamic deterministic VRP with time windows, Bent and Van Hentenryck (2004) introduced the *Multiple Plan Approach (MPA)* which significantly improved the greedy approach in terms of serviced customers by maintaining not only one plan but a *set of plans* during the dynamic routing operation, and currently following the *distinguished plan* with the highest value of the *consenus function*, which measures the similarity to other plans in the set, also denoted as *the least commitment strategy*.

In online VRPs with dynamic requests, some worst-case outcomes could be avoided with so-called *Plan-at-Home (PAH)* algorithms, by replanning routes only at the depot and *disregarding* dynamic requests depending on the relative request location and the current server position to the depot (Ausiello et al., 2001; Jaillet & Wagner, 2008b). Furthermore, *rule-based* policies, like threshold-algorithms have proven effective in online matching and resource allocation problems (cf. Pavone et al., 2022). Some authors proposed rule-based algorithms also in OOBSRP (J. Zhang et al., 2016, 2017). Other simple anticipative mechanisms have proven effective in the area of online routing with dynamic customer requests, such as *waiting strategies* or *relocation* of vehicles to *strategic locations* (D. J. Bertsimas & van Ryzin, 1993; Branke et al., 2005; Mitrović-Minić & Laporte, 2004; Moretti Branchini et al., 2009). *Strategic Waiting* is also a trending but controversial topic in online order picking (Gil-Borrás et al., 2024; Pardo et al., 2024), which will be thoroughly discussed in Chapter 3, while *strategic relocations* have been implemented for the first time in OOBSRP in the research of this chapter. All the above non-myopic, *anticipatory* algorithmic strategies can be applied in the case of *dynamic deterministic* problems without the availability of accurate probability distributions.

In *dynamic stochastic* problems, more sophisticated algorithms can leverage the available stochastic information. For instance, the *Multiple Scenario Approach (MSA)* introduced by Bent and Van Hentenryck (2004) as an extension of the MPA is amongst the most widespread algorithmic schemes in routing with dynamic customer requests. The MSA samples the given probability distributions to create possible future requests that are added to the known requests in *scenarios*, for which routing plans are created and then projected to the real requests by removing the artificial customers from the plans. Another approach is to model the problem as a Markov decision process, and, to overcome the *curse of dimensionality*, solve it heuristically, e.g. with *value function approximation methods* or *deep reinforcement learning (DRL)* (Bono et al., 2021; Ulmer et al., 2018). We refer to Soeffker et al. (2022) for a comprehensive introduction to these methods. Similarly to MSA, D'Haen et al. (2023) proposed an anticipatory approach for online warehousing, where they integrated dummy orders generated from forecasts in the picking schedule. Cals et al. (2021) and Shelke et al. (2021) used DRL in online order picking. In Chapters 2 and 3 we validate the need and potential of such advanced anticipation mechanisms for OOBSRP.

1.1.3 Performance evaluation of online policies

Static (or *offline*) optimization problems, where the complete (possibly stochastic) problem information is available a priori, allow for a unique definition of an *optimal* solution. In this field, a whole number of researchers have specialized in the development of *exact* solution approaches dedicated to solving problem instances even for NP-hard problems of ever-increasing size to optimality. In the design of heuristic approaches, it is regarded as good practice to validate the computed solutions through their *gap to optimality* on small-sized instances. In the field of *dynamic* (or *online*) optimization problems, this practice has *not* achieved broad adoption, owing to a lack of consensus and challenges we will outline below. A key objective of this dissertation is to address these difficulties for two practical problems at focus. Without loss of generality, we focus the following discussions on a *minimization* objective.

In the implementation phase of an online policy, *optimal* decisions are not clearly defined, since every new information changes the cost of the suggested solution (Karp, 1992). In *dynamic stochastic (DS)* problems, a classical definition of an *optimal online policy*, adopted from the *static stochastic* case, is the one which minimizes the *expected* objective value in every decision. However, this approach has two obvious disadvantages. First, for most practical problems the computation is intractable due to the *curse of dimensionality* (c.f. Powell, 2011). Second, the *exact* distribution is usually unknown, and there exists no equivalent for *dynamic deterministic (DD)* problems.

In their seminal work, Sleator and Tarjan (1985) introduced the *complete information optimum (CIOPT)* as a benchmark, representing the *best-possible* result provided by an oblivious algorithm, with perfect information on the complete input, including the dynamic elements. Thereby they set the foundation of the for *classical* online problems widely adopted *competitive analysis*. *CIOPT*, which may also be interpreted as a *lower bound* for the performance of *any anticipatory* online algorithm, will play a crucial role in the remainder of this thesis. Tailored to the *DD* case, and similarly to the concepts of *approximation ratios* in *static* optimization (cf. Williamson & Shmoys, 2011), an online algorithm ALG is denoted *asymptotically α-competitive*, if there exists a constant $const$, such that for *all* finite input sequences I:

$$ALG(I) \leq \alpha \cdot CIOPT(I) + const \tag{1.1}$$

The *asymptotic competitive ratio* is then defined as the *infimum* over all constants α, such that ALG is asymptotically α-competitive. When the additive constant $const$ is equal to zero, we say that ALG is (*strictly-*) *α-competitive*. We refer to Chapters 2 and 4 for a detailed discussion (see also Borodin & El-Yaniv, 1998, for a comprehensive introduction to the topic).

The competitive ratios (CRs) of the *same* or similar policies may *differ* significantly depending on the nature, and assumptions of the online problem at hand. No example illustrates this as clearly as the *myopic Greedy* and *Reopt* policies. Consider, for instance, the classical *K-Server Problem* (Manasse et al., 1988), for which a sequence of requests arrive dynamically in a metric space and require one out of k available servers to move to the request location. The *Greedy* algorithm, which always moves the closest server to the request, has *no bounded* CR, while online policies exist with a *linear* ratio in k (cf., e.g., Koutsoupias & Papadimitriou, 1995, for a policy with a CR of $(2k+1)$). The *Reopt* policy similarly performs poorly in the worst case for the online shortest path problem with dynamically revealed edge blockages, known as the *Canadian Traveler Problem* (CTP). In this case, the CR grows *exponentially* with the number of blockages k, specifically reaching 2^{k+1}, whereas others have proposed *best-possible* algorithms with a CR of $2k+1$ (Xu et al., 2009). In other cases, *Greedy* and *Reopt* policies attain CRs *close* to the *lower bounds* for the *best possible* ratios of *deterministic* online algorithms (i.e., those who do not use *randomization*). For instance, in online multiprocessor scheduling with m identical machines, scheduling each dynamically arriving job on the machine that currently has minimum load achieves a CR of $2 - \frac{1}{m}$, which is *best-possible* in the case of $m = 2$ and $m = 3$ machines (Graham, 1969; Karp, 1992). This performance is very close to the proven lower bounds, which exceed 1.8 for a large number of machines (Albers, 1999; Rudin, 2001). Similarly, for the OTSP, *Reopt* is strictly 2.5-competitive, while the *best-possible* PAH policy (see Section 1.1.2) is only slightly better with a ratio of 2 (Ausiello et al., 2001). Note that OTSP and CTP showing such different performance of *Reopt* are both *single-server* dynamic *routing* problems, differing merely in their SoD. We will conduct competitive analysis for *Reopt* in Chapters 2 and 4, and our results will reveal a similar significant disparity in performance between the problems OOBSRP and TSP-DI.

By its very nature, *competitive analysis* serves as a metric for the *worst-case performance* of an online algorithm. Specifically, by definition, it computes the *policy's regret in the worst-case scenario*. Since risk-averse optimization is appropriate and desirable in a disaster situation (M. Yang et al., 2024; Zhong et al., 2020), it is reasonable focus our attention on this metric for TSP-DI in Chapter 4.

In other situations, the classical competitive analysis can provide valuable performance guarantees, but it was frequently criticized as too conservative and unrealistic. For instance, Koutsoupias and Papadimitriou (2000) stated that: ' [...], in the face of the devastating comparison against an all-powerful offline algorithm, a wide range of online algorithms (good, bad, and mediocre) fare equally badly; the competitive ratio is thus not very informative and fails to discriminate or to suggest good approaches.' To narrow the power gap between the oblivious algorithm generating CIOPT and the online policies under investigation, some authors have suggested enhancing the online policy with artificial advantages during the competitive analysis, e.g. through *resource augmentation* (Sleator & Tarjan, 1985), or *advanced information* (Allulli & Laura, 2005; Jaillet & Wagner, 2008b). Recently, Hwang et al. (2021) proposed a model that balances the protection of online decision-making from worst-case outcomes with the realistic assumption of stochastic regularities in the data, which online policies can exploit. They apply competitive analysis to a *mixed* model for a two-fare online resource allocation problem, assuming part of the input follows a *stochastic* pattern derived from past requests, while the rest can appear in its least favorable outcome. Other authors relaxed the assumption that *nothing* is known about the probability distribution of dynamic events (as in the DD settings) - without resorting to the often equally unrealistic assumption that *all* is known about it (as in the DS setting). In that sense, Koutsoupias and Papadimitriou (2000) assumed that the *actual* distribution of the inputs is a member of a *known class* of possible distributions Γ. The proposed *diffuse* competitive ratio then describes the worst ratio between the *expected* performance of the online policy and the *expected* CIOPT under a fixed but arbitrary distribution in Γ. Similarly, Jaillet and Wagner (2008b, 2010) imposed some very *general* stochastic assumptions on the problem data, by limiting for instance the *stochastic nature* of the arrival time sequence of the requests in online VRP, (by imposing either the *order statistics property* or a *Poisson process* model for the latter) without specifying distributional parameters like the *expected rate*. As it is reasonable to assume some *stochastic patterns* for the order placement in e-commerce, we adopt the same analysis for OOBSRP, in Chapter 2.

Analytical guarantees, as mentioned above, are often difficult to compute for *realistic* problems, as the complexity of the study increases with each additional problem feature. Therefore, many authors to rely on *simulations* in their performance analysis. However, they typically use other online policies as the sole benchmark, as discussed in Chapter 3. In our studies of Chapters 2–4, we also conducted simulations to assess the *average* performance of the investigated policies, thereby using, however, *Complete-Information Optimal policy Solutions* (CIOSs) as a benchmark to obtain a more reliable understanding of the policies' performance compared to *optimal* decision making on small instances, in line with established practices in static optimization.

1.1.4 Robotic applications in dynamic environments

In real-world applications, the successful implementation of online optimization is closely tied to the integration of automation, technology, and robotics. One reason is that robots are more *responsive* than humans and can easily adapt to frequent changes in policy solutions consequential to dynamic events. For instance, in OOBSRP, the introduction of autonomous mobile picking carts (cf. Azadeh et al., 2019; Löffler et al., 2022, 2023) may ultimately pave the way for *interventionist* picking policies (cf. Section 1.1.2). In the scenario described by Löffler et al. (2022), pickers follow a robotic cart, which can effortlessly guide them through spontaneous changes in the picking route caused by interventions. As such, we incorporate this technology in our study of Chapter 2.

In some online environments, robots may push the limits of what can be achieved with dynamic policies in the non-robotized counterpart of the problem. One impressive example is the mobility on demand system studied by Pavone et al. (2012), which considers autonomous vehicles. Traditional car or bike-sharing services face difficulties in handling the dynamic demand balancing problem, which occur, sometimes in an unpredictable manner, when some origins or destinations are more popular than others at particular times of the day. Associated policies, including for instance, dynamic trip pricing and operator-based relocation, sometimes inevitably let vehicles pile up at some stations, allow depletion at others, or, perform an increased number of unnecessary expensive rebalancing trips (cf. Eliyan & Kerbache, 2024, for an overview). With empty vehicles driving autonomously between the stations, Pavone et al. (2012) were able to find dynamic rebalancing policies that keep the system *stable*.

In delivery applications, authors have demonstrated the added efficiency and profitability from combined truck and drone fleets in the case of static- (cf. Moshref-Javadi & Winkenbach, 2021; Otto, Agatz, et al., 2018; Srinivas et al., 2022, for an overview), and also dynamic- environments (Ulmer & Thomas, 2018). However, our study of Chapter 4 will show that drones can push the *algorithmic limits* for the TSP-DI, when using it creatively and not just as a delivery assistant for the truck. Specifically, when leveraging its speed and ability to acquire knowledge on the road status from the air in a specialized *surveillance role*, worst-case scenarios (competitive ratios) can be significantly improved compared to policies that do not make use of this technology.

However, the usage of robots comes at the cost of adding challenging aspects to the planning problem, such as stochastic breakdowns or the need for *replenishment schedules* (cf. Chapter 5).

1.2 The contribution of this dissertation

1.2.1 List of papers and author's contribution

The following chapters of the dissertation are based on four co-authored papers:

- Catherine Lorenz [a] [b], Alena Otto [a], Michel Gendreau [c] (2024). Picking Operations in Warehouses with Dynamically Arriving Orders: How Good is Reoptimization? *Under review.*
  Contribution: Leading.

- Catherine Lorenz [a] [b], Alena Otto [a], Michel Gendreau [c] (2024). On picking operations in e-commerce warehouses: Insights from the complete-information counterpart. *Working paper.*
  Contribution: Leading.

- Alena Otto [a], Bruce Golden[d], Catherine Lorenz [a] [b], Yuchen Luo [e], Erwin Pesch [f], Luis Aurelio Rocha [a] (2024). On delivery policies for a truck-and-drone tandem in disaster relief. *IISE Transactions,* 1–17, DOI: 10.1080/24725854.2024.2410353
  Contribution: Considerable.

- Catherine Lorenz [a] [b], Nicola Mimmo [b], Alena Otto [a], Daniele Vigo [b] (2024). Very large-scale neighborhood search for drone routing with energy replenishment. *Under review.*
  Contribution: Leading.

The published or submitted versions of the papers may differ slightly from those presented in this dissertation. While the core content remains unchanged, minor adjustments in contributions or writing may have occurred during the revision process. The submission and publication statuses of the listed papers reflect their state as of the dissertation's publication date.

Each paper listed above indicates the contribution role of the dissertation's author. Two distinct roles were assumed, depending on the tasks undertaken, as defined by the Contributor Roles Taxonomy (CRediT) (cf. National Information Standards Organization, 2022), described as follows:

- *Leading*: Main responsible for the conceptualization, formal analysis, investigation, methodology, software, validation, data curation, writing – original draft, writing – review and editing.

- *Considerable*: Participation in formal analysis (mathematical proofs), investigation (literature review and development of hybrid solution approach), validation, writing – original draft, writing – review and editing.

Section 1.2.2 summarizes the research contributions of each of the four papers and draws the relations between them.

[a]Chair of Management Science / Operations and Supply Chain Management, University of Passau, Passau, Germany

[b]Department of Electrical, Electronic and Information Engineering and CIRI-ICT, University of Bologna, Viale del Risorgimento, 2 - 40136, Bologna, Italy

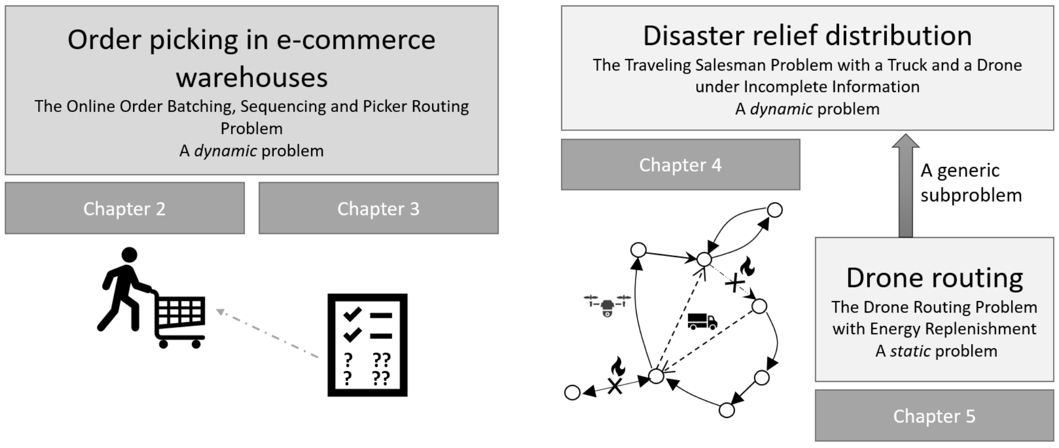
[c]Department of Mathematics and Industrial Engineering and CIRRELT, Polytechnique Montréal, Montréal, Canada

[d]Robert H. Smith School of Business, University of Maryland, College Park, U.S.

[e]Department of Mathematics, University of Maryland, College Park, U.S.

[f]Chair of Management Information Science, University of Siegen, Siegen, Germany

Figure 1.1: The optimization problems considered in this thesis


Order picking in e-commerce warehouses
The Online Order Batching, Sequencing and Picker Routing Problem
A *dynamic* problem
Chapter 2
Chapter 3
? ??
? ??
Disaster relief distribution
The Traveling Salesman Problem with a Truck and a Drone under Incomplete Information
A *dynamic* problem
Chapter 4
A generic subproblem
Drone routing
The Drone Routing Problem with Energy Replenishment
A *static* problem
Chapter 5

1.2.2 Research contribution and outline

This thesis explores selected topics in online optimization within two distinct application areas, as illustrated in Figure 1.1. Chapters 2 and 3, examine dynamic order-picking operations in e-commerce warehouses, while Chapter 4 focuses on collaborative truck-and-drone delivery operations following a disaster, addressing incomplete information regarding road conditions. Chapter 5 complements the previous one, as it focuses on a fundamental subproblem related to the usage of drones in practical (online) applications: the integration of energy replenishments into the drone's route, necessitated by its limited battery capacity. We now provide a summary and outlook of the following chapters. Table 1.1 provides a *compact overview* of the main results and contributions.

Chapter 2: Picking Operations in Warehouses with Dynamically Arriving Orders: How Good is Reoptimization?.
In warehousing logistics, a central component, which accounts for up to 60-70% of the warehouse's operating cost (cf. Marchet et al., 2015) is the planning of the order picking task. In fast-paced e-commerce, incoming orders must be integrated dynamically into the picking process, as a high reactivity is an important key to customer satisfaction. In this chapter, we formally introduce the *Online Order Batching, Sequencing, and Picker Routing Problem (OOBSRP)* (see also Section 1.1.1) in two variations: with manual pushcarts and robotic picking carts. When we initiated this research, analytical results in the context of online order picking were largely confined to examining system properties and queuing behavior of first-come-first-serve picking (Le-Duc & De Koster, 2007; Van Nieuwenhuyse & De Koster, 2009) on one hand, and the competitive analysis of Henn (2012) for a subproblem of OOBSRP, on the other hand. Although several authors have developed sophisticated rule-based, anticipatory- or learning-based algorithms, or, have embedded fast metaheuristics in reoptimization frameworks for OOBSRP and its subproblems (cf. Pardo et al., 2024, for an overview), the performance validation of these algorithms mainly relied on comparative computational studies against simpler benchmark policies, leaving us in the dark about the potential for improving these online algorithms, and their gap to an optimal policy.

This chapter makes a significant contribution by providing, for the first time, comprehensive *performance guarantees* for an online algorithm applied to the two studied OOBSRP variants, through an examination of its gap to CIOPT (see Section 1.1.3), in

Table 1.1: Focus and highlights of the papers in this thesis

Paper (short title)	Focus of paper	Methodology	Main contributions
Efficacy of *Reopt* for dynamic order picking (Chapter 2)	Analytical	Competitive- , Asymptotic- , & Stochastic Analyses; Computational study	- Statement and proof of Reopt's almost sure asymptotic optimality under very general stochastic assumptions for OOBSRP - Lower- and upper bounds for Reopt's asymptotic- and overall worst-case performance - Validation of Reopt's near-optimal performance in experimental study across broad range of warehouse configurations
Picking in e-commerce warehouses: insights from CIOS (Chapter 3)	Managerial	Pattern analysis; Algorithmic recommendations; Exact approach	- Dynamic programming formulations for OOBSRP under perfect information to get CIOS - Detection of decision patterns in CIOSs and actionable advice to enhance online picking with simpler algorithms; e.g. rejection of waiting - Quantification of potential of advanced anticipation mechanisms in different warehouse settings
Truck-and-drone delivery policies in disaster relief (Chapter 4)	Analytical	Competitive Analysis; Computational study	- Statement of exponential growth of Reopt's competitive ratio in TSP-DI in the number of network damages k - Detection of alternate policy with drone surveillance (SF) with a competitive ratio linear in k and superior to any non-surveillance policy - Design of hybrid policy which combines *Reopt* and SF and excels in average- and worst-case performance in experiments
VLNS for drone routing with replenishment (Chapter 5)	Algorithmic	Time complexity analysis; Metaheuristic approach; Exact approach	- Proposition of a novel very large-scale neighborhood (BS-R) of exponential size for DRP-E - Development of a pseudo-polynomial search procedure (VLNS), that searches BS-R entirely - Flexible adjustment of VLNS to various DRP-E variants and and algorithmic schemes - Improvement of state-of-the-art heuristics through VLNS-based approaches in experiments - Optimal solutions in some settings with up to 149 locations through exact offshoot of VLNS

specialized *probabilistic asymptotic analyses*, *asymptotic- and strict competitive analyses*, and a *computational study* encompassing a wide range of warehouse configurations. The chosen policy for this study is the widely adopted and straightforward *immediate reoptimization (Reopt)* in an interventionist and non-interventionist variation (c.f. Section 1.1.2). The performance of this purely myopic policy sheds light on the need and potential of anticipatory online algorithms, which incorporate, for instance, waiting strategies and learning.

The main results of this chapter include the statement and proof of Reopt's *almost sure asymptotic optimality* for all studied variants, under very general stochastic assumptions on the input instances of OOBSRP, which are unknown and unused by Reopt. Furthermore, we demonstrate that *Reopt* is asymptotically 2-competitive for OOBSRP, and that this ratio is tight for interventionist Reopt. The close-to-optimal performance of *Reopt* is confirmed by the computational study. Given these results, *Reopt* provides a tough benchmark for anticipatory policies. For the widespread assumption that customer requests arrive according to a homogeneous Poisson process, our study is, to the best of our knowledge, the first to provide optimality results for a reoptimization policy in the general field of capacitated routing, without imposing an artificial infinite increase of the vehicle's capacity. In this sense, it extends the seminal work of Jaillet and Wagner (2008b).

Chapter 3: On picking operations in e-commerce warehouses: Insights from the complete-information counterpart.

We proceed with the study of the OOBSRP. While the theoretical results from Chapter 2 indicate strong performance of the myopic *Reopt* under the *makespan minimization* objective, which aims to reduce costs through picker time and wages, much of the success of major e-commerce players is attributed to their ability to balance this goal with the often conflicting demand for fast deliveries. To achieve this, they heavily invest in AI-based forecasting to anticipate future customer order characteristics. This chapter shifts from the theoretical, asymptotic, and stochastic assumptions of the previous chapter, adopting a managerial perspective. Specifically, we analyze a *perfect anticipation algorithm* for picking operations that computes *Complete-Information Optimal Solutions (CIOSs)*, under full knowledge of future customer orders, including ordered items and arrival times. This is done for both the *makespan* and *average order turnover* objectives, the latter focusing on delivery speed. The goal is to assess how far simple online picking policies deviate from the optimal solution, identify *decision patterns* in CIOSs that could improve those policies through simple algorithmic enhancements and actionable steps for both objectives, and quantify the remaining improvement potential for advanced anticipation mechanisms in different warehouse settings. This study is the first of its kind, as computing CIOSs requires solving a static version of the OOBSRP with release times (OBSRP-R). No existing exact algorithm for order batching and picker routing can directly address release times, as they disrupt both the polynomial solvability of the routing subproblem (cf. Bock et al., 2024) and the logic underlying current batching methods.

The *algorithmic* contribution of this paper lies in the development of an *exact dynamic programming (DP)* formulation for OBSRP-R with the makespan objective, and a heuristic variant of this DP for the average order turnover objective. The DP can solve instances with 15-21 orders, encompassing 25-48 picking locations. As demonstrated with a *Kruskal Wallis Test*, the associated CIOSs exhibit statistically significant stability at this scale. The *practical* contribution of the Chapter consists of the in-depth analysis of the observed CIOSs, revealing the importance of *anticipatory elements and optimization* in *routing, batch formation*, and *-selection*. Importantly, through the analysis of CIOSs, we could shed light on one of the least understood concepts in the online warehousing literature, namely *strategic waiting*, which deliberately delays picking for a better fit of future orders. Although perceived as highly promising, it was showing mixed results in previous studies.

Indeed, both the makespan and the orders' turnover *can* be simultaneously improved with simple algorithmic enhancements, that have largely been overlooked in the warehousing literature. Specifically, our study promotes the investment in technologies that enable *intervention* (cf. Section 1.1.2), which affects 62-64% of all orders in CIOSs on average and reduces the *average observed gap to CIOS* significantly in all studied policies, (e.g., for *Reopt* reduced by 3.0 and 16.6 percentage points for the makespan and turnover objective, respectively). The *strategic relocation* of the picker to the 'gravity center' of anticipated future orders during idle time, can significantly improve the orders' turnover. However, our study *advises against* strategic waiting, which is *very brief and well-timed* in CIOSs (lasting just 12-70 seconds on average), and thus as demonstrated in our study, results in significant opportunity costs for online algorithms that can only be mitigated by fully accurate order anticipation. In line with the results in Chapter 2, we find that the improvement potential from advanced anticipation is quite limited in OOBSRP, (e.g. after applying optimization and simple enhancements, just 3-4% of improvement potential is left for reducing the picker's time), thus we recommend to focus data science efforts in e-commerce on an upstream planning level.

We turn to the second online problem of the thesis.

Chapter 4: On delivery policies for a truck-and-drone tandem in disaster relief.
In the aftermath of a disaster, the timely provision of relief supplies like water or medication

is of utmost importance. Unfortunately, the deliveries are often hindered by impassable road damages, discovered in a *dynamic* fashion by the delivery trucks at their approach, causing potentially escalating long truck detours. In the presented *Traveling Salesman Problem with a Truck and a Drone under Incomplete Information (TSP-DI)*, a drone, which can surpass the road damages, is assisting with the deliveries, but is limited to carrying only one package at a time and requires periodic meetings with the truck for reload. The conservative *competitive analysis* (cf. Section 1.1.3) of the online delivery operation is highly significant under the unpredictable circumstances of a catastrophe, in which the worst-possible regret should be impeded.

Thus, the contribution of this chapter consists of a comprehensive *parametric* investigation of *competitive ratios* for the online reoptimization policy (Reopt) – widely used for TSP-DI in practice, and several alternative delivery policies. The newly proposed policies use the drone - aside from delivery- innovatively, to acquire knowledge on the network via *surveillance*, cf. Section 1.1.4. Also the *average* performance of these policies is compared in an *extensive computational study*, based on their *observed ratio* to CIOS.

In contrast to the previous Chapters, the results demonstrate the *unfavorable* competitive ratio of *Reopt* for TSP-DI, which grows *exponentially* in the number of damaged edges k. On the other hand, we find that the alternative, seemingly conservative policy SF, which holds back deliveries until complete knowledge on the important roads has been acquired in a prior surveillance tour of the drone, has a *better* competitive ratio, which grows *polynomially* in k. Despite SF dominating *Reopt* in the worst-case, our computational experiments show that *Reopt* performs better *on average*. Thus we design a *hybrid* policy, $\mathrm{S}_{25\%}^{\mathrm{hybrid}}$ which combines the best of *Reopt* and SF, and has state-of-the-art average- *and* worst-case ratios to CIOS.

Chapter 5: Very large-scale neighborhood search for drone routing with energy replenishment.
A major limitation caused by the technicality of the study, is that the previous chapter did not consider the energy limitations of the drone during surveillance in the policies SF and $\mathrm{S}_{25\%}^{\mathrm{hybrid}}$. In practice, the drone must regularly interrupt its (surveillance) operation to swap its battery at one of its designated stationary charging stations or, by meeting with a mobile replenishment station (MRS) in well-selected replenishment locations. In a disaster scenario, the MRS could be either the truck, moving along edges of known status, or a mobile charging robot designed for this purpose (cf. Barrett et al., 2018). The resulting *static subproblem* belongs to the generic class of *drone routing problems with energy replenishment (DRP-E)*, which is characterized by challenging synchronization constraints between vehicles. The existing solution approaches for DRP-E-like problems in the literature mainly focus on route-first-split-second heuristics, which are heavily sub-optimal in some cases, and metaheuristic approaches such as the *adaptive large neighborhood search (ALNS)* which primarily aim to *diversify* the searched solutions. However, the literature lacks a *powerful intensification procedure* around promising DRP-E solutions. In the context of the online policies from Chapter 4, DRP-E is solved *repeatedly* at each encounter of a road damage in the surveillance phase, thus a promising initial solution for the new problem data might be derived easily from the previously planned route. In conclusion, although of *general significance*, the development of an *efficient improvement procedure for static and generic DRP-E* may be particularly beneficial in the context of online TSP-DI.

In this chapter, we propose a *very large-scale neighborhood* which synergetically leverages two large-sized polynomially solvable DRP-E subproblems (SP1 and SP2). The number of feasible solutions in the resulting neighborhood is a multiple of those in SP1 and

SP2, and, thus, *exponential* in the input size of the problem. As a main contribution, we develop a *non-trivial search procedure, VLNS,* which examines this neighboorhood *entirely*, in a computational time that remains *polynomial* in the problem size. VLNS is a two-stage dynamic programming approach, for which the desired trade-off between accuracy and runtime can be flexibly adjusted with just a single parameter. The proposed VLNS can flexibly accommodate many additional problem-specific features, and thus may be applied in a diverse range of applications. Also, VLNS is an improvement tool that may be embedded in any algorithmic scheme, meta- or math-heuristic. We additionally propose a high-performing exact solution method (DP-ILP) for DRP-E, that stems from the structure of VLNS.

Our computational results demonstrate the performance and the generalizability of VLNS; a simple VLNS-based local search achieves small gaps to optimality, and outperforms state-of-the-art heuristics on DRP-E variants from the literature. On the other hand, DP-ILP solves instances with up to 149 locations to optimality in some settings.

Chapter 2

Picking operations in warehouses with dynamically arriving orders: How good is reoptimization?

Catherine Lorenz, Alena Otto, Michel Gendreau

Abstract. E-commerce operations are essentially online, with customer orders arriving dynamically. However, very little is known about the performance of online policies for warehousing with respect to optimality, particularly for order picking and batching operations, which constitute a substantial portion of the total operating costs in warehouses. We aim to close this gap for one of the most prominent dynamic algorithms, namely reoptimization *(Reopt)*, which reoptimizes the current solution each time when a new order arrives. We examine *Reopt* in *the Online Order Batching, Sequencing, and Routing Problem (OOBSRP)*, in both cases when the picker uses either a manual pushcart or a robotic cart. Moreover, we examine the *noninterventionist Reopt* in the case of a manual pushcart, wherein picking instructions are provided exclusively at the depot. We establish *analytical* performance bounds employing *worst-case and probabilistic analyses*. We demonstrate that, under generic stochastic assumptions, *Reopt* is almost surely asymptotically optimal and, notably, we validate its near-optimal performance in computational experiments across a broad range of warehouse settings. These results underscore *Reopt*'s relevance as a method for online warehousing applications.

2.1 Introduction

E-commerce has emerged as an important channel of retail trade, emphasizing fast deliveries and large product assortments, while pressuring for low logistics costs. This makes warehousing an operational challenge and a leading determinant of competitiveness in the e-commerce market. The central component of efficient warehousing is order picking, which accounts for up to 60-70% of the total operating costs (T.-L. Chen et al., 2015; Marchet et al., 2015). Most warehouses are so-called *picker-to-parts* warehouses (Napolitano, 2012; Vanheusden et al., 2023), in which order pickers travel around the warehouse to collect inventory items (see Figure 2.1). In a widespread *pick-while-sort* setup, pickers try to collect several orders simultaneously to save on costs. They use a *manual pushcart* (Figure 2.1a) or a *robotic cart* (Figure 2.1b). Each cart contains several bins, and each bin can accommodate an order. In this paper, we focus on picking operations in picker-to-parts warehouses and examine both technologies, manual pushcarts and robotic carts.

Figure 2.1: An order picker with a cart in a picker-to-parts warehouse

Source: KBS Industrieelektronik GmbH. License: CC BY-SA 3.0

(a) Pushcart

Source: SSI Schäfer

(b) Robotic cart

E-commerce operations are *essentially online*, with customer orders arriving dynamically. However, *very little* is known about the performance of the *online policies* for handling these orders with respect to *optimality*. A prominent online policy is *reoptimization*, which myopically optimizes picking operations for the set of available orders at specific points in time. Moreover, reoptimization can leverage the recent progress in exact algorithms and heuristics developed for the *offline* setting, in which all the customer orders are *known* in advance (cf. Löffler et al., 2023; Schiffer et al., 2022; Valle et al., 2017). Nevertheless, it remains open whether reoptimization makes sense for picking operations in warehouses. Optimization literature provides examples both of good (Hwang & Jaillet, 2018; Jaillet & Wagner, 2008b) and unfortunate (cf. Borodin & El-Yaniv, 1998) performance of reoptimization in online problems.

In this paper, we study reoptimization analytically and experimentally in a generic order picking setting, which is known as *the Online Order Batching, Sequencing, and Routing Problem (OOBSRP)* in the taxonomy of Pardo et al. (2024). In OOBSRP, given the dynamically arriving orders, we have to determine *a) [batching]* which orders to pick together in a single cart tour and *b) [routing]* how to schedule the picking operations and how to route the picker. In the following, Section 2.1.1 reviews the literature, and we state the article's contribution in Section 2.1.2.

2.1.1 Literature review

Online order picking and batching problems have attracted much interest in the literature, see the surveys of Boysen et al. (2019), Y. Li et al. (2022), Pardo et al. (2024), and Van Gils et al. (2019) and Vanheusden et al., 2023. Yet, very little is known about *optimal* policies. The available research can be roughly classified into three threads. The first one analyzes intuitive policies (such as 'first come first served', S-shape routing, etc.). Here, important insights on the system stability and policy parameters are formulated using, in most cases, the tools of the queuing theory (cf. Le-Duc & De Koster, 2007; Schleyer & Gue, 2012; Van Nieuwenhuyse & De Koster, 2009; Yu & De Koster, 2008; Yu & De Koster, 2009). Within this thread, Bukchin et al. (2012) analyze optimal decisions on the waiting time before picking a batch for a significantly simplified problem (e.g., the time to collect a batch does not depend on the locations of the respective items in the warehouse). Secondly, a number of papers design sophisticated online policies based on metaheuristics, AI, or rule-based approaches (e.g., Cals et al., 2021; D'Haen et al., 2023; J. Zhang et al., 2016, 2017). Finally and prominently, a number of authors use reoptimization as the online policy of choice (see Dauod & Won, 2022; Giannikas et al., 2017; Lu et al., 2016).

In all these studies, reoptimization is performed *immediately* at the arrival of a new order. In all the discussed cases, the designed policies are usually benchmarked against simple policies, so that their *performance gap to optimality is generally unknown*.

To the best of our knowledge, only Henn (2012) and Alipour et al. (2020) compare selected policies to optimality. Both papers focus on the worst-case performance using *competitive analysis*. Henn (2012) studies reoptimization in a special case of OOBSRP with a manual pushcart, where the picker routes start and end in the depot and follow a predefined simplified scheme, which defines the problem as the *Online Order Batching Problem (OOBP)* according to the classification of Pardo et al. (2024). In his analysis, reoptimization can only be performed when the picker retrieves a new cart in the depot; it is performed either immediately when some recently arrived orders are available or after some deliberate waiting time. The concept of waiting times seems appealing, because further orders may arrive that make better-matching batches with the available orders. In Henn (2012), the introduction of the waiting time did not improve the results of the algorithm. The paper establishes a *lower bound* of 3/2 for the *strict competitive ratio* for any deterministic online algorithm and proves an *upper bound* of 2 for the *asymptotic competitive ratio* for their reoptimization policy (see definitions in Section 2.3). The tight competitive ratios remain unknown. Alipour et al. (2020) extend the results of Henn (2012) to multiple pickers.

On a more general note, OOBSRP can be described as a nontrivial blend of two well-known optimization problems – *the online traveling salesman problem (online TSP)* (since the picker is routed over a set of locations) and *the online batching problem (OBP)* (since the incoming orders have to be grouped into batches to be processed simultaneously in one cart tour). The seminal work of Ausiello et al. (2001) introduces two versions of online TSP: *the nomadic one (N-TSP)*, for which the route of the online server ends at the last visited location, and *the homing one (H-TSP)*, which requires the server to return to the depot. As we will explain in Section 2.2, N-TSP and H-TSP resemble OOBSRP in the special case of infinite cart capacity in the cases where a robotic cart and a manual pushcart are used for picking, correspondingly. In OBP, the machine can process up to $c \in \mathbb{N}$ jobs simultaneously, and the processing time equals the processing time of the longest job. The objective is to schedule the processing of dynamically arriving non-preemptive jobs on this machine in order to minimize the makespan (cf. G. Zhang et al., 2001). OBP can be interpreted as a special case of OOBSRP with a manual pushcart in a single-aisle warehouse with the depot located at the end of the aisle. Let us denote as *Reopt* the *immediate* reoptimization policy, which reoptimizes each time a new order arrives. The *worst-case* performance of *Reopt* has been estimated as follows: The strict competitive ratio of *Reopt* equals 2.5 for N-TSP (Ausiello et al., 2001), does not exceed 2.5 for H-TSP (Ausiello et al., 2001), and equals 2 for OBP (Liu & Yu, 2000). Moreover, although better policies are possible, *Reopt*'s results in these problems are quite close to the best achievable worst-case results (see Ascheuer et al., 2000; Ausiello et al., 2001; G. Zhang et al., 2001). Further, Ascheuer et al. (2000) proved that the result of *Reopt* in H-TSP is at most twice the complete-information optimum objective value plus a constant additive factor that characterizes the diameter of the studied metric space. In none of the aforementioned studies, *Reopt* was compared empirically to an optimal algorithm.

Only very few papers in the literature perform probabilistic competitive analysis for online routing problems with *capacity restrictions*. The reason is that the service times of the tours formed for each capacitated vehicle by nontrivial policies, such as *Reopt*, usually have a nonzero length and are highly interdependent, which makes a formal analysis very hard. Therefore, most existing studies analyze variants of the 'first come first served' policy to leverage the methods of the queuing theory (cf. e.g., D. J. Bertsimas & van Ryzin, 1993). An exception is a prominent article by Jaillet and Wagner (2008b), who studied *the online TSP with precedence and capacity constraints (OTSP-PC)*. Jaillet and Wagner (2008b) examine a problem extension with several salesmen, propose online algorithms with the best-possible strict competitive ratio for several problem variants, perform competitive analysis under resource augmentation, and, for specific problem variants and under several additional assumptions, perform probabilistic asymptotic analysis of online algorithms for a renewal process of arriving requests and an arrival process with order statistics property. However, OTSP-PC studied by Jaillet and Wagner (2008b) *differs* from OOBSRP as it allows *splitting* of orders (called requests) among several tours, whereas OOBSRP does not. Consequently, once the first item of an order (called *city of a request* in OTSP-PC) is collected, a commitment to collect *all* the remaining items of this order within the current tour emerges in OOBSRP (see Example 1 of Section 2.3.4). This is nontrivial and changes the character of the problem completely. Overall, the idea of a probabilistic asymptotic analysis of *Reopt* as well as the nature of examined arrival processes in this paper were inspired by Jaillet and Wagner (2008b). However, this paper's methodology is *distinct* in several essential ways. For instance, Jaillet and Wagner (2008b) rely on a number of additional assumptions in their probabilistic analysis that do not apply to OOBSRP and are not used in this paper, such as Euclidean space, one item per order, and – in case of a renewal process for order arrivals – infinitely growing cart capacity (called server capacity) with the number of orders n. This radically changes the mechanics of the proofs and, as in Theorem 3, requires new proof techniques.

To sum up, the optimality analysis of reoptimization for OOBSRP, which combines the features of routing (online TSP) and batching (OBP) has not been performed so far. Although the results of *Reopt* in online TSP and OBP are promising, it is absolutely unclear, whether it retains its good performance in the more general case of OOBSRP. Furthermore, this paper is the first to prove almost sure asymptotic optimality of a reoptimization algorithm in case of a fixed server capacity for any capacitated online routing problem, under a renewal process model for arrival times.

2.1.2 Contribution

OOBSRP, which describes picking operations in e-commerce, is a central problem in the warehousing literature. However, we still do not know the performance gap to optimality for *any* OOBSRP policy. We aim to close this gap for one of the most prominent dynamic algorithms, namely immediate reoptimization *(Reopt)*, which reoptimizes the current solution each time a new order arrives. Therefore, we establish *analytical* performance bounds by means of *worst-case and probabilistic analysis*, and *empirically* validate the *almost optimal* performance of *Reopt*, which is indicated by our analytical findings. In particular,

- We analyze OOBSRP for two widespread technologies: manual pushcarts and robotic picking carts. Since in the former case, transmitting instructions to the picker is only possible at the depot in some types of warehouses cf. C.-M. Chen et al., 2010, we also examine a so-called *non-interventionist* version of *Reopt* for manual pushcarts. *Non-interventionist Reopt* does not allow the modification of started batches, therefore reoptimization only takes place when the picker is at the depot.

- Our main result states that *Reopt* is *almost surely asymptotically optimal* under very general stochastic assumptions in all the examined picking systems. Stated differently, in any sufficiently large instance, the result of *Reopt* coincides, with probability one, with the outcome of the optimal policy for this instance and cannot be improved. This is the case when the order arrival times can be modeled as so-called order statistics, or as a Poisson process with sufficiently small rates. In the case of a Poisson process, we also show that *Reopt* achieves a slightly weaker notion of optimality when the order arrival rate is sufficiently large.

- In the subsequent analysis, we drop stochastic assumptions and examine the performance of *Reopt* in *any* large instance as well as in instances of arbitrary sizes. Among others, we prove that the worst-case performance of *Reopt* compared to the optimum (*competitive ratio*) approaches 2 in all studied picking systems, in sufficiently large instances. In other words, no policy can improve upon the results of *Reopt* by more than 50% in such instances.
- Our experiments, which feature typical warehouse settings and order arrival rates, reveal *almost optimal* performance of *Reopt* already in moderate-sized instances.

Our analysis offers significant insights into long-standing discussions within the warehousing literature. The first pertains to the merits of *waiting*, wherein the picker interrupts picking to wait in the depot (or field) for additional orders, in the hope for the arrival of better-matching orders for the next batch (Bukchin et al., 2012; Pardo et al., 2024). Waiting is an integral part of prominent fixed-time-window batching and variable-time-window batching policies (Pardo et al., 2024; Van Nieuwenhuyse & De Koster, 2009). The second involves *anticipation*, which seeks to predict upcoming order properties based on historical data (Ulmer, 2017). In prevalent designs, anticipatory algorithms *reserve* slots in batches for future orders, resulting in the under-utilization of the cart for some time and potentially leaving out well-matching orders. Both waiting and anticipation incur immediate costs for uncertain future benefits. The performance gaps of *Reopt*, which eschews both *waiting* and *anticipation*, establish a clear benchmark for assessing the benefits of these concepts. In scenarios where *Reopt* closely approaches complete-information optimality (which is a strong concept of optimality, cf. Section 2.2), the introduction of waiting or anticipation cannot significantly improve the results and could potentially lead to deterioration.

We emphasize the novelty of the performed probabilistic analysis of *Reopt*. This analysis is one of the few research endeavors to scrutinize optimality gaps of a nontrivial online policy for a variant of the capacitated vehicle routing problem, diverging from the conventional ‘first come first served’ approach, and assuming realistic capacity restrictions (see the discussion in Section 2.1.1).

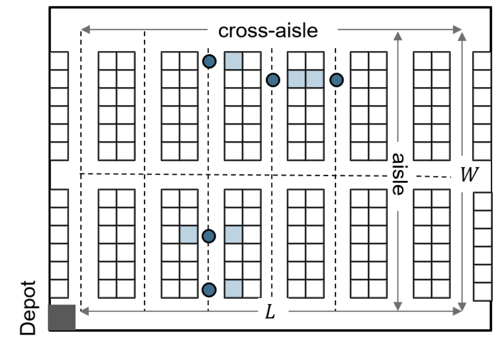
We proceed as follows. Section 2.2 states OOBSRP formally, Section 2.3 presents theoretical analysis of *Reopt* for OOBSRP, and Section 2.4 reports on computational experiments. We conclude with a discussion in Section 2.5.

2.2 Problem description

A typical warehouse (see Figure 2.2) resembles a rectilinear grid of length L and width W, which has $a \geq 1$ equidistantly positioned parallel aisles and $b \geq 2$ equidistantly positioned cross-aisles. The depot is located at some fixed location l_d in the grid. The picker moves in a rectilinear way through aisles and cross-aisles, which results in a particular distance metric $d(\ )$. As explained in Figure 2.2, the picker can access items only from the aisles, in particular, no items are located in cross-aisles.

Orders $(o_1, o_2, \ldots, o_j, \ldots)$ arrive dynamically over time and each order o_j is associated with some number $k(j) \in \mathbb{N}$ of *picking locations* in the warehouse, where the ordered *items* have to be retrieved. Following the hierarchical planning process in warehouses,

Figure 2.2: A typical warehouse layout.



Note. A warehouse of length L and width W with three cross-aisles and seven aisles. Circles mark access points to pick highlighted items.

OOBSRP assumes that the picking location of each ordered item has been already determined, therefore, we use these terms interchangeably. Recall that the cart of the picker consists of $c \in \mathbb{N}$ bins, so that, to save on costs, she collects up to c orders at a time during one cart tour in a so-called *batch*. Orders cannot be split between different bins and each bin may contain only one order at a time. Parameter c is called *batching capacity*.

We examine *picking systems* with two different types of carts: manual pushcarts and robotic carts; we will denote these carts as *pushcart* and *robot*, respectively, in the following. The main difference between them is that the picker has to bring the pushcart to the depot each time after a batch has been collected; whereas the robot can drive there autonomously and the picker can proceed to pick the items ‘in the field’ into the timely arrived new robot.

OOBSRP considers one picking zone of a single picker. Therefore, when we talk about the warehouse layout below, we refer to the layout of a single zone of the warehouse.

We make the following assumptions:

- *W.l.o.g.*, we assume that the picker travels at unit speed, so that we talk about the distances and picker travel times interchangeably. In our theoretical analysis, we set the picking times of retrieving an item from the shelf and placing it in the cart to zero for simplicity. Later on, in the empirical analysis of Section 2.4, we consider non-zero picking times.
- Following the literature on robotic carts (e.g., Löffler et al., 2022; Žulj et al., 2022), we assume that an empty robot is immediately available for the picker after she has completed a batch. Indeed, two robots per picker are usually sufficient to avoid picker waiting (cf. Löffler et al., 2022).

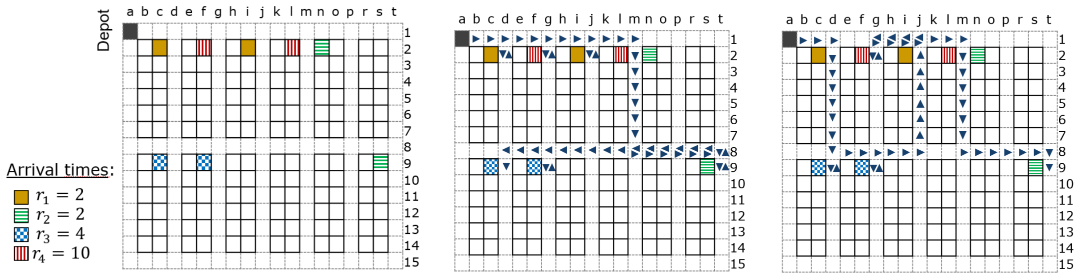
Before stating OOBSRP in Section 2.2.2, we introduce the *offline* problem variant with complete information (OBSRP*) in Section 2.2.1. Figure 2.3 states an illustrative example that we use throughout this section.

2.2.1 Offline problem variant with complete information: OBSRP*

An *instance* I of OBSRP* *with a specific type of cart* has the following input:

- Infrastructure parameters, such as the location of the depot l_d, width W, length L, the number of aisles a and cross-aisles b, as well as the batching capacity c;
- the *sequence of* dynamically arriving *orders* $o = (o_1, ..., o_n)$, where $o_j := \{s_j^1, ..., s_j^{k(j)}\}$ refers to the set of $k(j) \geq 1$ picking locations for the jth order;

Figure 2.3: Illustrative example.



(a) Instance I **(b)** $CIOPT$ route with *robot* **(c)** *Reopt* route with *robot*

Note. Items of all $(n = 4)$ orders are marked in different colors.

- the respective *sequence of arrival times* $r = (r_1, ..., r_n)$, with r_j being the arrival time of o_j and $r_1 \leq ... \leq r_n$. Observe that in OBSRP*, where all the information is known in advance, arrival times can be simply interpreted as *release times* for picking the items of the corresponding orders.

The objective is to minimize the *total completion time* of picking, which describes the picker's working time. In the system with a pushcart, the picker has to return the pushcart to the depot in the end. On the contrary, in the system with a robot, the total completion time describes the time passed until the picker picks the *last item* of the last batch. Given this objective, an optimal solution of OBSRP* for both cases of a pushcart and a robot is uniquely defined by:

- a mutually disjoint partition of the orders into batches: $\{o_1, ..., o_n\} = S_1 \cup S_2 \cup \ldots \cup S_f$, $S_l \cap S_k = \emptyset \ \forall k, l \in \{1, ..., f\}, k \neq l$; each batch contains at most c orders.

- a sequence of these batches π^{batches}, which determines the order in which the batches are processed by the picker.

- for each batch S_l, a permutation of the picking locations of all the included orders, which determines in which order the picker collects the items.

Indeed, in an optimal picking schedule, the picker will pick each item at the earliest possible time (which accounts for the arrival times of the orders) given the sequence of batches π^{batches} and the visiting sequences of picking locations for each batch. We denote the optimal objective value of instance I as *complete information optimum* $CIOPT(I)$ and a respective optimal solution as $CIOPT$.

Table 2.1 illustrates $CIOPT(I)$ for the instance I with $n = 4$ orders provided in Figure 2.3a in the case of a pushcart and a robot, respectively.

Note that in the optimal solutions for both cases, the picker leaves the depot towards the first picking location even before the arrival of the respective order, since the arrival times and picking locations of all the orders are known in advance. Also observe that grid cells in Figure 2.3 have a unit length, e.g., $d(a1, d2) = 4$ as the picker has to traverse four units (cell lengths) between the cell centers of the depot $a1$ and $d2$.

In an optimal solution of OBSRP* with a pushcart, the picker arrives at each picking location after the arrival of the corresponding order. The first batch consists of two orders $(S_1 = \{o_1, o_3\})$ and is completed in $(4 + 7 + 5 + 10 + 10) = 36$ time units, after the picker's return to the depot. The second batch $S_2 = \{o_2, o_4\}$ is picked in $(7 + 19 + 13 + 0 + 13) = 52$ time units. Overall, $CIOPT$ requires $(36 + 52) = 88$ time units.

Table 2.1: Algorithms and their solutions for the illustrative instance I from Figure 2.3

Algorithm	Objective value	Sequence of batches $\pi^{\text{batches}} = (S_1, S_2, \ldots)$	Picker route[ii)] π
Case of a *pushcart*			
CIOPT[i)]	88	$(\{o_1, o_3\}\{o_2, o_4\})$	$(a1, d2, d9, g9, j2, a1, g2, t9, m2, m2, a1)$
Non-interventionist *Reopt*	100	$(\{o_1, o_2\}, \{o_3, o_4\})$	$(a1, d2, j2, t9, m2, a1, d9, g9, m2, g2, a1)$,
Interventionist *Reopt*	90	$(\{o_1, o_4\}\{o_2, o_3\})$	$(a1, \mathbf{c1}, d2, \mathbf{g1}, g2, m2, j2, a1, m2, t9, g9, d9, a1)$
Case of a *robot*			
CIOPT	52	$(\{o_1, o_4\}, \{o_2, o_3\})$	$(a1, d2, g2, j2, m2, m2, t9, g9, d9)$
Interventionist *Reopt*	54	$(\{o_1, o_3\}, \{o_2, o_4\})$	$(a1, \mathbf{c1}, d2, \mathbf{d6}, d9, g9, j2, g2, m2, m2, t9)$

Note. [i)] *CIOPT* refers to an exact algorithm for the complete-information counterpart of instance I. [ii)] The picker positions at reoptimization events of the interventionist *Reopt* are marked in bold.

Observe that in *CIOPT* for OBSRP* with a robot (c.f. Figure 2.3b), the picker waits for the arrival of order o_4 for one time unit at location $g2$ while performing the first batch. Overall, batch $S_1 = \{o_1, o_4\}$ is picked in $(4 + 5 + [1] + 5 + 5) = 20$ time units. After having completed the first batch, the loaded robot returns to the depot autonomously, a new empty robot is immediately available, and the picker continues with the next batch $S_2 = \{o_2, o_3\}$ from her current location $m2$. *CIOPT* requires $(20 + 32) = 52$ time units.

2.2.2 Online problem and the reoptimization policy

In the *online* problem, OOBSRP, neither the number of orders n, nor the arrival times of orders are known ahead of time, and the characteristics of each order o_j are revealed at its arrival r_j. Therefore, online algorithms represent *policies*, which prescribe a decision given the current state and available information. We analyze the online algorithm *Reopt*, which optimizes picking operations based on the available information without any anticipation of future orders.

In traditional warehouses with a pushcart, the picker can receive instructions only at the depot, for example, because of pick-lists that are printed out, so that an already commenced batch cannot be replanned. Therefore, we differentiate between *non-interventionist Reopt* and *interventionist Reopt*. In *non-interventionist Reopt*, reoptimization occurs at each return of the picker to the depot provided newly arrived orders are available. If all the available orders have been completed, the picker simply stays at the depot until further orders arrive. Let reoptimization take place at some time t, then it amounts to solving the *complete-information* problem OBSRP* for the orders, which arrived before t *and* have not been picked so far; obviously, these orders are treated as immediately available.

To the contrary, in the *interventionist Reopt*, reoptimization is performed at each arrival r_j of a new order o_j. Also in this case, the planner solves a variant of OBSRP* for the already arrived, but not yet processed orders; however, she takes the following aspects into account: *i)* if some bins of the cart contain picked items, the respective orders must be completed as part of the *current* batch; *ii)* in the current batch, empty bins are free to be (re)assigned to any available and not yet commenced order, *iii)* the picker route starts from her current position in the warehouse.

To sum up, we examine three picking systems for OOBSRP, each featuring a cart technology and the applied *Reopt* algorithm: One with a pushcart and a noninterventionist *Reopt* policy (*Pcart-N*), one with a pushcart and interventionist *Reopt* (*Pcart*), and one with a robot and interventionist *Reopt* (*Robot*).

Table 2.1 illustrates the objective values $Reopt(I)$ of *Reopt* for the instance I from Figure 2.3 in the three considered picking systems: *Pcart-N*, *Pcart*, and *Robot*. In all three systems, the picker stays at the depot until the first two orders arrive at time $r_1 = r_2 = 2$.

In *Pcart-N*, at the time $t = 2$ of the first (re)optimization, only orders o_1 and o_2 are available and the batch $S_1 = \{o_1, o_2\}$ with the completion time of $(4 + 8 + 16 + 13 + 13) = 54$ is formed. Although new orders arrive at times 4 and 10, no replanning can take place until the picker returns to the depot. The next reoptimization occurs at time $(54 + 2) = 56$ and the remaining orders are assigned to the next batch. Altogether, $Reopt(I) = 56 + 44 = 100$ for *Pcart-N*.

In contrast, in *Pcart* and *Robot*, interventionist *Reopt* reoptimizes the current solution at each arrival of a new order. In both systems, at $r_3 = 4$ the picker is at decision point $c1$, and the planned routes for the batch $S_1 = \{o_1, o_2\}$ are $(a1, d2, m2, t9, j2, a1)$ and $(a1, d2, j2, m2, t9)$ in the case of *Pcart* and *Robot*, respectively. For *Pcart*, the new batch sequence is $(\{o_1\}, \{o_2, o_3\})$, and the new route $(a1, \mathbf{c1}, d2, j2, a1, m2, t9, g9, d9, a1)$. This route is interrupted at the arrival of o_4 at $r_4 = 10$, when the picker is at $g1$, resulting in the final solution of Table 2.1 with $Reopt(I) = 90$. For *Robot*, the batching is first changed to $(\{o_1, o_3\}\{o_2\})$ and the route is changed to $(a1, \mathbf{c1}, d2, d9, g9, j2, m2, t9)$. Order o_4 arrives when the picker is at position $d6$ having picked the first item of o_1. Therefore, at the next reoptimization at time $r_4 = 10$, one bin in the current batch is reserved for order o_1. So, the new batching is $(\{o_1, o_3\}, \{o_2, o_4\})$ and results in $Reopt(I) = 33 + 21 = 54$ for *Robot* (c.f. Figure 2.3c).

2.3 Extended competitive analysis for *Reopt* in OOBSRP

In this section, we perform competitive analysis for *Reopt*. In competitive analysis, the result of some algorithm $ALG(I)$ for instance I is compared to the best possible online policy for this instance. Obviously, the best possible online policy is that of an oracle, which perfectly foresees the arrival time and the composition of each order, and the result of this policy is $CIOPT(I)$ (cf. Fiat & Woeginger, 1998).

We apply three different ratios to highlight the various aspects of OOBSRP performance. We start in Section 2.3.2 with the *almost surely (a.s.) asymptotical competitive ratio* $\sigma^{a.s.}_{asymp.}(ALG)$, which assumes a probabilistic lens and disregards improbable events. We say that $\sigma^{a.s.}_{asymp.}(ALG) = \alpha$, if:

$$\mathbb{P}(\lim_{n\to\infty} \frac{ALG(I(n)^{rand})}{CIOPT(I(n)^{rand})} = \alpha) = 1, \tag{2.1}$$

where $I(n)^{rand}$ is a random instance with n orders following given stochastic assumptions. Note that $\sigma^{a.s.}_{asymp.}(ALG)$ strongly depends on the stochastic assumptions under consideration.

To analyze *general* large instances in Section 2.3.3, we turn to the *asymptotic competitive ratio* $\sigma_{asymp.}(ALG)$. We call algorithm ALG *asymptotically α-competitive* if, for any instance I, there exists a constant $const$, which is independent of the number and characteristics of the orders in I, but may depend on the warehouse geometry, such that:

$$ALG(I) \leq \alpha \cdot CIOPT(I) + const \tag{2.2}$$

The *asymptotic competitive ratio* $\sigma_{asymp.}(ALG)$ is then defined as the infimum over all constants α, such that ALG is asymptotically α-competitive.

Finally, to include small instances in our analysis (Section 2.3.4), which may display anomalies, we use the *strict competitive ratio* $\sigma(ALG)$, which is the worst-case ratio between $ALG(I)$ and $CIOPT(I)$ over *all* possible instances I of the problem:

$$\sigma(ALG) = \sup_{I} \frac{ALG(I)}{CIOPT(I)}. \tag{2.3}$$

Observe that the strict competitive ratio derives from the asymptotic competitive ratio by forcing the constant $const$ to be zero.

We begin by stating the key properties of OOBSRP in Section 2.3.1. Afterward, Section 2.3.2 establishes almost sure asymptotic optimality of *Reopt* in all the examined systems under general stochastic conditions. Section 2.3.3 and Section 2.3.4 examine the asymptotic and the strict competitive ratios of *Reopt*, respectively.

2.3.1 Properties of OOBSRP

The results of this section are used throughout the following discussion. We state bounds for a warehouse traversal in Lemma 1 as well as for the results of $Reopt$ and $CIOPT$ in Lemmas 2–4, respectively. We conclude by showing asymptotic optimality of *Reopt* for a special OOBSRP case, where each order contains exactly *one* item.

Let us denote an OOBSRP instance with n orders as $I(n)$. We denote as $I(n)^{r=0}$ the variant of some given instance $I(n)$, in which all the orders are instantly available, i.e., with $r_1 = r_2 = \ldots = r_n = 0$.

Lemma 1 (Upper bound for the warehouse traversal)**.** *Notwithstanding the cart technology – a pushcart or a robot – and given an arbitrary starting position of the picker, the shortest time needed to visit all the picking locations in the warehouse and return to any given location is bounded from above by the following constant u. The bound u is tight.*

$$u = (a+1)W + 2L. \tag{2.4}$$

Proof. See Supplement 2.7.1. □

Lemma 2 (Upper bound for $Reopt(I)$)**.** *In Pcart-N, Pcart, and Robot, the results of the respective Reopt policy for some instance $I(n)$ cannot be worse than the following bound:*

$$Reopt(I(n)) \leq r_n + CIOPT(I(n)^{r=0}) + u \tag{2.5}$$

Proof. At time r_n, when the last order arrives, *Reopt* reoptimizes having the complete information and requires no more time than the following simple policy: Complete the currently open batch and return to the depot in at most u time (cf. Lemma 1), then follow the route of $CIOPT$ for the instance $I(n)^{r=0}$ to collect the remaining orders. □

For *Pcart* and *Pcart-N*, let us define the *makespan* $\mathcal{M}(o_j)$ of order o_j as the length of the shortest route that starts at the depot, covers all items $s_j^i \in o_j$, and ends at the depot. Similarly, for $Robot$, we define the *makespan* $\mathcal{M}(o_j)$ of order o_j as the length of the shortest route that starts in any picking location $s_j^s \in o_j$, covers all picking locations in o_j, and ends at any arbitrary picking location $s_j^t \in o_j$.

Lemma 3. *In Pcart-N, Pcart, and Robot, the complete information optimum for any instance $I(n)^{r=0}$, for which all orders are initially available, cannot be better than the following bound:*

$$CIOPT(I(n)^{r=0}) \geq \frac{\sum_{i=1}^{n} \mathcal{M}(o_i)}{c}, \tag{2.6}$$

Proof. This bound assumes the ideal case in which each batch can be fully packed with c perfectly matching orders. These orders are then collected simultaneously during the picking process at the walking cost of only one of these orders, $\mathcal{M}(o_j)$, for o_j in the batch. In the case of *Robot*, it further ignores possible walking time between the items of subsequent *distinct* batches. □

The following Lemma 4 summarizes some important straightforward relations, which we state without a proof.

Lemma 4 (Lower bounds for $CIOPT(I)$). *In Pcart-N, Pcart, and Robot, and any instance $I(n)$:*

$$CIOPT(I(n)) \geq r_n \tag{2.7}$$

$$CIOPT(I(n)) \geq CIOPT(I(n)^{r=0}) \tag{2.8}$$

$$\geq CIOPT(J^{r=0}) \qquad \forall J \subseteq I(n) \tag{2.9}$$

Proposition 1. *In Robot, in the special case of OOBSRP with single-item orders, i.e., $k(o_j) = 1\ \forall j \in \{1, \ldots, n\}$, Reopt is asymptotically optimal, i.e., $\sigma_{asymp.}(Reopt) = 1$.*

Proof. Observe that *Robot* with single-item orders is equivalent to the online nomadic TSP. So, at the arrival of the last order at time r_n, *Reopt* reoptimizes having the complete information and collects the remaining orders in no more time than that required by the following simple policy: Traverse the warehouse in S-shape motion once. This traversal requires a constant amount of time, dependent only on the warehouse geometry. Using (2.7) and the definition of the asymptotic convergence in (2.2), we complete the proof. □

2.3.2 Asymptotic optimality of *Reopt*

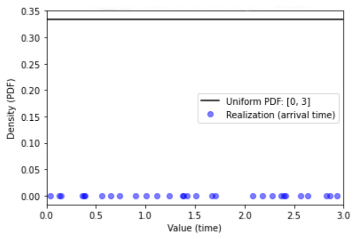
In this section, we show that, under general stochastic assumptions, when the number of incoming orders is sufficiently large, *Reopt* is asymptotically optimal with a probability of one (*almost surely*) in all the examined systems: *Pcart-N*, *Pcart*, and *Robot*.

In a probabilistic analysis, we consider *random* instances, and such an instance with $n \in \mathbb{N}$ orders is denoted as $I(n)^{rand}$. We denote the involved random variables in capital letters, e.g., orders $O = (O_1, O_2, \ldots, O_n)$ and arrivals $R = (R_1, R_2, .., R_n)$. We assume that *distinct* customer orders O_j in O are *independently identically distributed (i.i.d.)*. For mathematical completeness, we redefine O_j as a multivariate random variable $O_j = (K_j, S_j^1, ..., S_j^{K_j})$ with K_j picking locations $S_j^1, \ldots, S_j^{K_j}$. Further, in case of *Robot*, we exclude the trivial case of single-item orders (cf. Proposition 1) and assume $\mathbb{E}[K_1] > 1$. Observe that the specified stochastic assumptions explicitly allow product items S_j^i and S_j^l of the *same* order O_j to be correlated, such as phones and matching protective cases, which are usually ordered together.

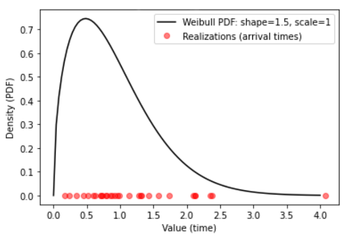
We assume that the random sequences R and O are independent. In the following, we will discuss two common stochastic models for the arrival time sequence R.

We begin by examining the order statistics property of arrival times in Section 2.3.2.1, an order arrival model suggested in the context of online routing problems by Jaillet and Wagner (2008b). This model considers a fixed number n of orders, with arrival times that are *independent and identically distributed (i.i.d.)* according to a fixed but *arbitrary* distribution. The *i.i.d.* assumption is natural in e-commerce warehousing, since customers act independently. The distribution can be arbitrary (see Figure 2.4), e.g., uniform, single-peaked, skewed, multimodal (the latter applies, for instance, when most orders arrive during lunch breaks and evenings). Order statistics arrival model can have

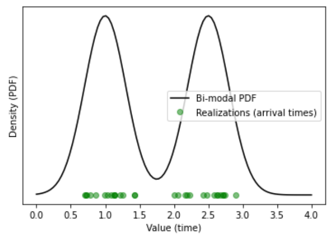
Figure 2.4: Realizations of 30 order arrival times with the order statistics model for different distributions



(a) Uniform distribution



(b) Weibull distribution



(c) Bi-modal distribution

Note. The three graphs show the probability distribution function (PDF) of the respective distributions (line) and the dots on the x-axis represent the realizations of the arrival times.

both *finite* and *infinite* support; in the former case, the orders arrive within a finite time interval, like a shift or a day, while in the latter, order arrival times are unrestricted. For an example of the latter case, consider picking a stock of n limited-edition cell phones from a warehouse that exclusively contains those items. The picking process concludes when all n items have been picked.

The analysis in Section 2.3.2.2, where order arrivals follow a homogeneous Poisson process, builds upon the results of Section 2.3.2.1. We proceed with a formal definition of the two models and the analytical results in Sections 2.3.2.1 and Section 2.3.2.2.

2.3.2.1 Case of the order statistics property of arrival times

The *order statistics property of arrival times* can be defined as follows. The arrival times of the orders are i.i.d. realizations $Y_j, j \in \{1, ..., n\}$, of a generic random variable $Y \geq 0$ with an arbitrary, given distribution and mean $\mu_Y < \infty$. To derive the arrival times of the orders, we sort the realizations of Y, the j^{th} arrival time is the j^{th} order statistic: $R_j = Y_{(j)}$, $Y_{(1)} \leq Y_{(2)} \leq ... \leq Y_{(n)}$.

Theorem 1 states one of the main results of this work.

Theorem 1. *In Pcart-N, Pcart, and Robot, given the specified stochastic assumptions, including the order statistics property of arrival times:*

$$\lim_{n\to\infty} \frac{Reopt(I(n)^{rand})}{CIOPT(I(n)^{rand})} = 1 \quad \textit{a.s.} \tag{2.10}$$

Proof. Following a similar line of proof as Jaillet and Wagner, 2008b, we first recall two relations from Lemmas 4 and 2, respectively:

$$CIOPT(I(n)^{rand}) \geq CIOPT(I(n)^{rand,r=0}) \tag{2.11}$$

$$Reopt(I(n)^{rand}) \leq CIOPT(I(n)^{rand,r=0}) + R_n + u \tag{2.12}$$

Together, they imply

$$\frac{Reopt(I(n)^{rand})}{CIOPT(I(n)^{rand})} \leq 1 + \frac{n}{CIOPT(I(n)^{rand,r=0})} \cdot \frac{R_n + u}{n} \tag{2.13}$$

Since $\mu_Y < \infty$, Lemma 6 of Jaillet and Wagner, 2008b can be applied:

$$\lim_{n\to\infty} \frac{R_n}{n} + \frac{u}{n} = 0 \qquad \text{a.s.} \tag{2.14}$$

At this point, the proof of Jaillet and Wagner, 2008b developed for the capacitated online TSP is not applicable and we proceed in a unique way. To show that $\frac{n}{CIOPT(I(n)^{rand,r=0})}$ is bounded from above by a constant (which does not depend on n), we use the lower bound from Lemma 3:

$$\frac{CIOPT(I(n)^{rand,r=0})}{n} \geq \frac{1}{c} \cdot \frac{\sum_{j=1}^{n} \mathcal{M}(O_j)}{n} \tag{2.15}$$

Recall that essentially makespan $\mathcal{M}(O_j)$ of order O_j is the shortest picker route to collect all items in O_j. Given the stochastic assumptions, $\mathcal{M}(O_j), j \in \{1, ..., n\}$, are i.i.d.. So by the Strong Law of Large Numbers:

$$\lim_{n\to\infty} \frac{1}{c} \cdot \frac{\sum_{j=1}^{n} \mathcal{M}(O_j)}{n} = \frac{1}{c} \cdot \mathbb{E}[\mathcal{M}(O_1)] > 0 \qquad \text{a.s.} \tag{2.16}$$

We only consider the case of a strictly positive $\mathbb{E}[\mathcal{M}(O_1)]$. Otherwise, the problem is trivial, and *Reopt* is optimal per definition in *Pcart-N* and *Pcart*; in *Robot*, the problem reduces to the single-item order case for which *Reopt* is asymptotically optimal by Proposition 1.

Observe that $\frac{1}{c} \cdot \mathbb{E}[\mathcal{M}(O_1)]$ is a constant that does not depend on n. The convergence of (2.16) implies that the right-hand sight of (2.15) is larger than some *nonzero* constant $\zeta < \frac{1}{c}\mathbb{E}[\mathcal{M}(O_1)]$ a.s. if the number of orders n is large enough, i.e., $\forall n > n_0$ for some n_0:

$$\frac{CIOPT(I(n)^{rand,r=0})}{n} \geq \zeta \Leftrightarrow \frac{n}{CIOPT(I(n)^{rand,r=0})} \leq \frac{1}{\zeta} \qquad \text{a.s.} \tag{2.17}$$

This, together with (2.13) and (2.14), completes the proof:

$$\frac{Reopt(I(n)^{rand})}{CIOPT(I(n)^{rand})} \leq 1 + \frac{n}{CIOPT(I(n)^{rand,r=0})} \cdot \frac{R_n + u}{n} \xrightarrow{n\to\infty} 1 + 0 \qquad \text{a.s.}$$

□

The interpretation of Theorem 1 for an order statistic with a *finite*-support arrival-time distribution Y such as during a shift, is quite intuitive. As the number of orders arrived during this shift increases, the picker becomes overloaded, and most orders will have to be picked *after* the shift ends, regardless of the picking policy. Therefore, Reopt, which picks optimally after the end of the shift when no more orders arrive, converges to optimality. However, the interpretation is less straightforward for a random arrival time distribution Y with *infinite* support, such as a normal distribution in the limited-edition cell phone example in Section 2.3.2. In this case, as the number of available products increases, $n \to \infty$, the arrival time of the last order placement, R_n, will extend to infinity (as for arbitrary $x < \infty$, $\mathbb{P}(R_n > x) = 1 - \mathbb{P}(Y \leq x)^n \to 1$ by definition of Y's infinite support). This implies that the picking system remains *dynamic* throughout the entire picking process. Even for these systems, Theorem 1 proves that *Reopt* can keep up with an optimal picking strategy, with probability one.

2.3.2.2 Case of the homogeneous Poisson process for arrival times

Let random variables $X_j, j \in \mathbb{N}$, which represent the time between the j^{th} and the $(j-1)^{th}$ order arrival, be i.i.d. exponentially distributed with mean $\frac{1}{\lambda}$, then the sequence of order arrival times follows a *homogeneous Poisson process* with rate λ.

A central notion for the analysis of Poisson-arriving orders is the *queue* $Q^{Reopt}(I(n)^{rand})$, which is the set of pending orders at time R_n in instance $I(n)^{rand}$ for policy *Reopt*. We prove in Theorem 2 that *Reopt* is *a.s.* asymptotically optimal if $\lim_{n\to\infty} \frac{|Q^{Reopt}(I(n)^{rand})|}{n} = 0$, i.e. if with probability one, the queue of pending orders does not grow as fast as the total number of received orders. Note that this condition would be necessary under the *Reopt* policy to safeguard the warehousing system against potential instability.

Because the queue length $|Q^{Reopt}(I(n)^{rand})|$ is generated by two independent sequences of i.i.d. random variables – the picking locations and the inter-arrival times – we believe that by the 0-1-Laws, the probability of the event $\lim_{n\to\infty} \frac{|Q^{Reopt}(I(n)^{rand})|}{n} = 0$ is either 1 or zero for a given rate λ, and we suspect that the probability increases monotonously in λ. In other words, we believe that $\lim_{n\to\infty} \frac{|Q^{Reopt}(I(n)^{rand})|}{n} = 0$ is true for small arrival rates until a fixed threshold $\overline{\lambda}$ (see Conjecture 1).

In the case of asymptotically increasing arrival rates, the probability that *Reopt* approaches optimality converges to 1 as well, as we prove in Theorem 3.

Observe the *stand-alone character* of this section's arguments in the warehousing research. The available queuing systems analysis, which is usually employed when order arrivals follow a Poisson process, requires i.i.d. service times of the batches. The latter is the case, for instance, for the 'first come first served' policy, when orders are serviced in the sequence of their arrival. This is not the case for *Reopt* and neither it is the case for $CIOPT$, where the service times of the batches are interdependent due to reoptimization.

Theorem 2. *In Pcart-N, Pcart, and Robot, if* $\lim_{n\to\infty} \frac{|Q^{Reopt}(I(n)^{rand})|}{n} = 0$ *a.s., then:*

$$\lim_{n\to\infty} \frac{Reopt(I(n)^{rand})}{CIOPT(I(n)^{rand})} = 1 \quad \textit{a.s.} \tag{2.18}$$

Proof. Exploiting the i.i.d. inter-arrival time property of the Poisson process, the proof combines the Strong Law of Large Numbers with the fact that the warehouse has bounded dimensions. We refer to the Supplement 2.7.2 for details. □

Observe that Theorem 2 demonstrates the asymptotic optimality of *Reopt* not only for a.s. *stable* picking systems, but also for certain systems with high order arrival rates, where there is a non-zero probability that the queue length grows to infinity over time. For instance, it applies to picking systems with queues growing as fast as $\sqrt{n}$, which is sublinear in n. In summary, the statement of Theorem 2 is quite general, extending from small to slightly too high order arrival rates.

Conjecture 1. *In Pcart-N, Pcart, and Robot, there is a rate* $\tilde{\lambda}$ *such that* $\mathbb{P}(\lim_{n\to\infty} \frac{|Q^{Reopt}(I(n)^{rand})|}{n} = 0) = 1$ *for all* $\lambda < \tilde{\lambda}$. *Thus, for all rates* $\lambda < \tilde{\lambda}$*:*

$$\lim_{n\to\infty} \frac{Reopt(I(n)^{rand})}{CIOPT(I(n)^{rand})} = 1 \quad \textit{a.s.} \tag{2.19}$$

It remains unclear whether *Reopt* is asymptotically optimal a.s. for rates $\lambda \geq \tilde{\lambda}$. To prove a somewhat weaker notion of convergence, we fix the time interval $[0, t]$ (e.g., a shift) and examine the orders $I(t)^{rand}$ arrived in this time interval $[0, t]$ given the arrival rate λ of the underlying Poisson process. Recall that, by definition, the number of orders $N(t)$ in the interval is Poisson($\lambda \cdot t$)-distributed. We denote by $\mathbb{P}_\lambda$ the underlying probability measure for a given arrival rate λ.

Theorem 3. *Let $\lambda_i > 0, i \in \mathbb{N}$ be any increasing sequence of arrival rates such that $\lambda_i \to \infty$ if $i \to \infty$. In Pcart-N, Pcart, and Robot, the following holds true:*

$$\lim_{i\to\infty} \mathbb{E}_{\lambda_i}[\frac{Reopt(I(t)^{rand})}{CIOPT(I(t)^{rand})}] = 1 \tag{2.20}$$

And for all $\epsilon > 0$:

$$\lim_{i\to\infty} \mathbb{P}_{\lambda_i}(\frac{Reopt(I(t)^{rand})}{CIOPT(I(t)^{rand})} \geq 1+\epsilon) = 0 \tag{2.21}$$

Proof. The detailed proof is provided in the Supplement 2.7.3. Recall that to establish convergence, Theorem 1 depended on a sublinear growth of the arrival time R_n of the n^{th} order ($\frac{R_n}{n} \xrightarrow[a.s.]{n\to\infty} 0$). This is not the case when the arrival time sequence is a Poisson process with some given inter-arrival time $\frac{1}{\lambda}$. Therefore, we must recur to a very different line of proof. In this proof, we perform a formal transition from the probability space described in Section 2.3.2.1 to the probability space of this section. The proof proceeds in three steps.

First, we want to leverage the convergence statement of Theorem 1. Therefore, we apply it in the case of instances $I(n)^{rand}$ with independent uniformly$[0,t]$-distributed arrival times for an arbitrary $t \in \mathbb{R}^+$. Since the ratio $\frac{Reopt(I(n)^{rand})}{CIOPT(I(n)^{rand})}$ is bounded (cf. Section 2.3.4), we can restate Theorem 1 as the *convergence in mean*.

Secondly, we assume that the order arrival times follow a Poisson process with some given arrival rate $\lambda \in \mathbb{R}^+$. We examine instances $I(t)^{rand}$ comprising all orders arriving within a predefined time interval $[0,t], t \in \mathbb{R}^+$. Let $N(t)$ be the random number of these orders. We then use the Order Statistics Property of the homogeneous Poisson process, in combination with the statements of the first step of this proof, to derive the *conditional* convergence in mean:

$$\lim_{n\to\infty} \mathbb{E}_{\lambda}[\frac{Reopt(I(t)^{rand})}{CIOPT(I(t)^{rand})} \mid N(t) = n] = 1 \tag{2.22}$$

Convergence implies that for any $\epsilon > 0$, we can find $n_\epsilon \in \mathbb{N}$ such that:
$\mathbb{E}_{\lambda}[\frac{Reopt(I(t)^{rand})}{CIOPT(I(t)^{rand})} \mid N(t) = n] \leq 1 + \frac{\epsilon}{2}$ for all $n > n_\epsilon$.

Thirdly, we compute the *unconditional* expectation $\mathbb{E}_{\lambda}[\frac{Reopt(I(t)^{rand})}{CIOPT(I(t)^{rand})}]$ using the Law of Total Expectation for distinct groups of instances — those with $N(t) > n_\epsilon$ and those with $N(t) \leq n_\epsilon$. After some transformations, we show that for any sequence of rates λ_i with $\lambda_i \xrightarrow{i\to\infty} \infty$ and any $\epsilon > 0$, one can choose $i_\epsilon \in \mathbb{N}$ in dependence of t and ϵ, such that $\mathbb{E}_{\lambda_i}[\frac{Reopt(I(t)^{rand})}{CIOPT(I(t)^{rand})}] \leq (1+\epsilon)$ for all $i \geq i_\epsilon$.

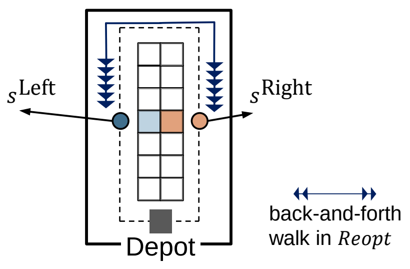
We use the Markov's inequality to derive the implication of (2.21) from (2.20). □

2.3.3 Performance of *Reopt* in general large OOBSRP instances

In this section, we abolish the stochastic assumptions and examine whether *Reopt* retains its good performance for general large problem instances. We show that even in the *worst-case*, the *Reopt* result cannot be improved by more than 50% in any given OOBSRP instance if it is sufficiently large. This is still an excellent performance.

Overall, having examined different worst-case examples for *Reopt* in OOBSRP with a pushcart and a robot, we see two underlying mechanisms of the *Reopt*'s suboptimal result: *(i)* unfortunate batching and, in case of the interventionist *Reopt*, *(ii)* unnecessary

Figure 2.5: *Pcart*: Unfortunate large instance for *Reopt*



Note. $c = 2$. Single-item orders with items s^{left} and s^{right} arrive such that, in *Reopt*, the picker keeps walking back and forth without completing any batch until the arrival of the last order.

walking. In *(i)*, *Reopt* places orders in one batch even if the savings from their batching are small compared to a separate picking of these orders (cf. Example 1). $CIOPT$, to the contrary, can anticipate upcoming better matching orders. In *(ii)*, *Reopt* completely changes the current route of the picker, including her next destination, at the slightest promise of time-saving. Paradoxically, this behavior may result in a prolonged back-and-forth walking without picking an item, which is avoided in $CIOPT$ (cf. the proof of Proposition 3).

In the following, Proposition 2 establishes the upper bound of 2 for $\sigma_{asymp.}(Reopt)$ in all the examined picking systems. Proposition 3 states that this worst-case ratio is *tight* for *Pcart* and *Robot*.

Proposition 2. *In Pcart-N, Pcart, and Robot, $\sigma_{\text{asymp.}}(Reopt) \leq 2$. In other words, for every instance $I(n)$*

$$Reopt(I(n)) \leq 2 \cdot CIOPT(I(n)) + const \tag{2.23}$$

where $const$ is the warehouse traversal time u from Lemma 1.

Proof. By Lemma 2 and Lemma 4:

$$\begin{aligned} Reopt(I(n)) &\leq r_n + u + CIOPT(I(n)^{r=0}) \\ &\leq 2 \cdot CIOPT(I(n)) + u \end{aligned} \tag{2.24}$$

□

Proposition 3. *In $Pcart$ and $Robot$, the upper bound of 2 is tight for the asymptotic competitive ratio, i.e.*

$$\sigma_{asymp.}(Reopt) \geq 2 \tag{2.25}$$

Note that in the case of $Pcart$, the statement requires a warehouse with at least two aisles.

Proof. We examine the case of *Pcart* and refer to Supplement 2.7.5 for the case of *Robot*.

Figure 2.5 provides an unfortunate example I^{worst} for $c = 2$ and a two-aisle warehouse. This example can be generalized for $c > 2$ and more general warehouse layouts.

Arriving orders in I^{worst} consist of one item, which is placed either in the left aisle (at position s^{left}) or in the right aisle (at position s^{right}). We simplify the notation for these single-order items and write $o_i = s^{\text{left}}$ or $o_i = s^{\text{right}}$, $\forall i \in \mathbb{N}$, respectively. Recall that W

and L are the width and the length of the warehouse, respectively. In this example, the picker can traverse both aisles in a cyclic tour of length $T = 2W + 2L$ from the depot. Let $d(l_d, s^{\text{left}}) = (l_d, s^{\text{right}}) = \frac{1}{2} \cdot d(s^{\text{left}}, s^{\text{right}}) = \frac{1}{2}T$.

Instance I^{worst} consists of $n = 2k$ orders, where $k \in \mathbb{N}_{\geq 4}$ is some *uneven* number. Each time, orders s^{left} and s^{right} arrive simultaneously. The idea is to set the inter-arrival times of the orders such that *Reopt* makes the picker oscillate between the positions of s^{left} and s^{right} in the upper half of the warehouse, without picking any item, until the last order arrives.

At time $r_1 = r_2 = 0$, *Reopt* batches the only two available orders together: $S_1 = \{o_1, o_2\}$ with $o_1 = s^{\text{left}}$ and $o_2 = s^{\text{right}}$, and plans to pick them in a circular tour. *W.l.o.g.*, let *Reopt* pick o_1 first (at time $\frac{1}{4}T$).

At time $r_3 = r_4 = \frac{3}{4}T - \frac{\delta}{2k-2}$, an order s^{left} and an order s^{right} arrive. In *Reopt*, the picker is currently at distance $\frac{\delta}{2k-2}$ above item s^{right} with o_1 placed in her cart. *Reopt* has two choices. The first is to follow the initial plan and pick batch $\{o_1, o_2\} = \{s^{\text{left}}, s^{\text{right}}\}$ to the end, then pick batch $\{o_3, o_4\} = \{s^{\text{left}}, s^{\text{right}}\}$, at a total cost of $2T$. The second is to turn back and complete a modified batch $S_1 = \{o_1, o_3\} = \{s^{\text{left}}, s^{\text{left}}\}$, then pick $S_2 = \{o_2, o_4\} = \{s^{\text{right}}, s^{\text{right}}\}$, at a total cost of $r_4 + \frac{3}{4}T - \frac{\delta}{2k-2} + \frac{T}{2} = 2T - 2 \cdot \frac{\delta}{2k-2}$. *Reopt* decides for the second option.

At time $r_5 = r_6 = \frac{5}{4}T - 3 \cdot \frac{\delta}{2k-2}$, an order s^{left} and an order s^{right} arrive, and the picker is at a distance of $\frac{\delta}{2k-2}$ above s^{left} with o_1 placed in her cart. *Reopt* has two choices. The first is to continue picking $S_1 = \{o_1, o_3\} = \{s^{\text{left}}, s^{\text{left}}\}$ and then $S_2 = \{o_2, o_4\} = \{s^{\text{right}}, s^{\text{right}}\}$ and $S_3 = \{o_5, o_6\} = \{s^{\text{left}}, s^{\text{right}}\}$, at a total cost of $2T - 2 \cdot \frac{\delta}{2k-2} + T = 3T - 2 \cdot \frac{\delta}{2k-2}$. The second is to change the first batch again to $S_1 = \{o_1, o_2\} = \{s^{\text{left}}, s^{\text{right}}\}$, then pick $S_2 = \{o_3, o_5\} = \{s^{\text{left}}, s^{\text{left}}\}$ and $S_3 = \{o_4, o_6\} = \{s^{\text{right}}, s^{\text{right}}\}$, at a total cost of $r_6 + \frac{3}{4}T - \frac{\delta}{2k-2} + 2 \cdot \frac{T}{2} = 3T - 4 \cdot \frac{\delta}{2k-2}$. *Reopt* decides for the second option.

In general, at time $r_{2p-1} = r_{2p} = \frac{2p-1}{4}T - (2p-3) \cdot \frac{\delta}{2k-2}$ for $p \in \{2, .., k\}$, an order s^{left} and an order s^{right} arrive. If p is uneven, the picker is at distance $\frac{\delta}{2k-2}$ above s^{left} planning to pick $\frac{p-1}{2}$ batches of type $\{s^{\text{left}}, s^{\text{left}}\}$ first, then $\frac{p-1}{2}$ batches of type $\{s^{\text{right}}, s^{\text{right}}\}$. With the two recently arrived orders, *Reopt* decides to pick batch $S_1 = \{o_1, o_2\} = \{s^{\text{left}}, s^{\text{right}}\}$ instead, then $\frac{p-1}{2}$ batches of type $\{s^{\text{left}}, s^{\text{left}}\}$ and $\frac{p-1}{2}$ batches of type $\{s^{\text{right}}, s^{\text{right}}\}$. So, the picker turns around towards s^{right}. Similarly, if p is even, the picker is at distance $\frac{\delta}{2k-2}$ above s^{right} planning to pick $S_1 = \{o_1, o_2\} = \{s^{\text{left}}, s^{\text{right}}\}$ first, then $\frac{p}{2} - 1$ batches of type $\{s^{\text{left}}, s^{\text{left}}\}$, and then $\frac{p}{2} - 1$ batches of type $\{s^{\text{right}}, s^{\text{right}}\}$. With the two recently arrived orders, *Reopt* decides to pick $\frac{p}{2}$ batches of type $\{s^{\text{left}}, s^{\text{left}}\}$ first, then $\frac{p}{2}$ batches of type $\{s^{\text{right}}, s^{\text{right}}\}$. So, the picker turns around towards s^{left}.

If k is uneven, $Reopt(I^{\text{worst}}) = kT - \delta$, which equals the release time of the last order $r_{2k} = \frac{2k-1}{4}T - (2k-3) \cdot \frac{\delta}{2k-2}$ plus the time $(\frac{3}{4}T - \frac{\delta}{2k-2})$ to reach s^{right} and complete the first batch plus $(\frac{k-1}{2} \cdot 2 \cdot \frac{1}{2}T)$ to pick $\frac{k-1}{2}$ batches of type $\{s^{\text{left}}, s^{\text{left}}\}$ and $\frac{k-1}{2}$ batches of type $\{s^{\text{right}}, s^{\text{right}}\}$. In contrast, $CIOPT$ picks $S_1 = \{o_1, o_2\}$, then $\frac{k-1}{2}$ batches of type $\{s^{\text{left}}, s^{\text{left}}\}$ and $\frac{k-1}{2}$ batches of type $\{s^{\text{right}}, s^{\text{right}}\}$ such that $CIOPT(I(k)) = T + (k-1)\frac{T}{2} = \frac{k}{2}T + \frac{T}{2}$. This implies that

$$\lim_{k \to \infty} \frac{Reopt(I(k))}{CIOPT(I(k))} = 2 \tag{2.26}$$

By Lemma 5 in Supplement 2.7.4, $\sigma_{asymp}(Reopt) \geq 2$ follows immediately for *Pcart*.

For $c > 2$, we can place multiple single-item orders at positions s^{left} and s^{right}. Note that in the example above, *Reopt* faces ties. Initially, there are two alternative plans with the same cost – picking the orders together as $S_1 = \{o_1, o_2\}$ in a cyclic tour or picking them separately as $S_1 = \{o_1\}$ and $S_2 = \{o_2\}$. In the second plan, the example does not work. We can avoid ties by considering $o_1 = \{s^{\text{left}}, s'\}$, where s' is placed just above s^{left}, which makes *Reopt* to prefer picking both orders together. □

2.3.4 Performance of *Reopt* in instances of all sizes

In *any* OOBSRP instance, *Reopt* never exceeds more than 4 times the complete-information optimum in *Pcart* and *Robot* and no more than 2.5 times in *Pcart-N*, respectively (see Proposition 4). However, strict competitive ratios of *any* deterministic online algorithm are at least 2, 1.64, and 2 in *Pcart-N*, *Pcart*, and *Robot*, respectively, by Proposition 5. In other words, the performance of *Reopt* cannot be improved much across instances of all sizes.

Observe that the upper bound 2.5 on the strict competitive ratio for *Reopt* in *Pcart-N* is almost tight, as we can construct instances I with $\frac{Reopt(I)}{CIOPT(I)} = 2.5 - \tilde{\epsilon}$ for an arbitrary small $\tilde{\epsilon} > 0$ (see Example 1).

Proposition 4. *The following upper bounds hold for the strict competitive ratio:*

$$\text{In Pcart-N:} \quad \sigma(Reopt) \le 2.5 \tag{2.27}$$

$$\text{In Pcart and Robot:} \quad \sigma(Reopt) \le 4 \tag{2.28}$$

Proof of (2.27). We are in system *Pcart-N* and consider two cases.

In the first case, the picker in *Reopt* is idle at the depot at time r_n. Then, by Lemma 4:

$$Reopt(I) \le r_n + CIOPT(I^{r=0}) \le 2 \cdot CIOPT(I) \tag{2.29}$$

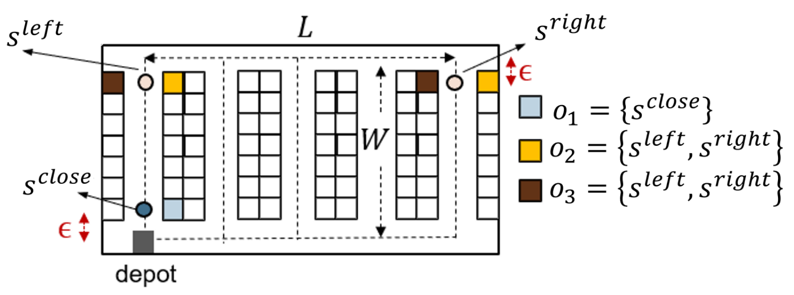
In the other case, at r_n, the picker in *Reopt* is collecting batch $\tilde{S}$, for which she left the depot at time $t(\tilde{S})$. Let $\tilde{j}$ be the index of the first order that arrives *after* time $t(\tilde{S})$. Let $I_{\ge\tilde{j}}$ denote an instance which contains the subset of orders $\{o_i | \ s.t. \ i \ge \tilde{j}\}$ and assumes these orders are *instantly available*. We denote by $\tilde{J}$ an instance that contains all orders in the queue of *Reopt* at time $t(\tilde{S})$, and that assumes all these orders are *instantly available*. By definition, at time $t(\tilde{S})$, *Reopt*'s plan to collect the orders of $\tilde{J}$ is *optimal* given the current information. Therefore the solution of *Reopt* is not worse than the following feasible policy: At $t(\tilde{S})$ complete first $\tilde{J}$ in time $CIOPT(\tilde{J})$. After that, the picker is at the depot and collects the remaining orders in at most $CIOPT(I_{\ge\tilde{j}})$ time. We denote $d_{max} = \max\{d(l_d, s^i_j) \mid s^i_j \in o_{\tilde{j}} \cup o_{\tilde{j}+1} \cup ... \cup o_n\}$, the maximal distance from the depot to an item in $I_{\ge\tilde{j}}$. Altogether:

$$\begin{aligned} Reopt(I) &\le t(\tilde{S}) + CIOPT(\tilde{J}) + CIOPT(I_{\ge\tilde{j}}) && (2.30) \\ &\le r_{\tilde{j}} + CIOPT(I_{\ge\tilde{j}}) + CIOPT(I) && (2.31) \\ &\le CIOPT(I) + d_{\max} + CIOPT(I) && (2.32) \\ &\le 2.5 \cdot CIOPT(I) && (2.33) \end{aligned}$$

Line (2.31) follows by Lemma 4. In $CIOPT$, the picker can move towards the first picking location prior to the arrival of the corresponding order, thus (2.32) ensues from (2.31): $CIOPT(I) + d_{\max} \ge r_{\tilde{j}} + CIOPT(I_{\ge\tilde{j}})$. Finally, since $CIOPT(I) \ge 2 \cdot d_{\max}$, (2.33) follows.

The logic above does not hold for *Pcart-Int* and *Robot*, because the initiated batch $\tilde{S}$ can be changed at each order arrival after $t(\tilde{S})$, possibly leading to a worse solution.

Figure 2.6: *Pcart-N*: Unfortunate small instance I^{unf} for *Reopt*.



Note. $c = 2, r_1 = r_2 = L + W,$ and $r_3 = L + W + \delta$ with $\delta \to 0$. *Reopt* myopically collects poorly matching orders o_1 and o_2 together.

Proof of (2.28). Consider *Pcart* and recall Lemma 2: At time r_n, when the last order arrives, *Reopt* reoptimizes having the complete information and requires not more time than the following simple policy: 1) Complete the currently open batch and move to the depot, then 2) use $CIOPT(I(n)^{r=0})$ to collect the remaining orders. For 1), the picker requires at most $r_n + CIOPT(I(n)^{r=0})$, because she can walk from the current position all the way back to the depot in r_n and then follow $CIOPT$ for picking the remaining items of the currently open batch (and skipping the rest). Thus, for 1) together with 2), by Lemma 4:

$$\begin{aligned} Reopt(I(n)) &\leq 2 \cdot r_n + 2 \cdot CIOPT(I(n)^{r=0}) \\ &\leq 4 \cdot CIOPT(I(n)) \end{aligned} \tag{2.34}$$

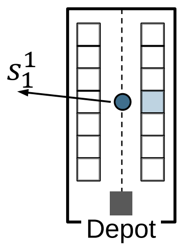
The proof for *Robot* proceeds along the same lines. Observe, however, that at the start of step 2) the picker may be away from the depot. However, she can follow $CIOPT$ in the *reverse* direction to collect the remaining items. □

Figure 2.6 provides an unfortunate instance I^{unf} for *Pcart-N* with $c = 2$, such that $\frac{Reopt(I^{unf})}{CIOPT(I^{unf})} \approx 2.5$.

Example 1. *In instance I^{unf} of Figure 2.6, Reopt receives two orders at time $r_1 = r_2 = W + L$: o_1 with only one item s^{close} at the closest picking location to the depot, $d(l_s, s^{close}) = \epsilon$, and o_2 with two far away picking locations that can be visited best in a cyclic tour of length $2(W + L)$ from the depot. Reopt forms batch $s_1 = \{o_1, o_2\}$, and as soon as the picker leaves the depot (at time $r_3 = W + L + \delta$, $\delta \to 0$), a third order $o_3 = o_2$ arrives. Although o_2 and o_3 are perfectly matching, Reopt in $Pcart - N$ cannot modify the commenced batch S_1, and completes the operation with $S_2 = \{o_3\}$ at time $Reopt(I^{unf}) = r_1 + 2(W + L) + 2(W + L) = 5(W + L)$. On the other hand, CIOPT picks the first item of o_2 and o_3 (in s^{right}) at time $W + L + \delta$ and completes its first batch $\{o_2, o_3\}$ at time $2(L + W) + \epsilon + \delta$. After picking o_1 separately, $CIOPT(I^{unf}) = 2(L + W) + 3\epsilon + \delta$. Since $\delta \to 0$ and ϵ is the distance from the depot to the closest picking location, we can construct instances with arbitrarily small $(3\epsilon + \delta) > 0$.*

We now denote by ALG an arbitrary deterministic online algorithm for OOBSRP in each of the studied systems (*Pcart-N*, *Pcart*, and *Robot*). An algorithm is *deterministic*, if for every fixed instance, it always generates the same result when run repeatedly. We differentiate deterministic algorithms from probabilistic algorithms, which may choose their

Figure 2.7: *Pcart-N*: Unfortunate small instance for all algorithms



Note. No action suits all possible future well: a) no further orders arrive, b) further orders arrive.

actions randomly. Note that ALG in *Pcart-N* must be an arbitrary *non-interventionist* algorithm, that can update the current picking plan only when the picker is at the depot.

Proposition 5. *For every deterministic online algorithm ALG for $OOBSRP$, the strict competitive ratio cannot be better as the following:*

$$\textit{In Pcart-N and Robot:} \quad \sigma(ALG) \geq 2 \tag{2.35}$$

$$\textit{In Pcart :} \quad \sigma(ALG) \geq 1.64 \tag{2.36}$$

Note that we require $c \geq 2$ for Pcart-N and Pcart.

Proof. The lower bounds for *Pcart* and *Robot* are based on Theorem 3.1 and Theorem 3.3 of Ausiello et al. (2001), respectively. In both cases, Ausiello et al. (2001) construct unfortunate instances for the online TSP *on the real line*. These instances can be transformed to the warehouse environment. The proof requires nontrivial extensions, which are given in the Supplement 2.7.6. Indeed, in the warehouse, the movements of the picker are not limited to a real line and, moreover, the batching capacity should be respected.

To prove the lower bound for any ALG in *Pcart-N*, consider the instance in Figure 2.7. Recall that the warehouse width is W and the picker moves at the unit speed. Initially, at $r_1 = 0$, order $o_1 = \{s_1^1\}$ with $d(l_d, s_1^1) = \frac{1}{2}W$ is available. Let ALG decide to dispatch the picker to collect the single-order batch $\{o_1\}$ at time t. In the following, we will construct an unfortunate instance for each possible value of t.

If $t \geq W$, no more orders arrive. ALG needs $(t + W) \geq 2W$ time to collect o_1 in a return-trip from the depot, while $CIOPT(I) = W$ for the resulting instance I.

If $\frac{1}{2}W \leq t < W$, then a second order $o_2 = \{s_2^1\}$ with $d(l_d, s_2^1) = t$ arrives at time $r_2 = t$ (but after the picker dispatched). Then, in the resulting instance I, batch $\{o_1, o_2\}$ can be collected in time $CIOPT(I) = 2t$, while ALG collects o_1 and o_2 in two separate batches with $ALG^{Pcart-N}(I) = t + W + 2t > 4t$.

If $t < \frac{1}{2}W$, then a second order $o_2 = \{s_2^1\}$ with $d(l_d, s_2^1) = \frac{1}{2}W$ arrives at time $r_2 = t$ (but after the picker dispatched). For the resulting instance I, it is possible to collect both orders in the single batch $\{o_1, o_2\}$ in time $CIOPT(I) = W$, while $ALG(I) = t + W + W \geq 2W$, because it picks o_1 and o_2 in separate batches.

In all the cases above, the ratio $\frac{ALG}{CIOPT(I)} \geq 2$ for the respective instance I in *Pcart-N*.

Observe that in the unfortunate examples for *Pcart-N* and *Pcart*, $c \geq 2$. If $c = 1$, a weaker lower bound of 1.5 can be straightforwardly derived in both cases from the unfortunate instance created for OOBSRP in *Robot* in the Supplement 2.7.6. □

2.4 Computational experiments

The analysis of Section 2.3.2 showed that *Reopt* may converge to an optimal policy with an increasing instance size. In this section, we compare the results of *Reopt* to optimality (measured by $CIOPT$) experimentally and investigate the gap to optimality in small instances.

Note the computational challenges associated with these experiments. In order to make representative measurements across diverse warehouse settings, we create 2600 OBSRP* instances and solve more than 2380 of those to *optimality*. The quantity of items that need to be picked in the created instances ranges between 4 and 50, and OBSRP*, which combines batching and routing, is NP-hard in the strong sense (c.f. Löffler et al., 2022). Given the scale of the endeavor, we limit our studies to the interventionist versions of *Reopt*, i.e., systems *Pcart* and *Robot*. Section 2.4.1 explains the data generation and Section 2.4.2 reports on experiments.

2.4.1 Data sets

We randomly generate instances grouped in 10 settings (see Table 2.2) that feature different warehouse environments. Each setting comprises 10 instances of each order size $n \in \{3, 4, \ldots, 15\}$. The procedure is repeated for two picking systems – *Pcart* and *Robot*. Thus in total, 2600 instances (2 systems × 10 settings × 13 sizes × 10) are created.

In *the Base setting (Base)*, the picker's speed equals 0.8 m/s (cf. Henn, 2012) and it takes 10 s to collect one item at each picking location. The warehouse consists of 10 aisles and 3 cross-aisles, each of which has a width of 3 m. One single shelf row marks the left and right outer borders of the storage area, respectively, and neighboring aisles are separated by two shelf rows placed alongside each other (cf. Figure 2.2). The shelves measure 3 m in width and 1 m in depth. One aisle contains 20 shelves. Thus, the warehouse has a length of 50 m (10 aisles × 3 m + $(1 + 9 \cdot 2 + 1)$ shelf rows × 1 m) and a width of 69 m (20 shelves × 3 m + 3 cross-aisles × 3 m). Each shelf has slots for three distinct *stock keeping units (SKUs)*, i.e., distinguishable product items, which results in a granularity of 1 m and provides 1200 slots for SKUs (20 shelves per aisle × 3 × 20 shelf rows) in the warehouse in total. The depot is positioned at the lower-left corner of the warehouse. The batch capacity is $c = 2$. The number of items $k(j)$ in each order o_j is uniformly distributed in set the $\{1, 2, ..., \overline{k}\}, \overline{k} = 4$, which reflects a common scenario in e-commerce, where the typical order contains fewer than two items on average (c.f. Boysen et al., 2019; Xie et al., 2023). The storage locations of ordered items are uniformly distributed among all storage locations *(uniform dispersion)*. The orders arrive according to a homogeneous Poisson process with an arrival rate r. In preliminary experiments, we investigated different arrival rates r for shifts of 4h, increasing the rates in steps of 10 orders/shift, and found that optimal arrival rates in the *Base* setting, at which the system remains stable, are $r^{\text{Pcart}} = 90$ orders/shift and $r^{\text{Robot}} = 110$ orders/shift for *Pcart* and *Robot*, respectively. So, we set the arrival rates to these numbers. For *Pcart*, the 130 instances in *Base*, which form 13 groups of 10 instances according to their size n, are generated randomly and independently from each other. For better comparability, the 130 instances in *Base* for *Robot* are identical to those in *Pcart*, except for the sequence of arrival times, which was produced using a higher stable rate r^{Robot}. We generate the remaining nine settings, by taking the instances of *Base* and changing their specific characteristics in a one-factor-at-a-time design (see Table 2.2). For better control of the impact of warehouse characteristics, we keep the instances unchanged (including the coordinates of the generated items and the order arrival times) over the different settings, unless explicitly stated otherwise.

Table 2.2: Overview of the 10 settings of the data set

Changed factor	Settings		
Number of aisles	–	Base (10 aisles)	Largewarehouse (20 aisles)
Number of cross-aisles	Lesscrossaisles (2 cross-aisles)	Base (3 cross-aisles)	–
Storage policy	–	Base (uniform)	Classbaseddispersion (class-based)
Maximal order size	Smallorders (2 SKUs)	Base (4 SKUs)	Largeorders (8 SKUs)
Batching capacity	Smallbatches ($c = 1$)	Base ($c = 2$)	Largebatches ($c = 4$)
Arrival rate	Smallrate ($r^{\text{Pcart}} = 70$, $r^{\text{Robot}} = 90$)	Base ($r^{\text{Pcart}} = 90$, $r^{\text{Robot}} = 110$)	Largerate ($r^{\text{Pcart}} = 110$, $r^{\text{Robot}} = 130$)

We investigate different batch capacities in settings *Smallbatches* and *Largebatches*, where we set $c = 1$ and $c = 4$, respectively (all other instance data remained unchanged). In *Smallrate* and *Largerate*, the arrival rates are set to $r^{\text{Pcart}} = 70$, $r^{\text{Pcart}} = 110$ and $r^{\text{Robot}} = 90$ and $r^{\text{Robot}} = 130$ for *Robot* and *Pcart*, respectively. We do it by generating new arrival times for the instances in *Base*. We investigate warehouse geometries in *Largewarehouse* (20 aisles) and *Lesscrossaisles* (2 cross-aisles). Observe that the lengths and widths of the shelves, aisles, and cross-aisles, as well as the number of shelves per aisle, remain the same so that the size of the warehouse changes accordingly. In particular, in *Largewarehouse*, the warehouse has a length of 100 m (20 aisles $\times$ 3 m + $(1 + 19 \cdot 2 + 1)$ shelf rows $\times$ 1 m) and a width of 69 m is identical to *Base*; in *Lesscrossaisles*, the warehouse length of 50 m is identical to *Base*, and the width is 66 m (20 shelves $\times$ 3 m + 2 cross-aisles $\times$ 3 m). In *Largewarehouse*, we generate the positions of the ordered items anew. We do not have to do it in *Lesscrossaisles*, since the number of SKU slots remains the same (1200) and only the middle cross-aisle disappears. In *Classbaseddispertion*, SKUs are stored in the warehouse according to their turnover rates, so that we take instances of *Base* and generate new picking positions. We randomly assign ordered items to three classes: A, B, and C with probabilities of 52%, 36%, and 12%. Items in A refer to high-turnover products, which occupy slots in the first aisle. Items in B occupy aisles 2 to 4. And aisles in C are low-turnover products, which occupy aisles 5 to 10. We randomly assign a storage position to each ordered item in the respective aisles of its class. Finally, in *Smallorders* and *Largeorders*, the maximal order size is set to $\bar{k} = 2$ and $\bar{k} = 8$, respectively. For this, we have to generate the orders (including their size, positions of the items, and arrival times) anew in the instances of *Base*. The generated instances can be found in the Supplement.

We performed the experiments on the Compute Canada cluster by using 1 CPU with a time and memory limit of at most 1h and 350G, respectively. For each instance I, we compute $CIOPT(I)$ by solving the respective OBSRP* exactly with the dynamic programming approach described in Lorenz et al. (2024). We calculate $Reopt(I)$ by letting the orders arrive dynamically according to their arrival rates and applying exact reoptimization as described in Section 2.2.2 using the straightforwardly adapted dynamic programming approach of Lorenz et al. (2024).

Table 2.3: Worst observed optimality ratios of *Reopt* in *Pcart*

Setting	The number of orders $n =$												
	3	4	5	6	7	8	9	10	11	12	13	14	15
Base	1.18	1.14	1.16	1.12	1.15	1.09	1.20	1.08	1.14	1.14	1.07	1.10	1.11
Largewarehouse	1.21	1.15	1.15	1.18	1.18	1.12	1.18	1.15	1.11	1.12	1.13	1.12	*1.10
Lesscrossaisles	1.16	1.24	1.18	1.10	1.10	1.14	1.14	1.10	1.12	1.09	1.07	1.13	1.13
Classbaseddispersion	1.20	1.14	1.18	1.12	1.16	1.11	1.14	1.14	1.13	1.11	1.10	1.10	*1.08
Smallorders	1.34	1.33	1.22	1.11	1.16	1.08	1.16	1.08	1.07	1.09	1.23	1.10	1.09
Largeorders	1.18	1.21	1.10	1.08	1.12	*1.17	*1.10	–	–	–	–	–	–
Smallbatches	1.15	1.15	1.08	1.08	1.07	1.06	1.06	1.05	1.05	1.04	1.04	1.04	1.03
Largebatches	1.17	1.17	1.31	1.16	1.17	1.12	1.13	–	–	–	–	–	–
Smallrate	1.14	1.17	1.13	1.11	1.08	1.12	1.05	1.05	1.10	1.15	1.08	1.12	1.09
Largerate	1.19	1.14	1.19	1.21	1.12	1.11	1.13	1.12	1.12	1.14	1.13	1.11	*1.08
Total	1.34	1.33	1.31	1.21	1.18	1.17	1.20	1.15	1.14	1.15	1.23	1.13	1.13

Note. * Results for less than 10 (out of 10) instances. CIOPT did not solve 1 or 2 instances to optimality – No results reported, since CIOPT solved less than 8 instances to optimality.

2.4.2 Optimality ratios of *Reopt* in *Pcart* and *Robot*

For each instance I, we compute the *optimality ratio* as $\frac{Reopt(I)}{CIOPT(I)}$. Tables 2.3 and 2.4 report worst observed optimality ratios in *Pcart* and *Robot*. The average observed optimality ratios for *Pcart* and *Robot* can be found in Tables 2.5 and 2.6 in Supplement 2.7.7.

According to our experiments, *Reopt*'s results are rather close to optimality even in small-sized instances. For the smallest tested instance sizes (with the number of orders $n < 5$), the average ratio over all settings equaled 1.10 and 1.09, for *Pcart* and *Robot*, respectively; for the largest tested instance sizes ($n > 13$), it decreased to 1.05, for both cart types. The worst observed optimality ratio across *all* the instances equaled 1.34 for *Pcart* and 1.32 for *Robot*; and the respective ratios for larger instances ($n > 13$) decreased and equaled 1.13 and 1.16, respectively.

To statistically validate the monotonic decreasing relationship between the observed optimality ratios and the instance size n, we conducted a Spearman's rank correlation test (see e.g., Corder & Foreman, 2009). For each of the 10 settings, we computed the Spearman's correlation coefficient for the optimality ratios across all solved instances of different instance sizes (130 observations for most settings, fewer for Largeorders and Largebatches, see Tables 2.3 and 2.4). In *Pcart* (*Robot*), for all settings except Largeorders, the coefficients ranged from -0.68 to -0.25 (from -0.61 to -0.27) with p-values of less than 0.04 (less than 0.01) indicating a moderate but statistically significant decrease. For Largeorders, the Spearman's correlation coefficient was -0.23 (-0.16) and the p-value of 0.07 (0.20) may be explained by the lower number of observations and the limitation in instance size n.

2.5 Conclusion

This paper performs an optimality analysis of reoptimization policies for *the Online Order Batching, Sequencing, and Routing Problem (OOBSRP)*. The focus is on *immediate reoptimization (Reopt)*, where the current (partial) solution adjusts each time a new order arrives. We consider warehouses employing manual pushcarts and warehouses employing robotic carts. Given that picking instructions in traditional warehouses may only be transmitted at the depot, we additionally examine the *noninterventionist* version of *Reopt* for manual pushcarts. In total, we investigate three systems, each featuring a specific reoptimization policy and cart technology: *Robot* for interventionist *Reopt* and

Table 2.4: Worst observed optimality ratios of *Reopt* in *Robot*

Setting	The number of orders $n =$												
	3	4	5	6	7	8	9	10	11	12	13	14	15
Base	1.18	1.24	1.12	1.10	1.18	1.05	1.12	1.09	1.06	1.09	1.07	1.11	1.08
Largewarehouse	1.32	1.14	1.19	1.24	1.12	1.08	1.12	1.09	1.07	1.08	1.11	1.09	1.10
Lesscrossaisles	1.25	1.25	1.31	1.12	1.16	1.13	1.13	1.10	1.17	1.11	1.14	1.16	1.12
Classbaseddispersion	1.15	1.22	1.10	1.18	1.14	1.18	1.08	1.09	1.08	1.04	1.08	1.10	1.10
Smallorders	1.19	1.29	1.17	1.11	1.10	1.08	1.13	1.07	1.08	1.06	1.06	1.12	1.07
Largeorders	1.12	1.11	1.22	1.07	1.10	*1.12	*1.10	–	–	–	–	–	–
Smallbatches	1.22	1.24	1.19	1.06	1.15	1.08	1.14	1.11	1.09	1.07	1.10	1.06	1.06
Largebatches	1.18	1.24	1.15	1.09	1.11	1.11	1.15	*1.12	–	–	–	–	–
Smallrate	1.22	1.15	1.11	1.08	1.07	1.09	1.09	1.10	1.10	1.09	1.06	1.10	1.08
Largerate	1.25	1.12	1.23	1.08	1.17	1.09	1.15	1.10	1.09	1.09	1.13	1.11	*1.08
Total	1.32	1.29	1.31	1.24	1.18	1.18	1.15	1.12	1.17	1.11	1.14	1.16	1.12

Note. * Results for less than 10 (out of 10) instances. CIOPT did not solve 1 or 2 instances to optimality – No results reported, since CIOPT solved less than 8 instances to optimality.

robotic carts, *Pcart* for interventionist *Reopt* and manual pushcarts, as well as *Pcart-N* for noninterventionist *Reopt* and manual pushcarts.

OOBSRP can be interpreted as a nontrivial combination of the *online batching problem* and the *online traveling salesman* problem. To the best of our knowledge, there is currently no clarity on the performance gaps to optimality for *any* OOBSRP policy. *Reopt* is chosen for analysis due to its prominence in warehousing literature and its ability to leverage recent algorithmic progress for the offline version of OOBSRP.

Our extended competitive analysis demonstrates the *almost sure asymptotically optimal* performance of *Reopt* under very general stochastic assumptions in all three examined systems – *Robot*, *Pcart*, and *Pcart-N*. Our stochastic assumptions can describe a variety of order arrival patterns relevant for practice. On the formal side, they encompass arrival times that are modeled as order statistics or as a homogeneous Poisson process with specific arrival rates.

Even if we drop stochastic assumptions and examine *any* theoretically possible OOBSRP instances, *Reopt* performs well, with no policy improving its result by more than 50% in sufficiently large instances. Anomalies are possible in small OOBSRP instances, which cannot be overcome by *any* deterministic algorithm (see strict competitive ratios in Section 2.3.4). However, no policy can improve *Reopt*'s result by more than 75% regardless of the instance size.

Having established the almost sure *asymptotic* optimality of *Reopt* analytically under various stochastic scenarios, we show empirically that *Reopt* remains rather close to the complete-information optimality already in small instances in *Pcart* and *Robot*.

This paper provides insights into ongoing debates regarding the benefits of waiting and anticipation for online policies in warehouse operations. Indeed, the performance gaps of *Reopt*, which eschews both waiting and anticipation, serve as a benchmark for assessing the benefits of these concepts (see Section 2.1.2).

Several questions remain for future research, including the open status of Conjecture 1 and determining *stable rates* λ when order arrival times follow a Poisson process. Further performance analysis is needed for other online algorithms, including simple policies like 'first-come-first-served' and S-shape routing. An optimal online policy is still unknown, especially for small OOBSRP instances. Exploration of additional OOBSRP variants and extensions, including multiple pickers, picking location assignment, advanced information on orders, and order due dates is suggested as avenues for future research. Furthermore, the analysis of alternative objectives, particularly those related to the service level (e.g., tardiness), remains an outstanding research area.

2.6 Acknowledgments

We would like to extend our gratitude to the Government of Quebec (Ministère des Relations internationales et de la Francophonie) and BayFor for their support. We also acknowledge Compute Canada for providing the computational resources necessary for this research.

2.7 Supplemental material

2.7.1 Supplement: Proof of Lemma 1

Let us call an intersection of an outer aisle and an outer cross-aisle a *corner*. The shortest time to traverse all the picking locations of the warehouse starting from the given position s and ending in the given position t cannot be larger than the time of the following route (see Figure 2.8): Move from s to some corner of the warehouse, visit all the picking locations following an S-shape route starting from this *(first)* corner and ending in the respective *last* corner, after that move to t. Obviously, the S-shape route to visit all the picking locations of the warehouse takes time $(a \cdot W + L)$. By a smart selection of the *first* corner, where we start this S-shape route, the total time ζ to reach this first corner from s and then reach t from the *last* corner, does not exceed $(W + L)$.

Let w_s^{min} and w_t^{min} be the shortest horizontal distances from s and t to reach an outer cross-aisle, respectively (see Figure 2.8). Observe that $w_s^{min} \leq 0.5 \cdot W$ and $w_t^{min} \leq 0.5 \cdot W$. If $w_s^{min} \leq w_t^{min}$, the picker shall move from s along w_s^{min}, then:

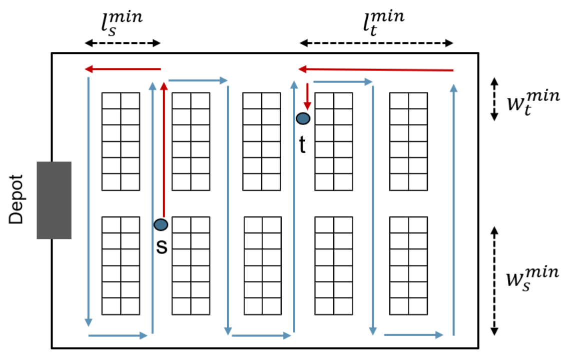
$$\text{vertical component of } \zeta \leq \ w_s^{min} + \max\{W - w_t^{min}, w_t^{min}\} \leq W \qquad (2.37)$$

If $w_s^{min} \geq w_t^{min}$, the picker shall select the first corner such, that she can move from the last corner to t along w_t^{min}, then:

$$\text{vertical component of } \zeta \leq \ \max\{W - w_s^{min}, w_s^{min}\} + w_t^{min} \leq W. \qquad (2.38)$$

By examining l_s^{min} and l_t^{min}, which are the shortest horizontal distances from s and t to reach an outer aisle, respectively, we receive that the horizontal component of $\zeta \leq L$.

Figure 2.8: Warehouse traversal starting in s and ending in t



2.7.2 Supplement: Proof of Theorem 2

Recall that $|Q^{Reopt}(I(n)^{rand}|$ is the queue length of available orders for *Reopt* at the arrival of the last order R_n. The assumption $\lim_{n\to\infty} \frac{|Q^{Reopt}(I(n)^{rand})|}{n} = 0$ a.s. implies:

$$\frac{Reopt(I(n)^{rand}) - R_n}{n} \leq \left(\frac{|Q^{Reopt}(I(n)^{rand})|}{c} + 1\right) \cdot \frac{u}{n} \xrightarrow{n\to\infty} 0 \quad \text{a.s.} \tag{2.39}$$

where u is the warehouse traversal constant from Lemma 1. The right-hand side refers to a feasible picking plan, when the cart is fully packed with c orders in each batch (if possible) and, to collect a batch, the picker traverses all the picking locations of the warehouse. Since the inter-arrival times of the customer orders $X_i, i \in \{1, ..., n\}$ are i.i.d., we can apply the Strong Law of Large Numbers:

$$\frac{R_n}{n} = \frac{\sum_{i=1}^{n} X_i}{n} \xrightarrow{n\to\infty} \frac{1}{\lambda} < \infty \quad \text{a.s.} \tag{2.40}$$

Putting (2.39) and (2.40) together:

$$\frac{Reopt(I(n)^{rand})}{CIOPT(I(n)^{rand})} \leq \frac{R_n + Reopt(I(n)^{rand}) - R_n}{R_n} \tag{2.41}$$

$$= 1 + \frac{Reopt(I(n)^{rand}) - R_n}{n} \cdot \frac{n}{R_n} \tag{2.42}$$

$$\xrightarrow{n\to\infty} 1 \qquad a.s. \tag{2.43}$$

2.7.3 Supplement: Proof of Theorem 3

This proof uses the Order Statistics Property of the homogeneous Poisson Process (cf. Feigin (1979)):

Proposition 6. *Suppose that order arrival times $R_1, R_2, ...$ follow a homogeneous Poisson process with rate λ. Let N be the number of arrival times in the given time interval $[0, t]$. Then, conditional on $N = n$, the successive arrival times $(R_1, ..., R_n)$ are distributed as the order statistics of n independent identically distributed random variables $(Y_i)_{i\in\mathbb{N}}$ with a uniform distribution on $[0, t]$.*

We use Proposition 6 to extend the analysis of Theorem 3 from the probability space described in Section 2.3.2.1 to the probability space of Section 2.3.2.2. Thereby, we use the following notation:

- Probability measures $\mathbb{P}_{os,unif(t)}$ for $t \in \mathbb{R}^+$, when the arrival times are independent uniformly$[0, t]$ -distributed random variables. Observe that in this case, order arrival times satisfy the order statistics property and, therefore, satisfy the conditions for Theorem 1.
- Probability measures $\mathbb{P}_{Pois(\lambda)}$ for $\lambda \in \mathbb{R}^+$, when the arrival times $R = (R_i)_{i\in\mathbb{N}}$ follow a Poisson Point process with rate λ.

We denote the optimality ratio of *Reopt* for any given random instance I^{rand} in a given picking system (*Pcart-N*, *Pcart* or *Robot*) as the respective measurable function $f : \Omega \to [1, \infty), I^{rand} \mapsto \frac{Reopt(I^{rand})}{CIOPT(I^{rand})}$, where Ω is the instance space.

The proof proceeds in four steps.

First, we start with independently and uniformly $[0,t]$ -distributed order arrivals. The *almost sure (a.s.)* asymptotic convergence of $f(I(n)^{rand})$ towards 1 (cf. Theorem 1) directly implies the *convergence in probability* of $f(I(n)^{rand})$ for $n \to \infty$ (cf. Proposition 10.3 in Le Gall (2022)). In *Pcart-N*, *Pcart* and *Robot*, the ratio $f(I(n)^{rand})$ is bounded for all $n \in \mathbb{N}$ (see Proposition 4), so is the expected value $\mathbb{E}_{os,unif(t)}[f(I(n)^{rand}] \leq \sigma(Reopt) < \infty$. Therefore, the convergence in probability implies the *convergence in mean* (see Proposition 10.4 in Le Gall (2022)):

$$\lim_{n\to\infty} \mathbb{E}_{os,unif(t)}[f(I(n)^{rand})] = 1 \qquad \forall t \in \mathbb{R}^+ \tag{2.44}$$

Second, suppose the arrival times follow a Poisson process with some rate $\lambda \in \mathbb{R}^+$ and let $I(t)^{rand}$ consist of all orders that arrive in the fixed time interval $[0,t], t \in \mathbb{R}^+$. Let denote the number of these orders as $N(t) \in \mathbb{N}$ and observe that $N(t)$ a Poisson-$(\lambda \cdot t)$-distributed random variable. Then by Proposition 6:

$$\mathbb{E}_{Pois(\lambda)}[f(I(t)^{rand}) \; | N(t) = n] = \mathbb{E}_{os,unif(t)}[f(I(n)^{rand})] \tag{2.45}$$

Thus, for any $\lambda \in \mathbb{R}^+$ and any $t \in \mathbb{R}^+$, by (2.44):

$$\lim_{n\to\infty} \mathbb{E}_{Pois(\lambda)}[f(I(t)^{rand}) \mid N(t) = n] = 1 \tag{2.46}$$

The convergence in (2.46) implies that for any $\epsilon > 0$, there exists $n_\epsilon \in \mathbb{N}$ such that

$$\mathbb{E}_{Pois(\lambda)}[f(I(t)^{rand}) \mid N(t) = n] \leq 1 + \frac{\epsilon}{2} \qquad \forall n > n_\epsilon \tag{2.47}$$

Third, by applying the Law of Total Expectation, we compute the *unconditional* mean by examining groups of instances with $N(t) > n_\epsilon$ and with $N(t) \leq n_\epsilon$:

$$\begin{aligned}\mathbb{E}_{Pois(\lambda)}[f(I(t)^{rand})] = {} & \mathbb{P}_{Pois(\lambda)}(N(t) \leq n_\epsilon) \cdot \; \mathbb{E}_{Pois(\lambda)}[f(I(t)^{rand}) \mid N(t) \leq n_\epsilon] \\ & + \sum_{n=n_\epsilon+1}^{\infty} \mathbb{P}_{Pois(\lambda)}(N(t) = n) \cdot \; \mathbb{E}_{Pois(\lambda)}[f(I(t)^{rand}) \mid N(t) = n]\end{aligned} \tag{2.48}$$

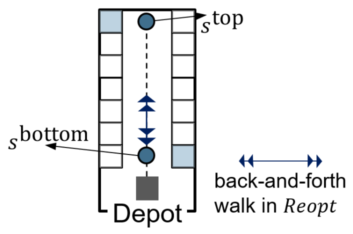
By Proposition 4 on the strict competitive ratio, $\mathbb{E}_{Pois(\lambda)}[f(I(t)^{rand}) \mid N(t) \leq n_\epsilon] \leq \sigma(Reopt) < \infty$. Moreover, the probability mass $\mathbb{P}_{Pois(\lambda)}(N(t) > n_\epsilon) \leq 1$ by definition. Hence:

$$\begin{aligned}\mathbb{E}_{Pois(\lambda)}[f(I(t)^{rand})] \leq {} & \sigma(Reopt) \cdot \mathbb{P}_{Pois(\lambda)}(N(t) \leq n_\epsilon) \\ & + \; (1 + \frac{\epsilon}{2}) \cdot \mathbb{P}_{Pois(\lambda)}(N(t) > n_\epsilon) \qquad (2.49) \\ \leq {} & \sigma(Reopt) \cdot \mathbb{P}_{Pois(\lambda)}(N(t) \leq n_\epsilon) + (1 + \frac{\epsilon}{2}) \qquad (2.50)\end{aligned}$$

Observe that n_ϵ does not depend on the arrival rate λ, but λ influences the probability of instances with $N(t) \leq n_\epsilon$. Let $\lambda_i > 0, i \in \mathbb{N}$, be any increasing sequence of arrival rates such that $\lambda_i \to \infty$ if $i \to \infty$. Recall that the cumulative distribution function of the Poisson distribution is the *regularized gamma function* Q, i.e., $\mathbb{P}_{Pois(\lambda)}(N(t) \leq n_\epsilon) = Q(n_\epsilon, \lambda \cdot t)$. By definition of Q for fixed n_ϵ, there exists an $i_\epsilon \in \mathbb{N}$ such that for all $i \geq i_\epsilon : \sigma(Reopt) \cdot \mathbb{P}_{Pois(\lambda)}(N(t) \leq n_\epsilon) = \sigma(Reopt) \cdot Q(n_\epsilon, \lambda_i \cdot t) \;\; \leq \frac{\epsilon}{2}$. After implementing this in (2.50):

$$\mathbb{E}_{Pois(\lambda_i)}[f(I(t)^{rand})] \leq 1 + \epsilon \qquad \forall i \geq i_\epsilon \tag{2.51}$$

Figure 2.9: *Robot*: Unfortunate large instance for *Reopt*



Note. $c = 2$. Two-item orders, each with one item s^{bottom} and one item s^{top}, arrive such that the picker in *Reopt* keeps walking back and forth without completing any batch until the arrival of the last order.

Since (2.51) is valid for any $\epsilon > 0$, we proved that for any increasing sequence $(\lambda_i)_{i\in\mathbb{N}}$ of arrival rates with $\lambda_i \to \infty$ if $i \to \infty$

$$\lim_{i\to\infty} \mathbb{E}_{Pois(\lambda_i)}[f(I(t)^{rand})] \to 1 \tag{2.52}$$

Finally, the second convergence statement of Theorem 3 follows by Markov's inequality for the nonnegative random variable $(f(I(t)^{rand}) - 1)$ and any given $\epsilon > 0$:

$$\mathbb{P}_{Pois(\lambda_i)}(f(I(t)^{rand}) \geq 1 + \epsilon) \leq \frac{\mathbb{E}_{Pois(\lambda_i)}[f(I(t)^{rand}) - 1]}{\epsilon} \to 0 \qquad \text{if } i \to \infty \tag{2.53}$$

2.7.4 Supplement: Auxiliary Lemma on the asymptotic competitive ratio

Lemma 5. *In Pcart and Robot, the following holds true:*

$$lim_{n\to\infty} \frac{Reopt(I(n))}{CIOPT(I(n))} = \alpha \tag{2.54}$$

$$\textit{together with} \quad \lim_{n\to\infty} CIOPT(I(n)) \to \infty \tag{2.55}$$

implies that $\sigma_{asymp.}(Reopt) \geq \alpha$.

Proof. In a proof by contradiction, suppose that for all $n \in \mathbb{N}$ and for all instances $I(n)$, $Reopt(I(n)) \leq (\alpha - \epsilon)CIOPT(I(n)) + const$ for two constants $const, \epsilon > 0$. Then $\frac{Reopt(I(n))}{CIOPT(I(n))} \leq (\alpha - \epsilon) + \frac{const}{CIOPT(I(n))})$ for all $n \in \mathbb{N}$. But since $\frac{const}{CIOPT(I(n))} \xrightarrow{n\to\infty} 0$, this is a contradiction to $lim_{n\to\infty} \frac{Reopt(I(n))}{CIOPT(I(n))} = \alpha$. □

2.7.5 Supplement: Proof of Proposition 3

Figure 2.9 provides an unfortunate example I^{worst} for *Robot* with $c = 2$ in a one-aisle warehouse; the example can be generalized for $c > 2$ and more general warehouse layouts.

Arriving orders in I^{worst} consist of two items each: s^{bottom} (placed at the bottom of the aisle) and s^{top} (placed on the top of the aisle). Recall that W is the width of the warehouse and l^d denotes the location of the depot. Let $d(l_d, s^{\text{bottom}}) = 2\epsilon$, $(l_d, s^{\text{top}}) = W$, and $(s^{\text{bottom}}, s^{\text{top}}) = W - 2\epsilon$.

Instance I^{worst} consists of $n = 2k + 1$ orders, $k \in \mathbb{N}$. The idea is to set the inter-arrival times of the orders such that *Reopt* makes the picker oscillate between s^{bottom} and the middle of the aisle without picking any item until the last order arrives.

The first order arrives at $r_1 = 2\epsilon$, the picker in *Reopt* picks item s^{bottom} of order o_1 at time 4ϵ.

At time $r_2 = \frac{1}{2}W + 3\epsilon - \frac{\delta}{2n-2}$, o_2 arrives and the picker in *Reopt* is at distance $\frac{1}{2}W - \epsilon - \frac{\delta}{2n-2}$ from s^{bottom}, with s^{bottom} of order o_1 placed in her cart. *Reopt* has two choices. The first one is to return to s^{bottom} and batch the new order o_2 together with o_1 at the total cost of $r_2 + \frac{1}{2}W - \epsilon - \frac{\delta}{2n-2} + W - 2\epsilon = 2W - 2 \cdot \frac{\delta}{2n-2}$. The second one is to continue to s^{top} and pick o_1 and o_2 in separate batches at the total cost of $2\epsilon + W + (W - 2\epsilon) = 2W$. *Reopt* decides for the first option.

At time $r_3 = W + 2\epsilon - 3\frac{\delta}{2n-2}$, o_3 arrives and the picker in *Reopt* is at distance $\frac{\delta}{2n-2}$ from s^{bottom}. Now *Reopt* has two choices. The first one is to move up and finish order o_1 in a separate batch by picking s^{top}, then collect orders o_2 and o_3 together at the total cost of $r_3 + W - 2\epsilon - \frac{\delta}{2n-2} + W - 2\epsilon = 3W - 2\epsilon - 4\frac{\delta}{2n-2}$. The second one is to collect item s^{bottom} of o_2, finish batch $S_2 = \{o_1, o_2\}$ and pick o_3 separately at the total cost of $r_3 + \frac{\delta}{2n-2} + 2(W - 2\epsilon) = 3W - 2\epsilon - 2\frac{\delta}{2n-2}$. *Reopt* will decide for the first option.

In general, at time $r_{2p} = \frac{2p-1}{2}W - (2p-5) \cdot \epsilon - (4p-3) \cdot \frac{\delta}{2n-2}, p \in \{2, 3, ..., k\}$, the picker in *Reopt* is at distance $\frac{1}{2}W - \epsilon - \frac{\delta}{2n-2}$ from s^{bottom}. Now *Reopt* has two choices. The first one is to return to s^{bottom} to batch the new order o_{2p} with the commenced order o_1, then pick all the remaining orders in pairs. The second one is to continue to s^{top} to finish the commenced batch $S_1 = \{o_1\}$, then pick the remaining orders in $p-1$ pairs and the last order in a separate batch. *Reopt* decide for the first option with the total cost of $r_{2p} + \frac{1}{2}W - \epsilon - \frac{\delta}{2n-2} + p(W - 2\epsilon) = 2pW - (4p-4)\epsilon - (4p-2)\frac{\delta}{2n-2}$.

At time $r_{2p+1} = pW - (2p-4)\epsilon - (4p-1) \cdot \frac{\delta}{2n-2}, p \in \{2, 3, ..., k\}$, the picker is at distance $\frac{\delta}{2n-2}$ from s^{bottom}. *Reopt* has two choices. The first one is to change the course and move up to s^{top} to finish order o_1 in a separate batch, then pick the rest of the orders in pairs. The second one is to pick up item s^{bottom} of o_2, finish batch $S_1 = \{o_1, o_2\}$, then pick all orders in pairs except from batch $S_{p+1} = \{o_{2p+1}\}$, which is picked separately. *Reopt* will decide for the first option at the total cost of $r_{2p+1} + W - 2\epsilon - \frac{\delta}{2n-2} + p(W - 2\epsilon) = (2p+1)W - (4p-2)\epsilon - 4p \cdot \frac{\delta}{2n-2}$.

Overall, to pick $n = 2k+1$ orders of I^{worst}, *Reopt* requires $Reopt(I^{\text{worst}}) = (2k+1)W - (4k-2)\epsilon - 4k \cdot \frac{\delta}{2n-2} = (2k+1)W - (4k-2)\epsilon - \delta$ time. $CIOPT$ will pick the first order in a separate batch, which is completed at time W, and collect the remaining orders in pairs, so that $CIOPT(I^{\text{worst}}) = W + k \cdot (W - 2\epsilon) = (k+1)W - 2k\epsilon$. It follows that $\frac{Reopt(I^{\text{worst}})}{CIOPT(I^{\text{worst}}))} = 2 - \frac{W - 2\epsilon + \delta}{CIOPT(I^{\text{worst}})} \xrightarrow{k \to \infty} 2$ as $CIOPT(I^{\text{worst}}) \xrightarrow{k \to \infty} \infty$. This on the other hand, implies that $\sigma_{asymp}(Reopt) \geq 2$ by Lemma 5.

2.7.6 Proof of Proposition 5

Theorem 3.1 for N-TSP and Theorem 3.3 for H-TSP by Ausiello et al. (2001) consider the case when requests are located on the *real line*.

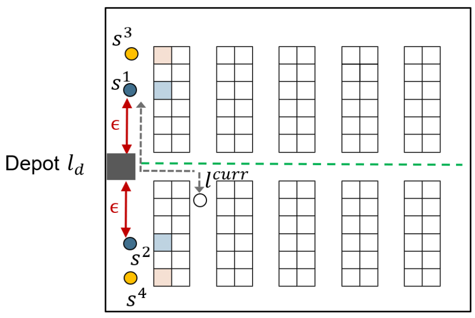
What remains is the translation of the specific positioning of the picking locations s^1 and s^2, from the real line to the warehouse, such that:

- They are at equal distance to the depot $d(l_d, s^1) = d(l_d, s^2) = \epsilon$.
- *Any* other location in the warehouse is at a larger distance to one of the picking locations:

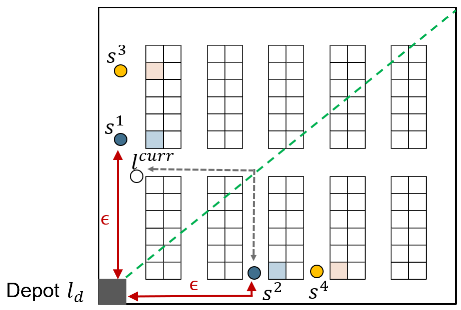
$$\max\{d(l^{curr}, s^1), d(l^{curr}, s^2)\} \geq \epsilon \tag{2.56}$$

Observe that $d()$ refers to the warehouse metric as explained in Section 2.2.

Figure 2.10: Positioning of the potential order locations s^1 and s^2 in Proposition 5 for *Robot* and *Pcart*



(a) Warehouse with the depot *not* located in the corner



(b) Warehouse with the depot in the corner

Note. Observe that $d(s^1, l_d)) = d(s^2, l_d) = \epsilon$ for some ϵ, which depends on the location of l^d. The dotted green line separates the warehouse in two sections. For each location l^{curr} in the same section as s^1, $d(l^{curr}, s^2) \geq \epsilon$ and for each location l^{curr} in the same section as s^2, $d(l^{curr}, s^1) \geq \epsilon$.

Figure 2.10 explains, how to position s^1 and s^2 if the depot l^d is located *i)* in a corner of the warehouse (Figure 2.10a) or *ii)* in any other location (Figure 2.10b).

To prove (2.56), compare $d()$ with the Manhattan metric $m()$:

- Obviously $d(l, l') \geq m(l, l')$ for any locations l, l' in the warehouse.
- The green dotted line in Figure 2.10 depicts locations l that are *Manhattan-equidistant* to s^1 and s^2 at a distance of *at least* ϵ: $m(l, s^1) = m(l, s^2) \geq \epsilon$.
- The green dotted line divides the warehouse into two *sections*: s^1 is in section 1 and s^2 is in section 2. If the picker's location l^{curr} is in section 1, the picker must cross the line to reach s^2 and vice-versa. Thus, for any location l^{curr} in the warehouse, $\max\{d(l^{curr}, s^1), d(l^{curr}, s^2)\} \geq \max\{m(l^{curr}, s^1), m(l^{curr}, s^2)\} \geq \epsilon$.

The case of *Robot*. The main idea of Theorem 3.1 for N-TSP is that the upcoming request (which we can interpret as a single-item order) is located in one of two locations – s^1 and s^2 – placed on opposite sides of the depot l^d at distance $d(l_d, s^1) = d(l_d, s^2) = \epsilon$. For any picker position l^{curr} on the *real line*, $\max\{d(l^{curr}, s^1), d(l^{curr}, s^2)\} \geq \epsilon$. Now, consider any algorithm ALG and the respective picker location l^{curr} at time r_1. There is an instance I^{worst} defined on the *real line*, in which request o_1 appears to the left of l_d if the picker's position l^{curr} is to the right of it, and *vice versa*; i.e., ALG is on the 'wrong side' of the depot in this instance. $CIOPT$ has complete information about the location of o_1 and moves immediately in the right direction. In *Robot*, an unfortunate instance I^{worst} in a warehouse of any geometry, with $n = 1$ order and $r_1 = \epsilon$ can be constructed by the same scheme as just described for of N-TSP. Indeed, by this scheme, $CIOPT(I^{worst}) = \epsilon$ and $ALG(I^{worst}) = \epsilon + \max\{d(l^{curr}, s^1), d(l^{curr}), s^2)\} \geq 2\epsilon$.

The case of *Pcart*. In *Pcart*, the proof uses the same positional scheme for the picking locations s^1 and s^2, and is *adapted* from the proof of Theorem 3.3 from Ausiello et al. (2001). It constructs seven unfortunate instances I^{w1} to I^{w7} depending on the decisions of the deterministic algorithm ALG:

- Instances I^{w1} and I^{w2}, each with one $(n = 1)$ single-item order arriving at $r_1 = \epsilon$. The order's position is at s^1 in I^{w1} and s^2 in I^{w2}, respectively.
- Instances I^{w3} and I^{w4}, each with two $(n = 2)$ orders arriving at $r_1 = \epsilon$ and $r_2 = 3\epsilon$. The first order has two items at positions s^1 and s^2 in both instances. The second order has one item at position s^1 in I^{w3} and s^2 in I^{w4}, respectively.

- Instance I^{w5} with one ($n = 1$) order arriving at $r_1 = \epsilon$ with two items at positions s^1 and s^2.
- Instances I^{w6} and I^{w7} with two orders ($n = 2$). The first order has the same two items s^1 and s^2 and arrival time $r_1 = \epsilon$ as in I^{w5}. The second order o_2 arrives at time $r_2 = (3+q)\epsilon$, where $0 \leq q < 4\left(\frac{9+\sqrt{17}}{8}\right) - 5 < 1.6$ and is computed as explained below. In I^{w6}, order o_2 has one item at position s^3 located in the direction of s^1 away from the depot with $d(s^3, l^d) = (1+q)\epsilon, d(s^3, s^1) = q\epsilon$ and $d(s^3, s^2) = (2+q)\epsilon$ (see Figure 2.10). In I^{w7}, the single item of order o_2 is at position s^4, which is constructed along the same lines as s^3, but in the direction of s^2 away from the depot.

In all these instances, $CIOPT$ has complete information. If $c \geq 2$,

$$CIOPT(I^{w1}) = CIOPT(I^{w2}) = 2\epsilon \tag{2.57}$$

$$CIOPT(I^{w3}) = CIOPT(I^{w4}) = 4\epsilon \tag{2.58}$$

$$CIOPT(I^{w5}) = 4\epsilon \tag{2.59}$$

$$CIOPT(I^{w6}) = CIOPT(I^{w7}) = (4+2q)\epsilon \tag{2.60}$$

In contrast, no algorithm ALG can differentiate between the instances I^{w1} to I^{w7} before time $t = \epsilon$, between instances I^{w3} to I^{w7} before $t = 3\epsilon$, and between instances I^{w5} to I^{w7} before $t = (3+q)\epsilon$. The case-by-case proof of Ausiello et al. (2001) shows that any deterministic ALG reaches the ratio of $\frac{ALG(I)}{CIOPT(I)} \geq \frac{9+\sqrt{17}}{8} \sim 1.64$ for at least one of the instances described above.

In the proof, we keep track of the distance from the current picker position l^{curr} in some arbitrary algorithm ALG to the *green dotted line* ($line$) (see Figure 2.10), which is the *shortest* distance from l^{curr} to any point on this line.

Let us denote $\sigma(ALG)$ simply as σ. Assume that ALG has $\sigma < (9+\sqrt{17})/8$. Then, in instances $I \in \{I^{w1}, I^{w2}\}$, the result of ALG should be less than $\sigma \cdot CIOPT(I) = 2\epsilon\sigma$ (see (2.57)). To achieve this, at time $t = \epsilon$, the picker must be at a point l_1^{curr} of distance $\leq (2\sigma - 3)\epsilon$ from $line$, since she has to cross the *line* to pick the item s^1 (in I^{w1}) or s^2 (in I^{w2}) and then return to the depot, which takes her at least $\epsilon + d(line, l_1^{curr}) + \epsilon + \epsilon$ time (see (2.56)).

Similarly, in instances $I \in \{I^{w3}, I^{w4}\}$, the result of ALG should be less than $\sigma \cdot CIOPT(I) = 4\epsilon\sigma$ (see (2.58)). Observe that at $t = 3\epsilon$, given l_1^{curr} and $r_{1,2} = \epsilon$, ALG was not able to pick at least one of the items – s^1 or s^2. *W.l.o.g*, let it be s^1. Then at time $t = 3\epsilon$, the picker must be at a point l_2^{curr} of distance $\geq (7 - 4\sigma)\epsilon$ from $line$ (see Ausiello et al. (2001) for more details).

It remains to examine instances $I \in \{I^{w5}, I^{w6}, I^{w7}\}$. Recall that the picker's position l_2^{curr} is situated at some distance $\geq (7 - 4\sigma)\epsilon$ from the $line$ at time $t = 3\epsilon$, thus can be shown that she has to cross the $line$ at least once to collect the remaining items as follows. Since, as explained above, the picker cannot have picked both items of the first order (s^1 and s^2) at $t = 3\epsilon$, there are two cases:

- The picker has picked neither s^1 nor s^2, then she will have to cross the $line$ at least once to serve the first order.
- $W.l.o.g.$, the picker has picked s^2. Then, if we look at both warehouse sections separated by the $line$, the picker has to be located at the part with s^2 from the line. Indeed, she started moving to s^2 at time $t = \epsilon$ being at the other side of the $line$ at a distance of at most $(2\sigma - 3)\epsilon$. Even if she moved directly to s^1 after having visited

s^2, at time $t = 3\epsilon$, she is at a distance of at most $3\epsilon - \epsilon - (\epsilon + (2\sigma - 3)\epsilon) = (\epsilon - (2\sigma - 3)\epsilon) < \epsilon$ from s^2. Therefore, also in this case, the picker has to cross the $line$ to collect the remaining item of the first order.

To sum up, in both cases in ALG, the picker crosses the $line$ at some time after $t = 3\epsilon$. Let denote this time $(3+q)\epsilon$. Since the result of ALG should be less than $\sigma \cdot CIOPT(I^{w5}) = 4\epsilon\sigma$, the picker should cross the $line$ at no later time than $(4\epsilon\sigma - 2\epsilon)$. Therefore, we have

$$q \leq 4\sigma - 5 \tag{2.61}$$

Finally, if in ALG the picker crosses the $line$ not later than time $(3+q)\epsilon = (4\sigma - 2)\epsilon$, see (2.61), then at least in one of the instances $I \in \{I^{w6}, I^{w7}\}$, $ALG(I) \geq (7+3q)\epsilon$ (see (2.60)), whereas $CIOPT = (4+2q)\epsilon$. The least value of σ that satisfies $\sigma \geq \frac{(7+3q)\epsilon}{(4+2q)\epsilon}$ given (2.61) is $\sigma = \frac{9+\sqrt{17}}{8}$.

2.7.7 Supplement: Average observed optimality ratios of *Reopt* in *Pcart* and *Robot*

Tables 2.5 and 2.6 report the average observed optimality ratios of *Reopt*, for the experiments described in Section 2.4.2.

Table 2.5: Average optimality ratios of *Reopt* in *Pcart*

Setting	The number of orders $n =$												
	3	4	5	6	7	8	9	10	11	12	13	14	15
Base	1.11	1.07	1.08	1.06	1.07	1.06	1.07	1.05	1.07	1.06	1.05	1.06	1.06
Largewarehouse	1.13	1.11	1.09	1.09	1.08	1.07	1.08	1.10	1.06	1.06	1.08	1.08	*1.06
Lesscrossaisles	1.11	1.10	1.07	1.07	1.06	1.07	1.07	1.05	1.07	1.04	1.05	1.07	1.07
Classbaseddispersion	1.11	1.07	1.08	1.06	1.08	1.06	1.07	1.05	1.03	1.04	1.05	1.05	*1.04
Smallorders	1.16	1.11	1.09	1.07	1.07	1.05	1.06	1.04	1.04	1.04	1.06	1.05	1.05
Largeorders	1.10	1.07	1.07	1.04	1.06	*1.05	*1.08	–	–	–	–	–	–
Smallbatches	1.10	1.07	1.06	1.05	1.04	1.04	1.04	1.03	1.03	1.03	1.03	1.02	1.02
Largebatches	1.10	1.09	1.12	1.09	1.07	1.06	1.08	–	–	–	–	–	–
Smallrate	1.10	1.11	1.08	1.06	1.05	1.06	1.03	1.04	1.04	1.06	1.04	1.06	1.05
Largerate	1.13	1.09	1.09	1.08	1.06	1.06	1.07	1.07	1.07	1.07	1.07	1.06	*1.04
Total	1.12	1.09	1.08	1.07	1.07	1.06	1.06	1.05	1.05	1.05	1.05	1.06	1.05

Note. * Results for less than 10 (out of 10) instances. CIOPT did not solve 1-2 instances to optimality.
– No results reported, since CIOPT solved less than 8 instances to optimality.

Table 2.6: Average observed optimality ratios of *Reopt* in *Robot*

Setting	The number of orders $n =$												
	3	4	5	6	7	8	9	10	11	12	13	14	15
Base	1.09	1.09	1.07	1.05	1.08	1.04	1.05	1.05	1.04	1.04	1.05	1.05	1.04
Largewarehouse	1.10	1.09	1.09	1.08	1.06	1.05	1.06	1.05	1.04	1.04	1.06	1.05	1.06
Lesscrossaisles	1.11	1.11	1.11	1.07	1.09	1.07	1.06	1.06	1.07	1.05	1.07	1.07	1.06
Classbaseddispersion	1.08	1.10	1.05	1.05	1.08	1.05	1.03	1.04	1.03	1.03	1.04	1.05	1.05
Smallorders	1.11	1.10	1.08	1.06	1.06	1.04	1.05	1.04	1.04	1.03	1.03	1.04	1.03
Largeorders	1.07	1.06	1.08	1.05	1.06	*1.07	*1.05	–	–	–	–	–	–
Smallbatches	1.10	1.08	1.07	1.05	1.07	1.05	1.04	1.04	1.05	1.04	1.05	1.04	1.03
Largebatches	1.09	1.08	1.08	1.06	1.07	1.04	1.05	*1.05	–	–	–	–	–
Smallrate	1.07	1.08	1.07	1.04	1.04	1.04	1.04	1.04	1.04	1.04	1.03	1.04	1.04
Largerate	1.12	1.08	1.09	1.04	1.08	1.06	1.06	1.05	1.06	1.05	1.06	1.06	*1.04
Total	1.09	1.09	1.08	1.06	1.07	1.05	1.05	1.05	1.05	1.04	1.05	1.05	1.04

Note. * Results for less than 10 (out of 10) instances. CIOPT did not solve 1-2 instances to optimality.
– No results reported, since CIOPT solved less than 8 instances to optimality.

Chapter 3

On picking operations in e-commerce warehouses: Insights from the complete-information counterpart

Catherine Lorenz, Alena Otto, Michel Gendreau

Abstract. A key factor in the success of major e-commerce players like Amazon or Zalando, is their ability to achieve the delicate balance between fast- and low-cost deliveries, enabling them to fulfill promises of same-day deliveries. To accomplish this, they rely on intelligent optimization algorithms that process incoming orders in a real-time (*online*) fashion, paired with advanced anticipation techniques like AI to forecast certain characteristics of future orders. At the level of warehousing operations, there exists so far no unbiased benchmark to assess the potential of such dynamic algorithms similar to optimality gaps in static optimization, as the computation of *optimal online solutions* suffers from the curse of dimensionality and therefore is intractable.

In this paper, we design a *perfect anticipation algorithm* that computes CIOSs (Complete-Information Optimal policy Solutions), an exact optimal algorithm under perfect information on future customer orders, including ordered items and arrival times, for picking operations in widespread *picker-to-parts* warehouses. We analyze CIOSs under two common objectives: makespan (minimizing costs representing picker's working time and wages) and average order turnover (minimizing speed of delivery), to identify those warehouse settings where simple planning heuristics perform well and those where significant improvements can be gained through investment in advanced anticipation mechanisms. We creatively leverage CIOSs to uncover decision patterns that can enhance simpler algorithms suited for uncertain environments. A central aspect of analyzing CIOSs is to gain clarity on the role and configuration of *strategic waiting*, which imposes deliberate delays in picking to improve future operations. Strategic waiting is regarded as one of the most promising tools to enhance the efficiency of online operations, simultaneously being one of the least understood concepts in the warehousing literature. Surprisingly, our detailed analysis of CIOSs downplays the importance of strategic waiting and explains why this may be the case. Instead, it offers actionable advice to significantly improve simultaneously operation costs and order throughput time in e-commerce picking based on fundamental concepts, that have been largely overlooked in the literature. To compute CIOSs, we designed a customized dynamic programming algorithm (DP) for the integrated Order Batching, Sequencing, and picker Routing Problem with Release times (OBSRP-R), which is the first exact algorithm for OBSRP-R in the literature.

3.1 Introduction

Worldwide warehousing operations cost businesses about €300 billion annually (Herrmann et al., 2019). E-commerce, a prevalent segment, presents challenges due to *dynamically arriving* customer orders and customer demands for both free and fast deliveries (Henderson, 2020). Meeting these demands is expensive as it necessitates allocating more resources to handle customer requests individually, thereby sacrificing economies of consolidation and scale in warehousing and logistics. Staying *low-cost, while being fast* is a challenge. Not surprising many express delivery start-ups went bankrupt (Meyersohn, 2023). Conversely, Amazon has successfully reduced delivery costs and speed, boosting its sales and profits and, partially because of this, becoming one of the world's largest companies by market capitalization alongside Alphabet, Microsoft, Apple, and Nvidia (Palmer, 2024). Amazon's CEO, Andy Jassy, stated: "We've re-evaluated every part of our fulfillment network over the last year. We obviously like the results, but don't think we've fully realized all the benefits yet" (Drummer, 2023). Amazon is continuing to expand its same-day delivery services globally (Herrington, 2024).

Achieving a delicate balance between fast and low-cost deliveries requires the use of intelligent optimization algorithms and AI in operations planning and control. Companies like Zalando and Amazon use AI to forecast and influence upcoming customer orders by leveraging intelligent recommendations and search result sorting (Chan & Wanab, 2024; S. O'Neill, 2024; Steinker et al., 2017). This allows them to *anticipate* certain characteristics of incoming orders. Emerging paradigms enable the direct integration of these predictions, along with contextual information, into optimization algorithms, thereby unlocking their combined potential (cf. Sadana et al., 2024). To asses the improvement potential of such algorithms, similar to *optimality gaps* in static optimization, we need a *perfect anticipation algorithm* that computes *CIOSs (Complete-Information Optimal policy Solutions)*. With CIOSs, we can identify warehouse settings where existing simple planning heuristics are close to the best achievable result, and those where significant improvements can be achieved by further investing in advanced anticipation mechanisms.

Another key motivation for designing a perfect-anticipation policy, which is an *optimal* policy in case of perfect anticipation, is the *limited knowledge* on optimal policies for any type of warehousing operations with dynamically incoming orders. This is not surprising, as optimal policies for many dynamic optimization problems are unknown due to their inherent complexity, known as the *curse of dimensionality*. (cf. Powell, 2011, for a detailed discussion). The goal of seeking optimal policies isn't solely to obtain them – since they are often computationally impractical for real-world instances – but to uncover decision patterns that can enhance simpler algorithms suited for uncertain environments (Wang et al., 2019). With this in mind, we analyze CIOSs to derive such insights.

This paper designs a *perfect-anticipation algorithm* that computes CIOSs, an *exact* optimal algorithm under perfect information on future customer orders, including ordered items and arrival times, for *warehouse picking operations*. We focus on operations inside *picker-to-parts warehouses*, where a worker travels to collect inventory items (see Figure 3.1). Despite recent advances in automation (see Azadeh et al., 2019, for a survey), these warehouses remain prevalent (Napolitano, 2012; Vanheusden et al., 2023) due to their superior flexibility in handling demand fluctuations, such as Black Friday (Boysen et al., 2019).

A central aspect of analyzing CIOSs is gaining clarity on the role and configuration of *strategic waiting*, which involves deliberate delays imposed to improve future operations. For example, consider picking operations from receiving orders to delivering them to the depot for packaging. These operations can be modeled as a queuing system with stochastic order arrivals. Queuing theory shows that all stable systems, even at full capacity, experience idle times and short queues due to variability in arrival times (Abouee-Mehrizi & O.Baron, 2016). Figure 3.2 illustrates that allowing the picker to wait at the depot when the queue is short – despite available picking requests – can significantly reduce picking completion time and improve the average order turnover. Although various policies for strategic waiting have been discussed in the literature (see Gil-Borrás et al., 2024, for a summary), findings are mixed. The recent survey of Pardo et al. (2024) describes strategic waiting as the *'by far least studied activity'*, which is expected to have *'a deep influence on the performance of the overall [planning] method [for warehousing operations]'* .

This paper's analysis relies on insights from perfect-anticipation algorithms with long computational times, as the studied optimization problems are hard-to-solve (NP-hard in the strong sense, see Section 3.3). Therefore, we had to narrow our focus to the most essential aspects:

- We focus on the *picking operations* in a warehouse, from order arrival until order delivery to the depot for further packaging and delivery, highlighting the *sort-while-pick* system, where all items of a customer order are collected into the same bin. We examine different classes of policies ranging from fully integrated to hierarchically-taken decisions on *order batching* (= grouping customer orders into picking lists served in one picking tour), *batch selection* (=selecting the picking list to be processed next), and *picker routing*. We analyze two common objective functions: *completion time, or makespan* (cost-oriented, focusing on picker's time) and *average order turnover time* (service-oriented, focused on fast deliveries). For a detailed discussion of warehousing objectives, we refer the reader to C.-M. Chen et al. (2010) and De Koster and Balk (2008).

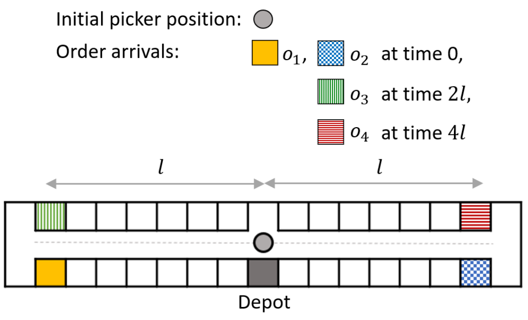
- We observed a common warehousing practice designed to avoid congestion and competition among pickers accessing items stored in the same location, by assigning those items to workers during an upstream planning stage (see Schiffer et al., 2022). Also comparative analyses in the literature reveal no differences in the performance rankings of alternative heuristic policies between single-picker and multiple-picker environments (Gil-Borrás et al., 2024). Based on this, we focus on the operations of a *single picker.*

- Besides traditional class-based storage, many e-commerce warehouses use a scattered pattern known as mixed-shelves storage. Unlike class-based storage, where each product item is assigned a picking position based on its picking frequency, mixed-shelves storage places individual items at multiple locations throughout the

Figure 3.1: Picking operation in a picker-to-parts warehouse

Source: Governo do Estado de São Paulo. License: CC BY 2.0

Figure 3.2: A picking instance which profits from strategic waiting



Note. Picking plan without waiting: picker completes o_1 at time $2l$; o_2 at time $4l$; o_3 at time $6l$; o_4 at time $8l$. Average order turnover time: $3.5l$.
Picking plan with waiting: picker jointly completes o_1 and o_3 at time $3l$; then she jointly completes o_2 and o_4 at time $5l$. Average order turnover time: $2.5l$.

warehouse. This approach is advantageous because e-commerce orders often contain only one or two items (Boysen et al., 2019), the product assortment is extensive, and demand distribution has a long tail (Brynjolfsson et al., 2003). Scattered storage reduces unproductive picker walking by increasing the likelihood that ordered items can be picked together from nearby locations. The state-of-the-art advice in the literature emphasizes the importance of allocating storage space to product items during upstream planning, over the selection of picking locations when planning a picking tour. In the latter case, simple myopic rules, such as the nearest-neighbor heuristic, are considered sufficient (see Weidinger & Boysen, 2018). Therefore, we assume the *picking positions of ordered items are predetermined*, which allows our analysis to apply to both class-based and mixed-shelves storage, subject to reservations discussed above.

As *our practical contribution*, we analyze CIOSs under two common objectives: makespan (minimizing costs representing picker's working time and wages) and speed of delivery (minimizing average turnover). Surprisingly, our detailed analysis of CIOSs downplays the importance of strategic waiting and explains why this may be the case. Instead, it offers actionable insights for significantly improving e-commerce warehousing operations across both objectives, as demonstrated through computational experiments. While these recommendations may seem intuitive in hindsight, they have been largely overlooked in the literature.

As *our theoretical contribution*, we design a customized exact *dynamic programming algorithm (DP)* for the *Order Batching, Sequencing, and picker Routing Problem with Release Times (OBSRP-R)*, classified after Pardo et al. (2024). To our knowledge, this is the first exact algorithm for OBSRP-R in the literature. Serving as a perfect-anticipation policy that computes CIOSs for the *online* counterpart OBSRP with dynamically arriving orders, our DP sets a benchmark for emerging AI-based anticipatory approaches. Furthermore, computational experiments show that standard mixed-integer programming solvers are unable to handle even small instances efficiently, highlighting the necessity of this tailored DP algorithm. We also identify analytical properties of optimal policies for online OBSRP.

We begin with a literature review (Section 3.2). Section 3.3 states the problem, Sections 3.4 and 3.5 describe the proposed DP and state some analytical properties of online policies, respectively. Section 3.6 presents detailed experimental studies and the results. This is the main section of the paper. Section 3.7 concludes with discussions and an outlook.

3.2 Literature review

Over the past decades, warehousing logistics has been a prominent topic in optimization literature. The majority of scientific efforts have concentrated on the picking operation; for a comprehensive overview, we refer to the recent surveys by Boysen et al. (2019), Pardo et al. (2024), and Vanheusden et al. (2023).

This section outlines three key literature streams relevant to our study: insights from optimal policies for planning warehousing operations with dynamically arriving orders; the role of advanced anticipation in warehousing operations; exact solution approaches for OBSRP-R and related problems. The latter refers to solution approaches for calculating the perfect-information optimum for the online OBSRP, serving as a benchmark for policies incorporating anticipation.

3.2.1 Insights from optimal policies for warehousing with dynamic orders

Although roughly one in four papers addresses warehousing operations with *dynamically* arriving orders, most focus on developing planning heuristics, including meta- and matheuristics, rule-based, and AI-based methods (Pardo et al., 2024).

The majority of studies evaluate the developed heuristics' performance by comparing them with other, often simpler, benchmark strategies. Such computational comparisons of heuristics (Giannikas et al., 2017; Gil-Borrás et al., 2024) offer insights into the relative effectiveness of the investigated algorithmic elements. For instance, optimal picker routing and optimal batching of available orders have shown significant improvements over simpler heuristics (Gil-Borrás et al., 2024; Henn, 2012). However, optimality gaps and the remaining potential for improvement, e.g., by advanced anticipation, remain unclear.

A few studies derived *analytical* performance guarantees for online policies. Henn (2012) and Alipour et al. (2020) provided performance guarantees for a *reoptimization* policy with threshold-based *waiting intervals*, for the batching and sequencing subproblem of online OBSRP. They proved that the investigated policy is 2-*competitive* for the studied problem variant, meaning that the resulting makespan is at most twice that of the CIOS. In Chapter 2 we proved that *immediate, myopic reoptimization* (Reopt) is an *optimal* policy for the more general online OBSRP under generic stochastic conditions, based on a probabilistic, asymptotic competitive analysis. This means that for sufficiently large instances, and under the objective of minimizing the operation's total completion time, the result of *Reopt* coincides with that of a perfect anticipation exact algorithm with a probability of one, and cannot be improved. However, the analysis of Chapter 2 applies to large instances where the number of orders approaches infinity.

The significance of strategic waiting was extensively studied for *first-come-first-served (FIFO)* order batching policies using queuing theory analysis (Gong & Gong, 2009; Le-Duc & De Koster, 2007; Van Nieuwenhuyse & De Koster, 2009). In FIFO-batching, orders are grouped as they arrive, and are released as a batch in fixed time intervals or when a fixed number of orders are queued. In the context of FIFO-batching, strategic waiting increases the likelihood of forming larger batches and, thus, realizes savings from picking multiple orders together when the queue is short.

To gain further insights into strategic waiting, Bukchin et al. (2012) developed an optimal waiting policy to minimize tardiness and overtime costs for pickers, by formulating the underlying Markov decision process, for orders arriving according to a Poisson point process. Because of the innate complexity of the online OBSRP, known as the curse of the dimensionality (cf. Powell, 2011), they studied the optimal strategic waiting in a setting, where the batching and routing decisions are significantly simplified and the picking time of a batch solely depends on the *number* of constituent orders. Interestingly, the selected objective function (overtime and tardiness) by Bukchin et al. (2012) significantly reduces the opportunity costs associated with waiting. However, even in this setting, the optimal policy recommended little to no waiting, except when lead times (the time available for picking an order, from its arrival time to its due date) were long compared to the picking time of an order. The authors suggest this may be due to the low order arrival rates used in the computational experiments.

Since then, a variety of approaches for implementing strategic waiting have emerged, ranging from fixed time windows to complex formulas (see Gil-Borrás et al., 2024, for an overview). For example, drawing on the batching machine problem, Henn (2012) evaluated several methods to schedule strategic waiting for the makespan objective. However, his experiments showed that no-wait policies performed better than those involving waiting. Overall, for more advanced and better-performing policies than FIFO-batching, the advice on strategic waiting remains mixed (Gil-Borrás et al., 2024; Henn, 2012).

Unlike the studies mentioned above, this paper develops an optimal policy for the perfect-anticipation counterpart of the online OBSRP.

3.2.2 The role of advanced anticipation in warehousing and related tasks

Although precise prediction of order sequences and arrival times is still elusive, anticipation can already be effectively utilized in warehouse operational planning.

The key to integrating anticipation into planning is to account for both immediate effects of decisions and their delayed consequences (cost-to-go). For instance, when deciding whether to dispatch a worker with a single order, the cost-to-go considers the probabilities and timings of future order arrivals, potential savings from larger batches, and quantifies the long-term impact of this decision. Estimating cost-to-go typically involves sampling and statistical aggregation techniques, including machine learning and deep learning methods (Bertsekas, 2020; Satton & Barto, 2018). Additionally, rather than optimizing based on given forecasts, these forecasts can be directly integrated into optimization. This approach incorporates contextual information – predictive factors driving the forecast, such as information on marketing campaigns – into the problem input. By simultaneously considering forecasting and optimization, the framework penalizes forecast errors that affect decisions while tolerating irrelevant ones (see Sadana et al., 2024, for a discussion).

Recent advancements in predicting customer ordering behavior have leveraged new data sources, such as weather data (Steinker et al., 2017), click data and page views (Ferreira et al., 2016; Huang & van Mieghem, 2014), consumer reviews (Y. Chen & Xie, 2008), and social media activities (Cui et al., 2018). Progress is moving towards hourly, item-level forecasts of arriving orders (Dai et al., 2022; Hamdan et al., 2023).

Few studies have integrated ML-based anticipation methods into warehouse picking operations. D'Haen et al. (2023) explored online OBSRP with fixed delivery truck schedules, aiming to minimize average order tardiness. They used historical data to introduce dummy orders with due dates aligned with truck departures, serving as placeholders for potential future arrivals. Cals et al. (2021) used deep reinforcement learning (DRL) to optimize e-commerce order batching in a combined picker-to-parts and parts-to-picker system. Shelke et al. (2021) proposed an end-to-end DRL method for scenarios where anticipated orders can be picked during one of the previous night shifts.

3.2.3 Exact solution approaches for order picking with complete information

To the best of our knowledge, no solution approaches have been developed specifically for OBSRP-R so far. While some papers provide mixed integer programs (MIP) for OBSRP-R or its subproblems (e.g. Alipour et al., 2020; Cals et al., 2021; Gil-Borrás et al., 2020, 2021, 2024; Henn, 2012; Pérez-Rodríguez et al., 2015; J. Zhang et al., 2017), these MIPs were mainly designed to communicate the problem clearly and were unable to solve even small instances. Given the poor performance of these MIPs and since these formulations focus on special cases of OBSRP-R or incorporate additional specific characteristics, we developed new MIPs in Supplement 3.8.1.

Even without release times, no paper has proposed an exact solution approach that simultaneously addresses *all* related picking decisions – order batching, sequencing, and picker routing – due to the problem's complexity. Adding release times increases this complexity. For example, the picker routing subproblem, which is polynomially solvable (Ratliff & Rosenthal, 1983; Schiffer et al., 2022), becomes NP-hard with release times (Bock et al., 2024).

Gademann and van de Velde (2005), Muter and Öncan (2015), and Wahlen and Gschwind (2023) developed exact Branch-and-Price (B&P) algorithms for the Order Batching Problem (OBP) with the objective of minimizing the single picker's travel distance. In the three studies, the linear relaxation of the problem was solved by a column generation approach. Thereby, the Master Problem was formulated as a set partitioning problem and the columns (i.e. the decision variables) constitute all possible batches, defined as a subset of orders. These formulations are characterized by difficult subproblems, as the *costs* of the decision variables, i.e. the distance functions of the batches, are inseparable, in the combined orders. Gademann and van de Velde (2005) first tackled the pricing problem with the help of a combinatorial branch-and-bound (B&B) tree with a maximum number of levels that coincides with the batching capacity. Muter and Öncan (2015) then generalized and improved their approach with a column pool strategy and acceleration strategies for the column generation subproblem. They implemented their approach for the traversal-, return-, and midpoint routing policies. The current state-of-the-art approach for OBP by Wahlen and Gschwind (2023) showed effective for *any monotone* routing policy, including *optimal routing*. They modeled the column generation pricing problem as a shortest path problem with resource constraints, that was solved with a labeling algorithm strengthened by strong completion bounds. Furthermore, they introduced two families of valid inequalities: capacity cuts and subset-row cuts, and two heuristics for OBP, strengthening the lower- and upper bounds, respectively, in the branching process to find the integer optimal solution. Although Wahlen and Gschwind (2023) were able to solve instances with several hundred orders, they identified batching capacity as a key factor driving the complexity of the OBP. This also applies to OBSRP-R and the DP approach developed in this paper. Importantly, note that the B&P approaches for OBP based on a set partitioning formulations *cannot* be extended to OBSRP-R. The additional *sequencing component* stemming from the order release times causes the batch travel times, and thus the cost functions associated with the columns, to depend not only on the combined orders but also on the *starting time* of the batch, invalidating the proposed strategies for the pricing subproblem, and in general the usage of a set partitioning formulation.

A different exact approach was suggested by Valle et al. (2016, 2017) to solve the joint order batching and picker routing problem (OBPR) minimizing the total distance traveled. They modeled the problem with a non-compact arc-based formulation using the *sparse-graph* representation of the warehouse. The model was embedded in a branch-and-cut algorithm which separates the exponential amount of connectivity constraints resulting from this formulation. Valle et al. (2017) introduced several problem-specific cuts to speed up the approach. In the context of online grocery shopping with a large number of items per order, the approach found optimal solutions for instances with up to 20 orders. Notably, the method cannot be applied in the case of OBSRP-R. The proposed MIP formulation specifically relied on the fact that each directed arc of the sparse graph is traversed at most once per batch in an optimal OBPR solution. This does not hold anymore in the case of release times and non-zero picking times; the same arc may be traversed at the beginning of the batch route to collect an item with an early release, and then again later to pick another item with a late release.

For the OBPR in a general multi-block warehouse with multiple depots, Schiffer et al. (2022) exactly optimized the pickers' travel distances, thereby allowing cartless-subtours. The batching subproblem in their study is different from OBSRP-R, in the sense that the orders (in their case *pick-lists*) must be batched in the sequence of their appearance, but may be dropped at one of the depots independently from the rest of the batch once completed, freeing cart capacity for the next order in the sequence. The developed layered graph algorithm for batch routing represents a new state-of-the-art, but is not applicable in the case of release times, as it can not account for the picker's arrival times at specific picking locations. Aerts et al. (2021) showed that the OBPR, without a sequencing aspect, can be modeled and solved as a clustered vehicle routing problem.

Scheduling in OBSRP-R is involved at two different levels. Firstly, the *completion times of individual orders* are accounted for in the objective function, when the orders' turnover times are minimized. Secondly, the schedule enforces that no item is picked before the release time of the respective order. While this is, to the best of our knowledge, the first study to propose an *exact* approach for scheduling at both levels, some metaheuristc approaches have been proposed for the Order Batching, Sequencing, and picker Routing Problem OBSRP considering the first-level scheduling aspect within the objective or the constraints (T.-L. Chen et al., 2015; Scholz et al., 2017; Van Gils et al., 2019).

3.3 Problem statement

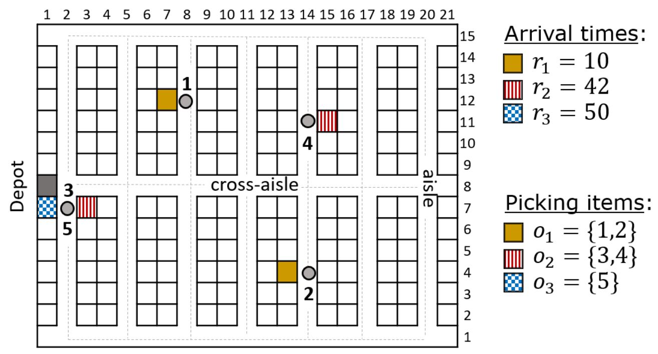
We define OBSRP-R in Section 3.3.1. Afterward, we formally propose two different objective functions in Section 3.3.2 and introduce the *online* variant of the problem in Section 3.3.3.

Throughout the rest of the paper, we abbreviate $\{1,\dots,n\}, \forall n \in \mathbb{N}$ simply as $[n]$. Table 3.1 summarizes the relevant notation introduced in this section.

3.3.1 The order batching, sequencing and picker routing problem with release times (OBSRP-R)

OBSRP-R describes the picking operation of a single picker equipped with a pushcart (cf. Pardo et al., 2024). For simplicity, we call the working zone of this picker a *warehouse*. The warehouse is rectangular and consists of $a \geq 1$ vertical *aisles* of length W, $b \geq 2$ horizontal *cross-aisles* of length L, and a depot l_d which is positioned at some arbitrary location in the warehouse (see Figure 3.3). We dub the access point to the storage location, from which an ordered item can be retrieved, as the *picking location* of this item. (see circles in Figure 3.3). Note that the picker can access the product items only from the aisles. Specifically, no items can be picked from the cross-aisles.

Figure 3.3: An instance $I^{\text{ex\_0}}$ of OBSRP-R



Note. A warehouse with $b = 3$ cross-aisles and $a = 7$ aisles. The width of each shelf, aisle, and cross-aisle is one, and the picker moves at uniform speed $v = 1$. The batching capacity is $c = 2$ and the time to pick an item is $t^p = 5$. The instance has $n^o = 3$ orders, the items of which are stored in the colored shelves. The grey circles in the adjacent aisles mark the respective picking locations of these items.

The picker begins her operation at the depot and navigates through the warehouse at a constant *speed*, v. She has a cart with c bins, as depicted in Figure 3.1, where each bin is dedicated to a different order, so that items from the same order are grouped in one bin. Consequently, up to c orders can be collected *simultaneously* (in one *batch*) to speed up the process. We denote c as the *batching capacity* in the following. Note that items from the same order cannot be split across different batches. Completion of a batch occurs when all of its items have been picked the picker returns the cart to the depot for unloading. Subsequently, the picker retrieves a new cart to commence the next batch.

Following the classical definition of OBSRP(-R) (cf. Pardo et al., 2024), and as motivated in Section 3.1, we assume that each ordered item is associated with a *unique* picking location as it enters the picking system. This framework establishes a specific distance metric $d()$ for all the product items stored in the warehouse: for any two items s_1 and s_2; $d(s_1, s_2)$ represents the *shortest walking distance* between their respective picking locations, navigated rectilinearly through the warehouse's aisles and cross-aisles. We extend the distance metric with the shortest walking distance between the depot and the location of any item s, noted as $d(l_d, s)$. It is important to highlight that the *triangular inequalities* are inherently satisfied: any three i, l, s representing a picking item or the depot, $d(i,s) \leq d(i,l) + d(l,s)$.

An *instance* I of the OBSRP-R has the following input (see Table 3.1):

- warehouse parameters, such as the number of aisles a, the number of cross-aisles b, length L, width W, distance metric $d()$;
- parameters of the picker and the cart, such as the picker's speed v, the batching capacity c, and constant pick time t^p to retrieve an item from its picking location;
- set of n^o orders. Each order $o_j, j \in [n^o]$:
  - represents a set of picking items $o_j := \{s_j^1, ..., s_j^{l_j}\}, l_j \in \mathbb{N}$;
  - is associated with a release time $r_j \in \mathbb{R}^+$.

W.l.o.g., we assume $r_1 \leq r_2 \leq ... \leq r_{n^o}$. We also abuse the notation and denote the release time of item s in order o_j simply as $r(s) := r_j$. Each order can have a different number of items. For convenience, we denote the total number of ordered items as

$n^i := \sum_{j=1}^{n^o} |o_j|$ and the set of all such items in the instance as $S := \bigcup_{j\in[n^o]}\{o_j\}$. The retrieval of each item requires fixed *pick time* t^p.

Let denote the *completion time* of an ordered item $s \in S$ in some solution σ as $C(s,\sigma)$, which is the time when the item is placed in the cart. We abuse the notation and drop the reference to σ, if it is clear from the context.

A *feasible solution* σ of OBSRP-R for a given instance I consists of:

- the picking sequence of the ordered items π, which is constructed as $\pi := (\pi^{B_1}, \pi^{B_2}, \dots, \pi^{B_f})$, $|\pi| = n^i$, based on the following components:
  - an ordered sequence of batches $\pi^{\text{batches}} = (B_1, B_2, \dots, B_f)$, such that the batches form a mutually disjoint partition of the orders: $\{o_1, ..., o_{n^o}\} = B_1 \cup B_2 \cup \ldots \cup B_f, B_l \cap B_k = \emptyset\ \forall k,l \in \{1,...,f\}, k \neq l$; each batch contains at most c orders. The number of batches $f \in \mathbb{N}$ is a decision variable.
  - for each batch B_l, a permutation of the items in the included orders π^{B_l}.
- the picking *schedule* consisting of a completion time $C(s)$ for each item $s \in S$, such that the routing requirements are respected, and no item s is picked before its release time $r(s)$.

Typically, the objectives in the order-picking process aim for each item to be collected at the *earliest possible time* (which accounts for the release times of the orders). Specifically, within the scope of the objectives discussed in this paper, an optimal picking schedule can be uniquely determined for any instance I from a given picking sequence π and its associated batches π^{batches}. This is achieved by applying the formulas from equations (3.1) to (3.2). For convenience, let denote the i^{th} item in π as $\pi[i]$.

$$C(\pi[1]) = \max\{\frac{1}{v} \cdot d(l_d, \pi[1]), r(\pi[1])\} + t^p \tag{3.1}$$

$$C(\pi[i+1]) = \begin{cases} \max\{C(\pi[i]) + \frac{1}{v} \cdot d(\pi[i], \pi[i+1]), r(\pi[i+1])\} + t^p, \\ \qquad \text{if } \pi[i], \pi[i+1] \text{ belong to the same batch} \\ \max\{C(\pi[i]) + \frac{1}{v} \cdot d(\pi[i], l_d) + \frac{1}{v} \cdot d(l_d, \pi[i+1]), r(\pi[i+1])\} + t^p, \\ \qquad \text{if } \pi[i], \pi[i+1] \text{ belong to distinct batches} \end{cases}$$
$$\forall i \in [n^i - 1] \tag{3.2}$$

The second case of equation (3.2) occurs when the picker, having completed a batch with item $\pi[i]$, returns to the depot for unloading. Subsequently, the picker starts a new batch using a fresh cart, beginning with the collection of item $\pi[i+1]$.

3.3.2 Objectives

We study two objectives: minimizing the makespan (referred to as *makespan*) and minimizing the average order turnover (referred to as *turnover*).

The *makespan* of a solution σ for a given instance, also called *total completion time*, is formally defined as follows:

$$z^{\text{makesp}}(\sigma) = \max_{s\in S}\{C(\sigma, s) + \frac{1}{v} \cdot d(s, l_d)\} \tag{3.3}$$

Table 3.1: Notation of the OBSRP-R and the DP approach

Parameters of an OBSRP-R instance I	
a	Number of aisles in the warehouse
b	Number of cross-aisles in the warehouse
L	Length of a cross-aisle
W	Length of an aisle
l_d	Depot
$d()$	Distance metric between items and depot
v	Picker speed
t^p	Picking time to retrieve one item from its storage location
c	Batching capacity: the maximum number of orders in a batch
n^o	Number of orders
$o_j = \{s_j^1, ...s_j^l\}$	Set of items requested by the j^{th} order, $j \in [n^o], l \in \mathbb{N}$
r_j	Release time of order $o_j, j \in [n^o]$, $r_1 \leq ... \leq r_{n^o}$
$r(s)$	Release time of an item s, i.e. release time of respective order, $r(s) := r_j$ if $s \in o_j$
n^i	Number of items, $n^i := \sum\limits_{j=1}^{n^o} \lvert o_j \rvert$
S	Set of all ordered items $S := \cup_{j \in [n^o]} o_j$
σ	Feasible solution
$\pi := (\pi^{B_1}, \pi^{B_2}, \dots, \pi^{B_f})$	Picking sequence of items; with π^{B_i} the picking sequence within a batch B_i
$C(\sigma,s)/C(\sigma,o_j)$	Completion time of item s/order o_j in solution σ (σ dropped if clear from context)
$z^{\text{makesp}}(\sigma)$	Makespan of solution σ
$z^{\text{turnover}}(\sigma)$	Average order turnover time of solution σ
Notation used in the DP-approach	
O	Sequence of orders sorted with respect to their release times
$\Theta_k = (s, m^o, S^{\text{batch}}, O^{\text{pend}})$	State at stage $k \in [n^i + 1] \cup \{0\}$;
	for $k \leq n^i$, k items have been picked at this state; Θ_{n^i+1} is the terminal state;
	$s \in S \cup l_d$ denotes the last picked item for $k \in [n^i]$;
	$m^o \in [c] \cup \{0\}$ counts the number of orders in the current batch;
	$S^{\text{batch}} \subseteq S$ set of items in orders of current batch that have not been picked;
	$O^{\text{pend}} \vert O$ is the sequence of pending orders
$X(\Theta_k)$	Set of feasible transitions from state Θ_k
$f(\Theta_k, x_k)$	Transition function for state Θ_k at stage k and transition $x_k \in X(\Theta_k)$
$g^{val}(\Theta_k, x_k)$	Immediate value $x_k \in X(\Theta_k)$ at state Θ_k, for $val \in$ {time, comp, comp+}
$\Omega^{val}(\Theta_k)$	Value of state S_k for $val \in$ {time, comp, comp+}
$\mathcal{A}^{\text{batch-comp}}$	Set of batch-completion states in state graph
$\chi(o_j)$	Shortest time to pick order $o_j, j \in [n^o]$ in a single-order batch

By definition of (3.3), the picking operation is completed once the picker returns the cart to the depot after all ordered items have been collected.

Similarly, an order is *completed*, when all items of its associated batch have been picked and the cart containing that batch has been brought to the depot. Formally, let denote $B(\sigma, o_j) \subseteq S$ the set of items in batch B_l if $o_j \in B_l$ in solution σ. Then, the completion time of order o_j in solution σ is defined as:

$$C(\sigma, o_j) := \max_{s \in B(\sigma, o_j)} \{C(s) + \frac{1}{v} \cdot d(s, l_d)\} \tag{3.4}$$

The reference to the solution σ can be dropped in the following if clear from the context. The *turnover* time of an order o_j describes the time from its arrival r_j to its completion. Thus the *average turnover time* of a solution σ for a given instance formally writes as:

$$z^{\text{turnover}}(\sigma) = \frac{1}{n^o} \sum_{j \in [n^o]} (C(\sigma, o_j) - r_j) \tag{3.5}$$

$$= \frac{1}{n^o} \cdot \sum_{j \in [n^o]} C(\sigma, o_j) - \frac{1}{n^o} \sum_{j \in n^o} r_j \tag{3.6}$$

Table 3.2: Optimal solutions for OBSRP-R instance I^{ex_0} from Figure 3.3

Solution σ_i^*, $i \in \{1,2\}$	$\pi(\sigma_i^*)$	$\pi^{\text{batches}}(\sigma_i^*)$	$z^{\text{makesp}}(\sigma_i^*)$	$z^{\text{turnover}}(\sigma_i^*)$
σ_1^*: Optimal solution for makespan objective	$(1,2,4,3,5)$	$(\{o_1,o_2\},\{o_3\})$	76	37.3
σ_2^*: Optimal solution for turnover objective	$(1,2,5,3,4)$	$(\{o_1\},\{o_3\},\{o_2\})$	99	34.7

Observe that the right-hand summand of (3.6) represents the average order arrival time and is a *constant*. Thus minimizing the average turnover time is equivalent to minimizing the *sum of order completion times*.

For instance I^{ex_0} in the illustrative example of Figure 3.3, *optimal solutions* for both objectives, makespan (see σ_1^*), and turnover (see σ_2^*), are provided in Table 3.2. In the example, the picker moves at unit speed $v = 1$. In the case of the makespan objective, by formulas (3.1)-(3.2), the items 1 and 2 from order o_1 are picked first, and have a completion time of $C(1) = d(l_d,1) + t^p = 10+5 = 15$ and $C(2) = c(1) + d(1,2) + t^p = 15+14+5 = 34$, respectively, since the picker arrives first at item 1, then at item 2, after the release time of the respective order, r_1. Then, the picker proceeds toward item 4 of order o_2, reached at time 41, and must wait for one time unit for its arrival, thus the completion time is $C(4) = r(4) + t^p = 42+5 = 47$. Finally, item 3 is picked at time $C(3) = C(4) + d(4,3) + t^p = 47+16+5 = 68$ and the first batch $B_1 = \{o_1,o_2\}$, as well as their included orders, are completed at time $C(o_1) = C(o_2) = C(3) + d(3,l_d) = 68+1 = 69$. The second batch $B_2 = \{o_3\}$ contains one item, picked at time $C(5) = C(3) + d(3,l_d) + d(l_d,5) + t^p = 68+1+1+5 = 75$, thus the third order is completed at time $C(o_3) = C(5) + d(5,l_d) = 75+1 = 76$. By definition (3.3), the associated optimal makespan is $z^{\text{makesp}}(\sigma_1^*) = 76$. The associated average turnover to this solution, which is not optimal, is given by formula (3.5): $z^{\text{turnover}}(\sigma_1^*) = \frac{1}{3}\cdot((C(o_1)-r_1) + (C(o_2)-r_2) + (C(o_3)-r_3)) = \frac{1}{3}\cdot(59+27+26) = 37.3$. In the second case of the turnover objective, the proposed optimal solution σ_2^* of Table 3.2 suggests to pick all orders separately. By the same formulas as above, the first order defines the first batch $B_1 = \{o_1\}$ and is completed at time $C(o_1) = 50$. The second batch $B_2 = \{o_3\}$ and its associated order are completed at time $C(o_3) = 57$. The last batch $B_3 = \{o_2\}$ and its associated order are completed at time $C(o_2) = 99$. This leads to the suboptimal makespan of $z^{\text{makesp}}(\sigma_2^*) = 99$ and the optimal average turnover time of $z^{\text{turnover}}(\sigma_2^*) = \frac{1}{3}((50-10) + (57-50) + (99-42)) = 47+57 = 34.7$

Mixed integer programs for OBSRP-R with both objectives (3.3) and (3.5) are provided in Supplement 3.8.1.

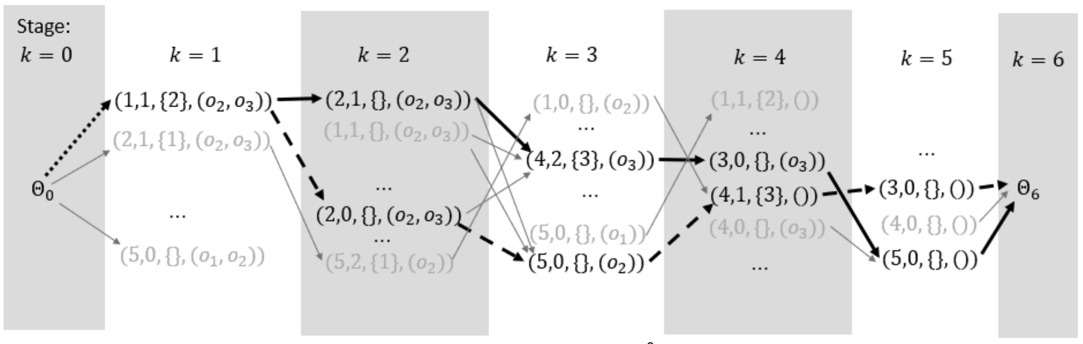
3.3.3 The online problem with dynamically arriving orders: OOBSRP

In reality, the information about arriving orders is not perfect. For a future order o_j, the set of picking items $s_j^1,...,s_j^l$ and the number of items l are only revealed upon the order's arrival. Similarly, the total number of orders, $[n^o]$, is unknown. The release times r_j are, in fact, the dynamically revealed *order arrival times*. We refer to the online variant of the problem as OOBSRP.

3.4 Dynamic program

In this section, we describe the developed *dynamic programming (DP) formulations* for OBSRP-R. Section 3.4.1 describes the associated state graph. Section 3.4.2 introduces the labeling algorithms to find optimal solutions for both objectives, as well as upper bounds for the turnover objective. Afterward, Section 3.4.3 introduces several dominance

Figure 3.4: Illustration of the state graph



Note. Extract of the state graph corresponding to the instance $I^{\text{ex\_0}}$ from Figure 3.3. The arcs in bold mark the path through the state graph that corresponds to the optimal solution for the makespan objective σ_1^*; the dashed arcs mark the path of the optimal solution σ_2^* for the turnover objective; and the dotted arc belongs to both optimal solutions (see Table 3.2). The states involved on at least one of these paths have a black font; some other states are included in light grey, as well as all other feasible transitions between two depicted states, which do not belong to the optimal paths.

relations, which are used to speed up the DP approaches. Table 3.1 summarizes the additional notation introduced in this section.

3.4.1 State graph

The following construction is identical for both studied objective functions.

Recall that we sort the orders non-decreasingly with respect to their release times. Let denote this sorted sequence of orders as O and, for convenience, abbreviate the expression *"O' is an ordered subsequence of O"* as $O'|O$.

OBSRP-R is reinterpreted as a sequential optimization problem with $k \in \{0,1,...,n^i, n^i+1\}$ stages. At stage $k \le n^i$, the picker has picked k items and, given the current *state* (such as her location, available bins in the cart etc.), faces the *subproblem* to pick the remaining items of the remaining orders. Her immediate *decision* at stage k is about the next item to collect - and, possibly about the closure or extension of the current batch. Stage n^i+1 is reserved for the terminal state as explained below. We depict the resulting sequential problem as a *state graph*. The nodes of this graph are called *states*. The directed edges depict *transitions* between the states of the subsequent stages and are associated with *decisions*.

States. We define state Θ_k at stage $k \in [n^i]$ as a tuple of the following four variables:

$$\Theta_k \in \{(s, m^o, S^{\text{batch}}, O^{\text{pend}}) | s \in S, m^o \in \{0,1,\dots,c\}, S^{\text{batch}} \subseteq S, O^{\text{pend}}|O\} \qquad (3.7)$$

Variable $s \in S$ denotes the last picked item. Integer $m^o \in \{0,1,\dots,c\}$ counts the number of orders currently assigned to the open batch. The *set* $S^{\text{batch}} \subseteq S$ accommodates the items of the orders from the current batch, which have not been picked yet. Sequence $O^{\text{pend}}|O$ is the *sequence* of pending orders, no items of which have been picked so far.

In the *initial* state $\Theta_0 = (l_d, 0, \{\}, O)$ at stage $k=0$, the picker is at the depot with an empty cart. The last stage $k = n^i+1$ consists of one state, dubbed *terminal state*. It describes the picker's return to the depot with all the orders processed: $\Theta_{n^i+1} = (l_d, 0, \{\}, ())$.

States of type $\Theta_k = (s, 0, \{\}, O^{\text{pend}})$ with $S^{\text{batch}} = \{\}$ and $m^o = 0$ are called *batch-completion states* in the following. In batch completion states, the picker has completed the batch by picking item s and moved to the depot for unloading, thus in these states, the picker's current position is l_d. In all other states, $\Theta_k = (s, m^o, S^{\text{batch}}, O^{\text{pend}}), m^o > 0$,

Table 3.3: OBSRP-R with a *pushcart*: Feasible transitions at stages $k = \{0, 1, \ldots, n^i - 1\}$

	Current state Θ_k	Transition $x_k \in X(\Theta_k)$ $\forall s_{k+1}$ s.t.	New state $\Theta_{k+1} = f(\Theta_k, x_k)$
1	$(s_k, 0, \{\}, O^{\text{pend}})$	$s_{k+1} \in o_j, o_j \in O^{\text{pend}}$	$(s_{k+1}, 1, o_j \setminus \{s_{k+1}\}, O^{\text{pend}} \setminus o_j)$
2	-//-	$s_{k+1} \in o_j, o_j \in O^{\text{pend}}, \lvert o_j \rvert = 1$	$(s_{k+1}, 0, \{\}, O_k^{\text{pend}} \setminus o_j)$
3	$(s_k, m^o, \{\}, O^{\text{pend}}), m^o < c-1$	$s_{k+1} \in o_j, o_j \in O^{\text{pend}}$	$(s_{k+1}, m^o + 1, o_j \setminus s_{k+1}, O^{\text{pend}} \setminus o_j)$
4	-//-	$s_{k+1} \in o_j, o_j \in O^{\text{pend}}, \lvert o_j \rvert = 1$	$(s_{k+1}, 0, \{\}, O^{\text{pend}} \setminus o_j)$
5	$(s_k, m^o, \{\}, O^{\text{pend}}), m^o = c-1$	$s_{k+1} \in o_j, o_j \in O^{\text{pend}}, \lvert o_j \rvert > 1$	$(s_{k+1}, m^o + 1, o_j \setminus s_{k+1}, O^{\text{pend}} \setminus o_j)$
6	-//-	$s_{k+1} \in o_j, o_j \in O^{\text{pend}}, \lvert o_j \rvert = 1$	$(s_{k+1}, 0, \{\}, O^{\text{pend}} \setminus o_j)$
7	$(s_k, m^o, S^{\text{batch}}, O^{\text{pend}}), m^o \leq c-1, \lvert S^{\text{batch}} \rvert > 1$	$s_{k+1} \in o_j, o_j \in O^{\text{pend}}$	$(s_{k+1}, m^o{+}1, S^{\text{batch}} \cup o_j \setminus s_{k+1}, O^{\text{pend}} \setminus o_j)$
8	-//-	$s_{k+1} \in S^{\text{batch}}$	$(s_{k+1}, m^o, S^{\text{batch}} \setminus \{s_{k+1}\}, O^{\text{pend}})$
9	$(s_k, m^o, S^{\text{batch}}, O^{\text{pend}}), m^o = c, \lvert S^{\text{batch}} \rvert > 1$	$s_{k+1} \in S^{\text{batch}}$	$(s_{k+1}, m^o, S^{\text{batch}} \setminus \{s_{k+1}\}, O^{\text{pend}})$
10	$(s_k, m^o, S^{\text{batch}}, O^{\text{pend}}), m^o \leq c-1, \lvert S^{\text{batch}} \rvert = 1$	$s_{k+1} \in o_j, o_j \in O^{\text{pend}}$	$(s_{k+1}, m^o{+}1, S^{\text{batch}} \cup o_j \setminus s_{k+1}, O^{\text{pend}} \setminus o_j)$
11	-//-	$s_{k+1} \in S^{\text{batch}}$	$(s_{k+1}, m^o, \{\}, O^{\text{pend}})$
12	-//-	$s_{k+1} \in S^{\text{batch}}$	$(s_{k+1}, 0, \{\}, O^{\text{pend}})$
13	$(s_k, m^o, S^{\text{batch}}, O^{\text{pend}}), m^o = c, \lvert S^{\text{batch}} \rvert = 1$	$s_{k+1} \in S^{\text{batch}}$	$(s_{k+1}, 0, \{\}, O^{\text{pend}})$

the picker's current position coincides with the picking location of the last picked item $s \in S$.

Transitions. We denote a set of feasible decisions, or *feasible transitions*, from state Θ_k at stage $k \leq n^i$ as $X(\Theta_k)$. The *transition function* $f(\Theta_k, x_k) = \Theta_{k+1}$ describes the next state after the execution of the decision $x_k \in X(\Theta_k)$ at state Θ_k.

At stage $k = n^i$, there is one only feasible transition in each state Θ_{n^i}, which is to move to the terminal state. At the remaining stages $k = \{0, 1, \ldots, n^i - 1\}$, transitions $x_k \in X(\Theta_k)$ refer to the selection of the next picking item s_{k+1}, and, potentially to the decision of extending- or completing the currently open batch.

Table 3.3 describes feasible transitions for OBSRP-R at stages $k \in \{0, ..., n^i - 1\}$. Lines 1 and 2 describe transitions from a batch-completion state which initiate a new batch. Note that, if the next batch starts by picking an item from a *single-item* order, the picker has two alternatives: Either to pick this item as part of a larger batch (line 1) or to limit the batch to this one order only (line 2). In lines 3 - 7 and 10, the currently open batch is extended by a pending order. This is only possible when the batching capacity is not exhausted ($m^o < c$). Thereby, in lines 3 to 6, there are still empty bins in the current batch, but all the items of the already assigned orders have been collected. If an item s_{k+1} from a *single-item* order is selected to be picked next, two alternatives must be distinguished: either the batch is completed by this order (i.e., the next state Θ_{k+1} is a batch-completion state, see lines 4 and 6), or further orders will be assigned to the current batch (lines 3 and 5). In lines 8-9 and 11-13, at least one item from a commenced order remains in S^{batch} and is selected to be picked next. Similarly to previous transitions, if only one item remains in S^{batch} and the batching capacity has not been depleted ($m^o \leq c - 1$) (lines 11 and 12), the picker can either close the batch after picking this remaining item and move to a batch-completion state (see line 12), or proceed by extending further the current batch (line 11).

The next Proposition claims a one-to-one correspondence between the paths in the constructed state graph and the feasible solutions of OBSRP-R.

Proposition 7. *For a particular OBSRP-R instance, each path from the initial state to the terminal state in the constructed state graph corresponds to exactly one feasible solution for this instance and vice-versa.*

Proof. Given that the first entry in each state at stage $k \in [n^i]$ represents one picked item, selected in the previous transition either from set S^{batch} or an order from sequence O^{pend} which contain only outstanding items, the sequence of these entries in a path of states forms a visiting sequence of picking locations π of Hamiltonian nature. A partitioning of this sequence into batches can be derived by closing one batch after each visit of a batch-completion state and opening a new batch with the picking item of the following state. By construction, the resulting batches form a disjoint partition of all n^o orders of the instance, and each batch contains at most c orders.

The other way around, each feasible solution translates to exactly one path of states connected by transitions from Table 3.3 by construction. □

Figure 3.4 illustrates an extract of the state graph corresponding to instance $I^{\text{ex\_0}}$ from Figure 3.3, highlighting the paths associated to the optimal solutions from Table 3.2.

3.4.2 Labeling algorithms

We first present the exact solution algorithms for the makespan- and turnover objective, in Sections 3.4.2.1 and 3.4.2.2, respectively. In Section 3.4.2.3, we propose a faster heuristic alternative for the turnover objective.

In all three presented algorithms, we propagate *labels* for each state of the state graph in a forward inductive manner (cf. e.g., Desrochers & Soumis, 1988; Irnich & Desaulniers, 2005, for a general introduction to labeling algorithms), and in all three algorithms, these labels include a *time value*. A time value $\Omega^{\text{time}}(\Theta_k)$, for state Θ_k at stage $k \in \{0, ..., n^i + 1\}$ represents the *time* to reach this state through a *particular path* in the state graph starting from the initial state Θ_0, i.e. by taking a sequence feasible picking, batching, and sequencing decisions.

The propagation of labels and the associated time values starts at the initial state with a time of $\Omega^{\text{time}}(\Theta_0) = 0$. Then, for every label associated to a state Θ_k at stage $k \in \{0, ..., n^i\}$, every feasible transition $x_k \in X(\Theta_k)$ towards a state $\Theta_{k+1} = f(\Theta_k, x_k)$ produces a new label, with a new time value for Θ_{k+1}. The time value of the new label is composed by the *immediate time* of the transition, noted $g^{\text{time}}(\Theta_k, x_k, \Omega^{\text{time}})$, and the time value of the considered label for the starting state of the transition:

$$\Omega^{\text{time}}(\Theta_{k+1}) = \Omega^{\text{time}}(\Theta_k) + g^{\text{time}}(\Theta_k, x_k, \Omega^{\text{time}}) \tag{3.8}$$

The immediate times represent the amount of time necessary for the transition between states. If the transition results in a state Θ_{k+1} with last-picked item s_{k+1}, it is composed of the maximum of the picker's walking time, and the waiting time for the release of s_{k+1} – plus the picking time t^p. To account correctly for the waiting time, $g(\Theta_k, x_k, \Omega^{\text{time}})$ depends on the current time at the departure state, i.e. the time value $\Omega^{\text{time}}(\Theta_k)$ of the departing label. The picker's walking time on the other hand only depends on the picker's current positions dictated by the states: if a state of the transition is a batch-completion state, her position is the depot, else, it is the picking location of item s_k or s_{k+1}, respectively. Let denote by $\mathcal{A}^{\text{batch-comp}}$ the subset of batch-completion states. Altogether, the immediate time of the transition from a label of state Θ_k, with time value $\Omega^{\text{time}}(\Theta_k)$, towards a state $\Theta_{k+1} = f(\Theta_k, x_k)$ is:

$$
\begin{aligned}
&g^{\text{time}}(\Theta_k, x_k, \Omega^{\text{time}}) \\
&= \begin{cases}
\max\{\frac{d(s_k,s_{k+1})}{v}; r(s_{k+1}) - & \Omega^{\text{time}}(\Theta_k)\} + t^p \\
 & \text{if } k < n^i, \Theta_k \notin \mathcal{A}^{\text{batch-comp}} \text{ and } \Theta_{k+1} \notin \mathcal{A}^{\text{batch-comp}} \\
\max\{\frac{d(l_d,s_{k+1})}{v}; r(s_{k+1}) - & \Omega^{\text{time}}(\Theta_k)\} + t^p \\
 & \text{if } k < n^i, \Theta_k \in \mathcal{A}^{\text{batch-comp}} \text{ and } \Theta_{k+1} \notin \mathcal{A}^{\text{batch-comp}} \\
\max\{\frac{d(s_k,s_{k+1})}{v}; r(s_{k+1}) - & \Omega^{\text{time}}(\Theta_k)\} + t^p + \frac{d(s_{k+1},l_d)}{v} \\
 & \text{if } k < n^i, \Theta_k \notin \mathcal{A}^{\text{batch-comp}} \text{ and } \Theta_{k+1} \in \mathcal{A}^{\text{batch-comp}} \\
\max\{\frac{d(l_d,s_{k+1})}{v}; r(s_{k+1}) - & \Omega^{\text{time}}(\Theta_k)\} + t^p + \frac{d(s_{k+1},l_d)}{v} \\
 & \text{if } k < n^i, \Theta_k \in \mathcal{A}^{\text{batch-comp}} \text{ and } \Theta_{k+1} \in \mathcal{A}^{\text{batch-comp}} \\
0 & \text{if } k = n^i
\end{cases}
\end{aligned}
\tag{3.9}
$$

3.4.2.1 Exact labeling for the makespan objective

For the makespan objective is cost minimization, the time value is the only value in each label and, by Bellman's Principle of Optimality (Bellman, 1952), it suffices to save only one label for each state, namely the one with the lowest time value, which represents the *time of the shortest-possible subpath* to reach this particular state through feasible transitions in the state graph. We note this value $\Omega^{*,\text{time}}(\Theta_k)$, for a state Θ_{k+1} at stage $k \in [n^o]$ and generate it inductively by the following Bellman equation:

$$
\Omega^{*,\text{time}}(\Theta_{k+1}) = \min_{(\Theta_k,x_k)\in f^{-1}(\Theta_{k+1})} \{\Omega^{*,\text{time}}(\Theta_k) + g^{\text{time}}(\Theta_k, x_k, \Omega^{\text{time}})\} \tag{3.10}
$$

The inverse image $f^{-1}(\Theta_{k+1})$ represents the set of all feasible transitions (see Table 3.3) to reach state Θ_{k+1}: $f^{-1}(\Theta_{k+1}) = \{(\Theta_k, x_k) \mid \Theta_k$ is a state at stage $k, x_k \in X(\Theta_k)$, and $f(\Theta_k, x_k) = \Theta_{k+1}\}$.

Finally, the optimal makespan for the given OBSRP-R instance equals the time value of the terminal state: $z^{*,\text{time}} = \Omega^{*,\text{time}}(\Theta_{n^i+1})$.

3.4.2.2 Exact labeling for the turnover objective

Recall that by definition (3.6), minimizing the average order turnover time is equivalent to minimizing the *sum of order completion times* of an instance. This objective will be represented by a *completion value* in every label generated throughout the exact labeling algorithm for the turnover objective. Specifically, the algorithm starts with one label of the initial state Θ_0 with a time value *and* a completion value of zero. Then, labels are propagated again in a forward-inductive motion from every label of every state Θ_k at stage $k \in \{0, ..., n^i\}$ through the feasible transitions $x_k \in X(\Theta_k)$. For every such transition and every originating label, the time value $\Omega^{\text{time}}(\Theta_{k+1})$ of the new label for state $\Theta_{k+1} = f(\Theta_k, x_k)$ is generated exactly by the formula (3.8). However, *additionally* for this new label, the associated completion value, noted $\Omega^{\text{compl}}(\Theta_{k+1})$ is generated, which represents the sum of the completion times of all completed orders *up to that state*. Very similarly to the time value, the completion value $\Omega^{\text{compl}}(\Theta_{k+1})$ is computed as:

$$
\Omega^{\text{comp}}(\Theta_{k+1}) = \Omega^{\text{comp}}(\Theta_k) + g^{\text{comp}}(\Theta_k, x_k, \Omega^{\text{time}}) \tag{3.11}
$$

Thereby, $\Omega^{\text{comp}}(\Theta_k)$ is the completion value of the originating label of the transition, and $g^{\text{comp}}(\Theta_k, x_k, \Omega^{\text{time}})$ is the *immediate completion* of the transition x_k from the state of the originating label Θ_k. Suppose that Θ_k is defined by the entries $(s_k, m^o, S^{\text{batch}}, O^{\text{pend}})$, see Section 3.4.1. Then, the immediate completion is non-zero for every transition towards a batch completion state Θ_{k+1} in $\mathcal{A}^{\text{batch-comp}}$ for $k < n^i$, and its value sums up the *time* the current batch is completed (i.e., $\Omega^{\text{time}}(\Theta_{k+1})$) for all orders in this batch. Specifically:

$$g^{\text{comp}}(\Theta_k, x_k, \Omega^{\text{time}}) \tag{3.12}$$

$$= \begin{cases} m^o \cdot \Omega^{\text{time}}(\Theta_{k+1}) & \text{if } k < n^i, \Theta_{k+1} \in \mathcal{A}^{\text{batch-comp}} \text{ and } S^{\text{batch}} \neq \{\} \\ (m^o + 1) \cdot \Omega^{\text{time}}(\Theta_{k+1}) & \text{if } k < n^i, \Theta_{k+1} \in \mathcal{A}^{\text{batch-comp}} \text{ and } S^{\text{batch}} = \{\} \\ 0 & \text{if } k = n^i \text{ or } \Theta_{k+1} \notin \mathcal{A}^{\text{batch-comp}} \end{cases} \tag{3.13}$$

Importantly, in the case of the turnover objective, we cannot use Bellman equations within the labeling algorithm to eliminate all but one label per state of the state graph. In fact, for every state $\Theta_k, k \in \{0, ..., n^i + 1\}$, we must conserve *all* labels generated by propagation which are *pareto optimal* with respect to the time value *and* the completion value. The reason is that a low completion value in a state contributes directly *in that moment* to the objective value, and a low time value improves the objective value over the *long term*. The need to conserve the Pareto-optimal front of labels for the turnover objective is further illustrated by Example 2.

The *optimal* sum of order completion times for the given OBSRP-R instance equals the *lowest* completion value of the terminal state, $\Omega^{*,\text{comp}}(\Theta_{n^i+1})$, and the optimal average turnover time $z^{*,\text{turnover}}$ follows straightforwardly by formula (3.6).

Example 2. *Consider the following instance I^{ex_1} with three single-item orders $o_1 = \{1\}$, $o_2 = \{2\}$ and $o_3 = \{3\}$, and arrival times $r_2 = r_2 = r_3 = 0$. The picker moves at unit speed $v = 1$, with a batching capacity of $c = 2$ and neglectable picking time $t^p = 0$. Consider the following distances: $d(l_d, 1) = 3, d(l_d, 2) = 5, d(1, 2) = 4$ and $d(l_d, 3) = 2$. There are two possible paths in the state graph to reach state $\Theta_2 = (2, 0, \{\}, \{o_3\})$:*

Path $P1 := \Theta_0 \rightarrow \Theta_1^1 = (1, 0, \{\}, \{o_2, o_3\}) \rightarrow \Theta_2$ represents the picking of orders o_1 and o_2 in two separate batches and leads to a label of time value $\Omega^{1,time}(\Theta_2) = 2d(l_d, 1) + 2d(l_d, 2) = 16$ and completion value $\Omega^{1,comp}(\Theta_2) = \Omega^{comp}(\Theta_1^1) + 1 \cdot \Omega^{1,time}(\Theta_2) = 6 + 16 = 22$.

Path $P2 := \Theta_0 \rightarrow \Theta_1^2 = (1, 1, \{\}, \{o_2, o_3\}) \rightarrow \Theta_2$ represents the joint picking of orders o_1 and o_2 in one batch and leads to a label of time value $\Omega^{2,time}(\Theta_2) = d(l_d, 1) + d(1, 2) + d(l_d, 2) = 12$ and completion value $\Omega^{2,comp}(\Theta_2) = \Omega^{comp}(\Theta_1^2) + 2 \cdot \Omega^{2,time}(\Theta_2) = 0 + 2 \cdot 12 = 24$.

We see that path $P1$ renders a lower sum of completion times to reach state Θ_2. Thus, considering only orders o_1 and o_2, this path would lead to the best objective value. However, if order o_3 must be picked, path P_2 leads to a better objective value than path P_1. This is because the label of $\Theta_3 = (3, 0, \{\}, ())$ through P_1 has the values $\Omega^{1,time}(\Theta_3) = \Omega^{1,time}(\Theta_2) + 2d(l_d, 3) = 16 + 4 = 20$, $\Omega^{1,comp}(\Theta_3) = \Omega^{1,comp}(\Theta_2) + 1 \cdot \Omega^{1,time}(\Theta_3) = 22 + 20 = 42$, and the label of Θ_3 through P_2 has the values $\Omega^{2,time}(\Theta_3) = \Omega^{2,time}(\Theta_2) + 2d(l_d, 3) = 12 + 4 = 16$, $\Omega^{2,comp}(\Theta_3) = \Omega^{2,comp}(\Theta_2) + 1 \cdot \Omega^{2,time}(\Theta_3) = 14 + 16 = 42$. The lower completion value $\Omega^{2,comp}(\Theta_3)$ is a better candidate for the final objective value.

3.4.2.3 Heuristic labeling for the turnover objective

In the exact approach from the previous section, the number of Pareto-optimal labels that must be preserved could be substantial. This renders the solution approach intractable even for small-sized instances in some settings. Therefore, we present a heuristic alternative, which only conserves *one* label for each state of the state graph.

Again in this algorithm, labels are propagated in a forward-inductive motion through the feasible transitions of the graph, and every newly generated label for every state $\Theta_{k+1}, k \in \{0, ..., n^i\}$ receives an associated time value $\Omega^{\text{time}}(\Theta_{k+1})$ by the same formula (3.8) as in the previous sections. In this algorithmic variant, also an additional *forward-looking completion value*, noted $\Omega^{\text{comp+}}(\Theta_{k+1})$ is computed. The latter represents the sum of the completion times for all the orders that are *already completed* at this state – plus the value of the current time of the state (i.e. $\Omega^{\text{time}}(\Theta_{k+1})$) summed up for all orders that are completed *in future states*. Intuitively, a low forward-looking completion value leverages both the long-term benefits of a low time value *and* the short-term benefits of a low completion value from the previous section.

In parallel and similarly to the time value, the forward-looking completion value of Θ_0 is zero, then the value in new labels is propagated inductively through the feasible transitions $x_k \in X(\Theta_k)$ for every label of every state Θ_k at stage $k \in \{0, ..., n^i\}$ as follows:

$$\Omega^{\text{comp+}}(\Theta_{k+1}) = \Omega^{\text{comp+}}(\Theta_k) + g^{\text{comp+}}(\Theta_k, x_k, \Omega^{\text{time}}), \tag{3.14}$$

where $g^{\text{comp+}}(\Theta_k, x_k, \Omega^{\text{time}})$ is the *immediate forward-looking completion* of the transition. The latter sums up the duration of the transition (i.e. $g^{\text{time}}(\Theta_k, x_k, \Omega^{\text{time}})$) for every uncompleted order at stage $\Theta_k = (s_k, m^o, S^{\text{batch}}, O^{\text{pend}})$:

$$g^{\text{comp+}}(\Theta_k, x_k, \Omega^{\text{time}}) = g^{\text{time}}(\Theta_k, x_k, \Omega^{\text{time}}) \cdot (m^o + |O^{pend}|) \qquad \forall k \in \{0, ..., n^i\} \tag{3.15}$$

In this heuristic, we just conserve *one label* per state $\Theta_{k+1}, k \in \{0, ..., n^i\}$ with *lowest forward-looking completion value*:

$$\Omega^{*,\text{comp+}}(\Theta_{k+1}) = \min_{(\Theta_k, x_k) \in f^{-1}(\Theta_{k+1})} \{\Omega^{\text{comp+}}(\Theta_k) + g^{\text{comp+}}(\Theta_k, x_k, \Omega^{\text{time}})\}, \tag{3.16}$$

Thereby ignoring the associated time value in the selection. Finally, the algorithm outputs $\Omega^{*,\text{comp+}}(\Theta_{n^i+1})$ of the terminal state and computes the suggested average order turnover time straightforwardly with from formula (3.6).

As we see in Example 3 the algorithm is indeed a *heuristic*, as it provides feasible solutions, which may be *sub-optimal* in the case of *waiting times* for incoming orders. In fact, the approach always finds the *optimal solution*, if the latter involves *no waiting time*.

Example 3. *Remember instance I^{ex_1} from Example 2. For I^{ex_1}, the heuristic labeling approach finds the optimal solution. Indeed, the time values of $\Omega^{time}(\Theta_1^1), \Omega^{time}(\Theta_1^2)$, $\Omega^{1,time}(\Theta_2), \Omega^{2,time}(\Theta_2)$ as in Example 3 remain, and the associated forward-looking completion values are: $\Omega^{comp+}(\Theta_1^1) = 3 \cdot 6 = 18, \Omega^{comp+}(\Theta_1^2) = 3 \cdot 3 = 9, \Omega^{1,comp+}(\Theta_2)$ $= 18 + 2 \cdot 10 = 38, \Omega^{2,comp+}(\Theta_2) = 9 + 3 \cdot 9 = 36$. Path $P2$ leads to a lower forward-looking completion time and thus, only the associated label remains which is propagated towards a new label for Θ_3, rendering the forward-looking completion value of $\Omega^{*,comp+}(\Theta_3) = 36 + 1 \cdot 4 = 40$ (which is indeed, better than the path through $P1$ would have rendered, which would be 42 in this case.)*
Now, consider an instance I^{ex_2} with the same parameters $v, t^p, c, o_1, o_2, o_3, r_1, r_2$, distance matrix d as for I^{ex_1}, but arrival time $r_3 = 17$. The algorithm for I^{ex_2} would again, by the same calculations as above, discard the label of $P1$ and only conserve the label of $P2$ at state Θ_2. This however, is an erroneous decision given the waiting time at state Θ_3: The immediate time g^{time} for the transition between Θ_2 with time value

$\Omega^{2,time}(\Theta_2) = 12$ and Θ_3 is $\max\{d(l_d,3); r_3 - \Omega^{2,time}(\Theta_2)\} + d(l_d,3) = 7$, leading to the sum of order completion times $\Omega^{*,comp+}(\Theta_3) = \Omega^{*,comp+}(\Theta_2) + 1 \cdot 7 = 43$. The alternate path to Θ_3 through $P1$ would have led to no waiting time and a superior sum of order completion times of 42.

3.4.3 Dominance rules for transitions in the DP formulation

In this section, we show how some transitions in the state graph of the dynamic program (DP) can be omitted, and, thus, the complexity of the DP can be reduced, without compromising the optimality of the labeling approaches from Sections 3.4.2.1 and 3.4.2.2.

For a given instance, a feasible solution σ is said to be *weakly dominated*, if there exists another feasible solution σ' with an identical or better objective value, i.e. if $z^{\text{makesp}}(\sigma') \leq z^{\text{makesp}}(\sigma)$ or $z^{\text{turnover}}(\sigma') \leq z^{\text{turnover}}(\sigma)$, in the case of the makespan and turnover objective, respectively. Remember that by Proposition 3.4.1, feasible solutions correspond to paths in the state graph from initial to terminal state, called *complete paths* in the following. Hereafter, we say that a *transition* in the state graph is *weakly dominated*, if, for *every* complete path involving this transition, the corresponding feasible solution is weakly dominated by another feasible solution, whose complete path does *not* involve this transition. By definition, weakly dominated transitions may be omitted in the DP.

Propositions 8, 9 and 10, identify weakly dominated transitions, based on the release time of their next selected picking item: intuitively, dominance occurs when it lies too far in the future. The propositions hold in case of the makespan objective, *and* in case of the turnover objective.

Proposition 8. *Consider a current state $\Theta_k = (s_k, m^o, S^{batch}, O^{pend})$ with $|S^{batch}| \geq 1$ and an associated time value $\Omega^{time}(\Theta_k)$. Then any transition $x_k \in X(\Theta_k)$ that visits next picking location $s_{k+1} \in o_j, o_j \in O^{pend}$ is weakly dominated, if*

$$r_j \geq \Omega^{time}(\Theta_k) + \frac{1}{v} \cdot (d(s_k, s) + d(s, s_{k+1})) + t^p \qquad (D1)$$

for some $s \in S^{batch}$.

Proof. See Supplement 3.8.2.1. □

In other words, Proposition 8 prohibits selecting item s_{k+1} with a late release time, if there is another item of a commenced order in the current batch ($|S^{\text{batch}}| \geq 1$) that can be picked first.

Proposition 9. *Consider a current state $\Theta_k = (s_k, m^o, \{\}, O^{pend})$ with $m^o > 0$ and an associated time value $\Omega^{time}(\Theta_k)$. Then any transition $x_k \in X(\Theta_k)$ that visits next picking location $s_{k+1} \in o_j, o_j \in O^{pend}$ is weakly dominated, if*

$$r_j \geq \Omega^{time}(\Theta_k) + \frac{1}{v}(d(s_k, l_d) + d(l_d, s_{k+1})) \qquad (D2)$$

Proof. See Supplement 3.8.2.2. □

In other words, Proposition 9 requires completing the current batch before moving to item s_{k+1} with a late enough release time, if all the items in the commenced orders of the current batch have been picked.

Finally, Proposition 10 describes the dominance of feasible transitions from batch-completion states. It prevents starting a new batch with a pending order $o_j \in O^{\text{pend}}$ that has a late release time, if we can complete another pending order $o_{\underline{j}}$ in a one-order batch first.

Proposition 10. *Consider a current state $\Theta_k = (s_k, 0, \{\}, O^{pend})$ and an associated time value $\Omega^{time}(\Theta_k)$. Then, any transition $x_k \in X(\Theta_k)$ that visits next picking location $s_{k+1} \in o_j, o_j \in O^{pend}$ is dominated, if there exists $o_{\underline{j}} \in O^{pend}$ such that*

$$r_j \geq \max\{\Omega^{time}(\Theta_k); r_{\underline{j}}\} + \chi(o_{\underline{j}}) + \frac{1}{v} \cdot d(l_d, s_{k+1}) \qquad (D3)$$

Where $\chi(o_{\underline{j}})$ is the minimum time to pick order $o_{\underline{j}}$ in a single-order batch.

Proof. See Supplement 3.8.2.3. □

The dominance rules $(D1)$, $(D2)$, and $(D3)$ from the previous Propositions are still valid, if we replace the right-hand side of the inequalities by an upper bound. Specifically, we introduce in each case a bound (*threshold*) that is independent of the particular picking locations, to efficiently integrate the dominance rules in the proposed algorithms. In this way, at the propagation of labels for each state $\Theta_k, k \in \{0, ..., n^i\}$ within DP, transitions $x_k \in X(\Theta_k)$, which select the next item s_{k+1} to pick, can be considered in the non-decreasing order of the release times of these items. Then, the propagation of labels for Θ_k can be *interrupted immediately*, when one of the release times $r(s_{k+1})$ exceeds the associated threshold (see Algorithm 1 for the efficient implementation of transitions within DP). In particular, we consider the following adapted dominance rules derived from (D1), (D2) and (D3):

For state $\Theta_k = (s_k, m^o, S^{\text{batch}}, O^{\text{pend}})$ and an associated time value $\Omega^{\text{time}}(\Theta_k)$, any transition $x_k \in X(\Theta_k)$ that visits next a picking location $s_{k+1} \in o_j, o_j \in O^{\text{pend}}$ is weakly dominated, if:

$$r_j \geq \begin{cases} \Omega^{\text{time}}(\Theta_k) + \frac{1}{v}(L+W) + t^p & \\ & \text{if } m^o > 0 \text{ and } S^{\text{batch}} \neq \{\} \qquad (\overline{D1}) \\ \Omega^{\text{time}}(\Theta_k) + \frac{1}{v}(L+W) & \\ & \text{if } m^o > 0 \text{ and } S^{\text{batch}} = \{\} \qquad (\overline{D2}) \\ \max\{r_{j_{min}}; \Omega^{\text{time}}(\Theta_k)\} + \frac{1}{v} \cdot (3L + & (a(o_{j_{min}}) + 2)W) + |o_{j_{min}}| \cdot t^p \\ & \text{if } m^o = 0 \qquad (\overline{D3}) \end{cases}$$

In $(\overline{D3})$, $o_{j_{min}}$ is the first order in the sequence O^{pend}, i.e. the uncompleted order with the smallest release time $r_{j_{min}}$, and $a(o_{j_{min}})$ represents the number of aisles which contain picking items of $o_{j_{min}}$. The thresholds in $(\overline{D1})$-$(\overline{D3})$ are indeed upper bounds for the right-hand sides of $(D1)$-$(D3)$, since $d(i,s) \leq L + W$ for all $s, i \in S \cup \{l_d\}$ and $\chi(o_j) \leq \frac{1}{v}(2L + (a(o_j) + 2)W + |o_j| \cdot t^p$ for all $j \in [n^o]$. The latter follows directly from Lemma 1 in Chapter 2, by reducing the warehouse to the relevant aisles for order o_j.

3.5 Analytical properties of online policies for the makespan objective

For the makespan objective in online OBSRP (OOBSRP), we can formulate several analytical properties. Lemma 6 states that, *ceteris paribus*, given an optimal policy and a fixed instance, the outcome will not worsen if we consolidate all system waiting and idle times and place them at the beginning. After this initial waiting period, the picker

Algorithm 1 Non-dominated transitions from state $\Theta_k = (s_k, m^o, S^{\text{batch}}, O^{\text{pend}})$

```
1: Let $o_{j_{min}} \in O^{\text{pend}}$ be the first order with the minimum release time in $O^{\text{pend}}$.
2: for all $s_{k+1} \in S^{\text{batch}}$ in arbitrary order do
3:     perform applicable transition(s) of Table 3.3
4: end for
5: for all $o_j \in O^{\text{pend}}$ in the given order of $O^{\text{pend}}$ do
6:     if $(\overline{D1})$-$(\overline{D3})$ is True then
7:         break and go to line 13;
8:     else
9:         for all $s_{k+1} \in o_j$ in arbitrary order do
10:            perform applicable transition(s) of Table 3.3
11:        end for
12:    end if
13: end for
```

can proceed with forming batches and routing following the original policy, but avoiding further delays.

Lemma 6. *Consider a fixed instance of OOBSRP with the makespan objective, and any perfect-anticipation solution (CIOS) for this instance, with picking sequence π^* and picking schedule $C^*(s), s \in S$. Let with $w^*(I)$ be the sum of all waiting and idle times of the associated schedule. Then there exists an alternative optimal solution, with identical makespan, picking sequence and batches, with an adjusted schedule $\tilde{C}^*(s), s \in S$, in which the picker starts the operation at time $w^*(I)$:*

$$\tilde{C}^*(\pi^*[1]) = \max\{w^*(I) + \frac{1}{v} \cdot d(l_d, \pi^*[1]), r(\pi^*[1])\} + t^p \tag{3.17}$$

Proof. The lemma follows directly from formulas (3.1) and (3.2). Details are provided in Supplement 3.8.3.1. □

Strategic relocation is a form of anticipation where the picker moves towards the picking position of an item before its actual arrival, it occurs frequently in the perfect-anticipation solutions (CIOSs) (see Section 3.6.2). Some studies in the routing literature demonstrated the positive effects of anticipatory routing during idle times (D. J. Bertsimas & van Ryzin, 1993). Lemma 7 limits the benefits of anticipatory strategic relocation of the picker in the case of the makespan objective. Specifically, it compares the optimal anticipatory policy with an algorithm ALG that constructs the same routes and the same batches as the associated CIOS, without performing strategic relocation, i.e. the picker in ALG remains at its last position until the arrival of the next planned picking item. The improvement potential for ALG is limited by the warehouse's dimensions $(W + L)$.

Lemma 7. *For any instance I of OOBSRP with the makespan objective, and any algorithm ALG which performs optimal batching, optimal routing, but no strategic relocation, the following holds:*

$$z^{ALG}(I) \leq z^*(I) + \max_{i,j \in S \cup \{l_d\}} \{d(i,j)\} \tag{3.18}$$

where $z^{ALG}(I)$ is the makespan of ALG, $z^*(I)$ is the optimal makespan with perfect anticipation, and S are the picking locations of instance I.

Proof. The lemma is proven by induction in Supplement 3.8.3.2 □

3.6 Results

E-commerce warehouses strive to balance low costs with quick order turnovers, though these objectives often conflict. To sharpen our observations, we analyze two idealized systems focusing solely on one objective, either *makespan* minimization (representing picker's time and wage costs) or order *turnover* minimization (representing average order turnover). Although these systems do not exist in pure form, our main conclusions align and lead to actionable recommendations that improve *both objectives simultaneously*.

For our experiments, we simulate a two-block rectangular warehouse with $b = 3$ cross-aisles and $a = 10$ aisles, following standard simulation frameworks from the literature. The depot is located on the central cross-aisle on the left side of the warehouse. The picker moves at $v =$0.8 m/sec and spends $t^p =$10 sec to collect an item at each picking location. In the *Basis* setting orders average 1.5 items (*Smallorders*), the order arrival rate is $r = 200$ orders/8h and the picking cart has a capacity of $c = 2$ bins. For comparison, orders at German Amazon warehouses average 1.6 items (Boysen et al., 2019; Xie et al., 2023). We also generate three additional settings using a one-factor-at-a-time design by examining larger orders (*Largeorders*, with 2.5 items per order on average), an increased order arrival rate ($r = 250$ orders/8h), and a larger cart capacity ($c = 4$ bins). We use the designed algorithm DP, which can solve instances with up to 70 item locations in a reasonable time, to generate CIOSs. We independently generate 20 instances per setting with $n^o = 15$ orders each, a standard instance size for similar types of analysis in the warehousing literature (cf. Bukchin et al., 2012). Additionally for the makespan objective, we confirm the *representativeness* of our analysis for larger instances using the nonparametric *Kruskal-Wallis test (KWT)* on independently generated *Basis* instances with $n^o = 15, 18,$ and 21 orders. Under general assumptions, the KWT assesses whether the measured metrics for instances with $n^o = 15$ orders remain consistent for instances with $n^o = 18$ and 21 orders. We expect the outcome of KWT for the turnover objective to mirror that of the makespan. In total, we use $20 \cdot (4 + 2) = 120$ instances in the analysis of CIOSs.

For the *makespan* objective, we computed the CIOSs for all instances using the *exact* DP approach from Section 3.4.2.1. After validating its performance in Section 3.6.1, we generated the associated CIOSs for the *turnover* objective with *heuristic* DP from Section 3.4.2.3.

All experiments have been conducted on the Compute Canada cluster using a maximum of 350 GB RAM and only one CPU for the DP approaches, but allowed the parallel computation with 32 CPUs (1 GB RAM each) per instance for the MIP approaches used as a benchmark in Section 3.6.2.The DP approaches were implemented with Python 3.11.4. and Gurobi 11.0.0. was used to solve the MIP models.

In the following, after validating the designed DP algorithms in Section 3.6.1, the main analysis of this paper, the investigation of CIOSs, is presented in Section 3.6.2.

3.6.1 Performance of the designed DP algorithms

Tables 3.4 and 3.5 validate the *exact* DP approach from Section 3.4.2.1 and the *heuristic* DP approach from Section 3.4.2.3 by comparing their performance to a standard MIP solver (refer to Supplement 3.8.1 for MIP models), using a one-hour runtime limit. The MIP solver had the advantage of utilizing parallelization with 32 CPUs, while the DP approaches did not use parallelization.

The MIP solver was not able to solve *any* instance to proven optimality, likely due to weak lower bounds.

Table 3.4: Performance of exact DP-approach for makespan objective compared to benchmarks for runtime limit of 1h

Instances			DP-approach			Gains of dominance rules			MIP solver		
		items n^i	Runtime (sec)				Runtime reduction			Opt. gap of UB	
Setting	n^o	med (max)	avg	max	#opt	#opt	avg	max	LB	avg	max
LargeOrders_c2_r200	15	37.5 (44)	1478	3430	18	0	26%	85%	–	11.6%	37.3%
SmallOrders_c2_r200	15	22 (26)	51	119	20	0	49%	99%	–	0.5%	3.7%
SmallOrders_c2_r250	15	22.5 (27)	74	123	20	0	27%	82%	–	0.9%	4.1%
SmallOrders_c4_r250	15	22.5 (25)	1428	3226	20	3	30%	72%	–	0.8%	4.4%
SmallOrders_c2_r200	18	27 (30)	755	1531	20	0	43%	96%	–	3.1%	13.0%
SmallOrders_c2_r200	21	29 (34)	1216	3396	5	5	–	–	–	0.0%	0.1%

Note. Column 3, Items n^i med (max): median (maximum) number of items in instance; *Column 6, DP-approach #opt:* number instances out of 20 solved in 1h to optimality. *Column 7, Gains of dominance rules, #opt:* additional optima found in 1h due to dominance rules; *Column 10, MIP solver LB:* lower bound found by MIP-solver; *Column 11-12, MIP solver Opt. gap of UB:* gap between upper bound of solver and DP solution. All values refer to the subset of instances solved by DP within 1h.

Table 3.5: Performance of heuristic DP-approach for turnover objective compared to MIP solver for runtime limit of 1h

Instances			DP-approach				MIP solver			
		items n^i	Runtime (sec)					gap of UB to DP sol		
Setting	n^o	med (max)	avg	max	#sol	#sol $\leq$ MIP UB	LB	avg	max	min
LargeOrders_c2_r200	15	37 (40)	1498	2512	16	16 (100%)	–	38%	141%	7%
SmallOrders_c2_r200	15	22 (26)	62	138	20	18 (90%)	–	9%	49 %	-7%
SmallOrders_c2_r250	15	22.5 (27)	93	159	20	20 (100%)	–	8%	26%	1%
SmallOrders_c4_r250	15	22 (25)	1576	3267	19	19 (100%)	–	27%	169%	2%
SmallOrders_c2_r200	18	27 (30)	955	1801	20	20 (100%)	–	13%	32%	1
SmallOrders_c2_r200	21	30 (34)	810	2410	4	4 (100%)	–	18%	27%	5%

Note. Column 3, Items n^i med (max): median (maximum) number of items in instance; *Column 6, DP-approach #sol:* number instances out of 20 solved in 1h (heuristically). *Column 7, DP-approach, #sol $\leq$ MIP UB:* number of DP-solutions that are better than the MIP's upper bound; *Column 8, MIP LB:* lower bound found by MIP-solver; *Columns 9-11, MIP solver gap of UB to DP:* gap between upper bound of solver and DP solution. All values refer to the subset of instances solved by DP within 1h.

For the *makespan* objective, DP solved almost all (103 out of 120) instances to *proven optimality* (see Table 3.4). Customized dominance rules were highly effective, reducing the DP's runtime by an average of 36%, and up to 99% in some cases, enabling the optimal solution of 8 additional instances.

For the *turnover* objective (see Table 3.5), the DP produced significantly better solutions than the MIP solver, with 98% of the solutions improving upon the MIP's upper bound.

Note that an increase in batching capacity significantly impacts the runtime of the DP approaches for both objectives.

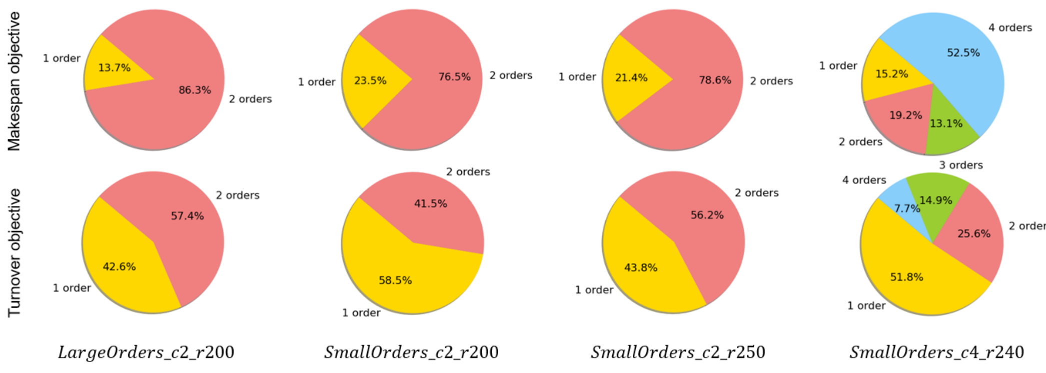
For our analysis of the CIOSs in the following sections, we ran the DP without a time limit for both objectives. Specifically, we computed *optimal solutions for all 120 instances* in the dataset for the makespan objective. Overall, Tables 3.4 and 3.5) emphasize that without the customized design of the DP algorithms, the CIOS analysis in the following sections would have been impossible.

3.6.2 Analysis of complete-information optimal solutions (CIOSs)

Depending on the objective, the operations of the warehouse change.

The *makespan* is effectively minimized in warehouses with high order arrival rates: With more frequent arrivals, long queues allow the picker to select orders whose items

Figure 3.5: Portion of batches of different sizes



Note. yellow: 1-order batches; red: 2-order batches; green: 3-order batches; blue:4-order batches.

are located close to each other into the same batch, thus, reducing the picking time per order. High arrival rates also decrease idle time, as there are fewer moments when no orders are available. As a result, most batches in CIOSs utilize the full capacity of the picking cart (see Figure 3.5). For a four-bin cart (*Small_c4_r200*), an average of 53% of all batches are composed of four orders. In the remaining settings with two-bin carts, batches of two orders account for 77-86% of all batches.

In contrast, warehouses that prioritize average *turnover* achieve the lowest turnover times at low order arrival rates. Here, the picker often remains idle at the depot, enabling immediate picking upon an order's arrival, usually in a single-order batch. Consequently, 48% of CIOS batches are single-order batches (see Figure 3.5). Indeed, picking a smaller order as a single-order batch first can offset savings from collecting multiple orders simultaneously, resulting in a smaller average order turnover.

We continue by examining batching (Section 3.6.2.1), routing (Section 3.6.2.2), and anticipation in CIOSs. Strategic waiting is discussed separately in Section 3.6.2.3, followed by additional forms of anticipation in Section 3.6.2.4. We focus on identifying decisions that improve *both* objectives simultaneously. For clarity, we compare certain decisions in CIOSs with two common planning heuristics: *first-in-first-out (FIFO)* batching and *myopic reoptimization (Reopt)*. FIFO-batching forms batches by assigning orders as they arrive, while *Reopt* is a purely myopic policy that reoptimizes based on the current queue of available orders.

3.6.2.1 Batching in CIOSs

Batching decisions are the most costly in terms of runtime when generating CIOS. This section contrasts batching decisions in CIOS with FIFO-batching, and makes a general observation how CIOS handle dynamically arriving orders.

Table 3.6 shows significant differences between CIOS and FIFO batches for both examined objectives – *makespan* and *turnover*. Unlike FIFO, 23%-34% of CIOS batches do not contain consecutively arrived orders for the former objective, 14%-29% of batches for the latter. When arranging orders in each CIOS batch by arrival time for the *makespan* objective, about 30% of orders are at altered positions in the resulting sequence compared to FIFO. The order sequence appears more important for warehouses operating under the *turnover* objective (overall share of orders with altered positions of 37%).

A primary reason for deviating from the FIFO order sequence in batching is to improve the matching of jointly collected orders. *In case of* $c = 2$, this matching quality can be measured as savings (cf. De Koster et al., 1999). The *savings* $\psi(B)$ for a batch B is

Table 3.6: Batching decisions in CIOSs compared to those in FIFO

Obj.	Setting	n^o	# batches with non-consecutive orders	# orders with altered positions
Makesp. Analysis	*LargeOrders_c2_r200*	15	2.7 (34%)	5.1 (34%)
	SmallOrders_c2_r200	15	2.2 (26%)	4.0 (26%)
	SmallOrders_c2_r250	15	1.9 (23%)	4.2 (29%)
	SmallOrders_c4_r250	15	1.3 (25%)	4.0 (26%)
	Overall	15	2.0 (27%)	4.3 (29%)
Makesp. Valid.	*SmallOrders_c2_r200*	15	26%	26%
	SmallOrders_c2_r200	18	23%	28%
	SmallOrders_c2_r200	21	18%	17%
	Kruskal Wallis test p-values		0.35	0.08
Turnover	*LargeOrders_c2_r200*	15	2.6 (29%)	6.5 (43%)
	SmallOrders_c2_r200	15	1.6 (16%)	5.0 (33%)
	SmallOrders_c2_r250	15	2.3 (24%)	5.7 (38%)
	SmallOrders_c4_r250	15	1.0 (14%)	5.4 (36%)
	Overall	15	1.8 (21%)	5.6 (37%)

Note. Average number of batches with non-consecutive orders and average number of orders that deviate from their FIFO position, per instance.

defined as the difference between the shortest picking time for B (noted $\chi(B)$) and the sum of the individual shortest picking times for each order in B (noted $\chi(o), o \in B$). The measure $\psi(B)$ is normalized from 0 to 1, where 0 indicates no savings (picking each order separately takes the same time as picking them jointly) and 1 indicates the maximum possible savings (picking both orders separately takes twice as long as picking them jointly):

$$\psi(B) = \frac{\sum_{o \in B} \chi(o) - \chi(B)}{\chi(B)} \tag{3.19}$$

For settings with $c = 2$, Table 3.7 compares the savings in two-order CIOS batches with those of *randomly* selected two-order batches. For the random batches, we sample from all possible two-order combinations for the same instance. Observe that CIOS batches have higher savings: about 70% of CIOS batches exceed the median savings of randomly generated batches for the *makespan* objective, and 77% for the *turnover* objective, respectively. The greater savings for the *turnover* objective can be attributed to higher opportunity costs associated with batching in this case, as discussed in the introduction to Section 3.6.2.

Observation 1 (Batching)**.** *Batching in CIOSs differs significantly from the myopic FIFO-rule, in particular, CIOS batches show a deliberate improvement in the quality of order matching.*

Upon a closer examination, batching in CIOS differs from most batching policies discussed in the warehousing literature (see Section 3.2 for an overview) in one key aspect: most orders are added to a batch *after* the picker has left the depot to collect items of this batch. This applies to 64% and 62% orders for the *makespan* and *turnover* objectives, respectively (see Table 3.8). Policies that allow modifications to an active batch, like in CIOS, are termed *interventionist*, while *non-interventionist* policies do not allow changes once a batch is started (cf. Giannikas et al., 2017). Similar to strategic waiting, intervention allows for savings from batching when order queues are short. However, since the picker does not stay idle but continues moving towards known orders instead of waiting, intervention is a *less costly* alternative to strategic waiting in terms of the opportunity costs, especially if no new orders arrive quickly.

Table 3.7: Quality of batches in CIOS: Savings of two-order batches in settings with batching capacity $c = 2$

Obj.	Setting	n^o	Mean CIOS batch savings	# CIOS batch savings > random median	Mean random batch savings
Makesp. Analysis	*LargeOrders_c2_r200*	15	0.36	4.8 (69%)	0.28
	SmallOrders_c2_r200	15	0.33	4.4 (68%)	0.26
	SmallOrders_c2_r250	15	0.37	4.7 (71%)	0.27
	Overall	15	0.35	4.6 (70%)	0.27
Makesp. Valid.	*SmallOrders_c2_r200*	15	0.33	68%	0.26
	SmallOrders_c2_r200	18	0.38	75%	0.27
	SmallOrders_c2_r200	21	0.33	71%	0.26
	Kruskal Wallis test p-values		0.02	0.18	0.77
Turnover	*LargeOrders_c2_r200*	15	0.37	4.2 (72%)	0.28
	SmallOrders_c2_r200	15	0.37	3.4 (79%)	0.26
	SmallOrders_c2_r250	15	0.40	4.5 (81%)	0.27
	Overall	15	0.38	4.0 (77%)	0.27

Note. Columns 4, 6, 8: Average of the mean savings of the formed 2-order batches per instance. *Columns 5, 7*: Number of 2-order batches whose savings are larger than the median of the random 2-order batches of this instance, averaged over all instances.

Table 3.8: Orders added by intervention in CIOS

			Makespan objective	Turnover objective
	Setting	n^o	# orders	# orders
Analysis	*LargeOrders_c2_r200*	15	7.7 (5%)	7.8 (52%)
	SmallOrders_c2_r200	15	10.3 (69%)	10.7 (71%)
	SmallOrders_c2_r250	15	8.8 (59%)	8.9 (59%)
	SmallOrders_c4_r250	15	11.4 (76%)	10.1 (67%)
	Overall	15	9.5 (64%)	9.4 (62%)
Validation	*Smallorders_c2_r200*	15	69%	
	Smallorders_c2_r200	18	62%	
	Smallorders_c2_r200	21	69%	
	Kruskal-Wallis test; p-value		0.58	

Note. Average number of orders per instance added by intervention.

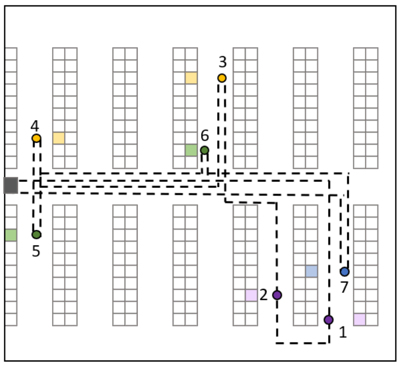
Observation 2 (Intervention in Batching). *Most orders in CIOSs are added to a batch dynamically, after the picker's departure from the depot to collect the items of this batch.*

3.6.2.2 Routing in CIOSs

Figure 3.6 shows a typical picker route for completing a batch in CIOSs, featuring three horizontal direction changes while crossing the central cross-aisle and two cases where sub-aisles (4th upper and 6th lower) are entered twice from the same side. We observed CIOS batch routes with up to five horizontal direction changes within the cross-aisles. This contrasts sharply with common routing heuristics like *S-shape* or *largest-gap* policies, where sub-aisles are entered at most once from each side, and horizontal direction changes while traversing the cross-aisles are limited to one and three, respectively (see Le-Duc & De Koster, 2007; Roodbergen & De Koster, 2001, for a detailed discussion). The key factor driving the CIOS routing pattern is *intervention* (cf. Observation 2), where newly arrived orders are added to an active batch.

Observation 3 (Routing). *CIOS routing differs significantly from common routing policies.*

Figure 3.6: Exemplary CIOS picker route for one batch



Note. The numbers indicate the visiting sequence. A CIOS route for an instance in $SmallOrders\_c4\_r250$ with $n^o = 15$ for the *makespan* objective.

3.6.2.3 Strategic waiting in CIOSs

As discussed in Section 3.1, strategic waiting – deliberately delaying picking to enhance future operations – is one of the least understood concepts in the warehousing literature.

Based on the analysis of CIOSs, we identify two types of strategic waiting. Note that, since CIOS operates with perfect anticipation, such waiting occurs at the first picking location of the future order:

- *Waiting for batch extension (B.ext.)*: The picker has collected all items for current orders but, instead of delivering the collected orders to the depot, waits in the field for the next order to add to the active batch, increasing savings from batching and avoiding additional trips to the depot.
- *Waiting for a better fit (B.fit)*: The picker waits at the picking location of a future order, even though items of other orders are available for picking, because the future order better fits the current batch route.

An illustrative example of waiting for B.fit is provided in Figure 3.2 in Section 3.1.

Table 3.9 compares strategic waiting and *idle time* in CIOSs. The latter occurs when all available orders have been picked and delivered to the depot. For both the *makespan* and *turnover* objectives, strategic waiting is significantly less than idle time. For the *makespan* objective, strategic waiting accounts for merely 5% of the makespan on average, compared to 15% of idle time; for the *turnover* objective, it is even 1% of waiting compared to 17% of idle time. Notably, while strategic waiting is frequent – occurring before 6% to 26% of orders depending on the setting and the objective – it is extremely brief, with the average waiting time lying between 12 and 70 seconds.

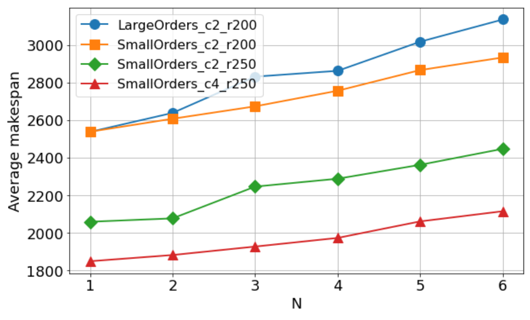
This explains the monotonous decrease in performance for both objectives of the (interventionist) *Reopt* policy when adopting a *variable time window strategy* (VTW) (cf. Gil-Borrás et al., 2024) which enforces picker waiting times at the depot, until there are at least $N \in \{1, ..., 6\}$ orders in the queue, see Figure 3.7. Even with the minimum amount of waiting for this strategy ($N = 2$) the waiting affects too many orders and is significantly longer than waiting in CIOS (depending on the setting and the objective, the average waiting lasts between 92 and 184 seconds), see Table 3.10. Furthermore, 21-52% of the waiting in this policy was *misplaced*, as the associated waiting times w_j for the arrival of the second order exceeded the time $\chi(o_j)$ in which the delayed order o_j could have been completed in a single-order batch.

Table 3.9: Strategic waiting in CIOSs

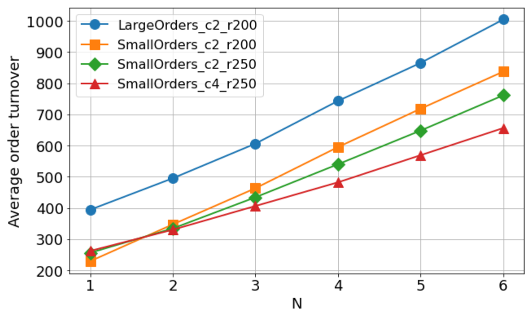
				Avg. total waiting time			Waiting for batch extension			Waiting for a better fit		
Obj.	Setting	n^o	Makesp.	Idleness	B.ext	B.fit	# orders	Avg time	80P time	# orders	Avg time	80P time
Makesp. Analysis	*LargeOrders_c2_r200*	15	2488	277 (10%)	15 (1%)	16 (1%)	0.6 (4%)	27	55	0.8 (5%)	22	32
	SmallOrders_c2_r200	15	2447	729 (27%)	77 (3%)	37 (1%)	1.1 (7%)	70	126	1.1 (7%)	35	46
	SmallOrders_c2_r250	15	1975	320 (15%)	58 (3%)	23 (1%)	1.1 (7%)	55	101	0.7 (5%)	35	50
	SmallOrders_c4_r250	15	1773	193 (9%)	111 (6%)	83 (4%)	1.7 (11%)	66	105	2.3 (15%)	36	67
	Overall	15	2171	380 (15%)	65 (3%)	40 (2%)	1.1 (7%)	59	106	1.2 (8%)	33	48
Makesp. Valid.	*SmallOrders_c2_r200*	15	2447	27%	3%	1%	7%	70	126	7%	35	46
	SmallOrders_c2_r200	18	2571	15%	7%	2%	12%	92	166	6%	39	36
	SmallOrders_c2_r200	21	3395	23%	6%	2%	14%	73	123	5%	52	69
	Kruskal Wallis test p-values		–	0.01	0.02	1.00	< 0.01	0.43	–	0.97	0.57	–
Turnover	*LargeOrders_c2_r200*	15	2569	275 (10%)	3 (0%)	17 (1%)	0.1 (1%)	34	51	0.8 (5%)	22	37
	SmallOrders_c2_r200	15	2496	791 (29%)	2 (0%)	16 (1%)	0.2 (1%)	12	20	1.2 (8%)	14	21
	SmallOrders_c2_r250	15	2016	374 (17%)	6 (0%)	11 (1%)	0.5 (3%)	14	21	0.8 (5%)	14	19
	SmallOrders_c4_r250	15	1842	294 (14%)	10 (0%)	13 (1%)	0.6 (4%)	17	32	1.0 (7%)	13	22
	Overall	15	2225	434 (17%)	5 (0%)	14 (1%)	0.3 (2%)	17	32	0.9 (6%)	15	24

Note. Columns 4-6: Average total time per instance spent waiting (as % of the makespan). *Columns 7 and 10, #orders*: Average number of orders per instance waited for in type B.ext- and B.fit-waiting, respectively. *Columns 8 and 11 (9 and 12), Avg time (80P time)*: Average (80-percentile) duration of B.ext- and B.fit-waiting, respectively.

Figure 3.7: Development of makespan and turnover when waiting for $N \in \{1, ..., 6\}$ orders at the depot (VTW) in *Reopt*



(a) Makespan objective



(b) Turnover objective

Table 3.10: Waiting times for $N = 2$ orders in *Reopt* (VTW)

	Makespan objective			Turnover objective		
Setting ($n^o = 15$)	# orders	Avg time	# $w_j > \chi(o_j)$	# orders	Avg time	# $w_j > \chi(o_j)$
LargeOrders_c2_r200	2.3 (15%)	134	22%	3.3 (22%)	116	23%
SmallOrders_c2_r200	3.8 (25%)	164	43%	4.5 (30%)	184	52%
SmallOrders_c2_r250	2.9 (19%)	103	21%	3.7 (25%)	115	33%
SmallOrders_c4_r250	2.8 (19%)	92	21%	3.9 (26%)	95	27%
Overall	3.0 (20%)	123	27%	3.9 (26%)	128	34%

Note. Columns 2 & 5, #orders: Average number of orders per instance delayed because of waiting for second order. *Columns 3 & 6, Avg time:* Average duration of waiting, after first order has arrived. *Columns 4 & 7,* $w_j > \chi(o_j)$*:* Percentage of delayed orders that could have been picked in one-order batch within waiting time.

We also assessed the common advice in the literature to avoid waiting if there are enough orders in the queue to complete a batch (c.f. Alipour et al., 2020; Giannikas et al., 2017; Gil-Borrás et al., 2024; Henn, 2012). Contrary to this advice, in 28% and 41% of the observed cases of waiting for a better fit in CIOSs, for the *makespan* and *turnover* objective, respectively, enough orders were available to form a full batch.

Differences in strategic waiting behavior in CIOSs are observed depending on the warehouse's objective.

Since one-order batches dominate under the *turnover* objective, strategic waiting aimed at better batching is less relevant. This is especially true for waiting for batch extensions, which is negligible—meaning the picker almost always returns orders to the depot when the order queue is empty. On average, strategic waiting accounts for only about 1% of the makespan and lasts just 12-17 seconds per occurrence in settings with small orders. For larger orders, where even one-order batches take longer, the tolerance for waiting is slightly higher (22-34 seconds per occurrence), though strategic waiting happens less frequently.

Observation 4 (Turnover: Strategic Waiting)**.** *Under the turnover objective, only a negligible amount of time is spent on strategic waiting, and each occurrence of strategic waiting is brief and well-timed.*

For the *makespan* objective, waiting for batch extension becomes more important than waiting for a better fit, accounting for 3% and 2% of the makespan, respectively. Additionally, when cart capacity is large ($c = 4$), creating more opportunities for batching, strategic waiting increases further, reaching 6% and 4% of the makespan for both types of strategic waiting, respectively. In this case, idle time is reduced ("cannibalized") by earlier strategic waiting (9% idle time for $c = 4$ compared to the 15% overall average).

Table 3.11: The effect of prior waiting (total idle + waiting time in CIOS) on reoptimization policy for the *makespan* objective

Setting ($n^o = 15$)	Improvement compared to standard *Reopt* Average	Best
LargeOrders_c2_r200	0.0%	0.0%
SmallOrders_c2_r200	0.1%	2.4%
SmallOrders_c2_r250	0.0%	0.2%
SmallOrders_c4_r250	0.3%	6.6%
Overall	0.1%	6.6%

Note. Average and best improvement per instance.

An interesting aspect of the *makespan* objective is that the precise *placement* of strategic waiting is almost irrelevant, as explained in Lemma 6 (Section 3.5). Table 3.11 tests the following hypothesis: Unlike the *turnover* objective, where both the amount and timing of waiting are critical, for the *makespan* objective, it may be sufficient to anticipate only the *amount of waiting*. By positioning this at the start and gaining additional information on incoming orders, we could potentially improve operations without needing to predict future orders in detail. To test this, we conducted an analysis where, for each instance I, we enforced initial waiting of $w(I)$ and applied myopic reoptimization (Reopt) afterwards. We calculated $w(I)$ as the sum of the total idle time and the strategic waiting time in the CIOS for this instance. As shown in Table 3.11, the results of *Reopt* did not improve in most cases, except for two instances in two different settings, where the improvement was significant (2.4% and 6.6%, respectively). This indicates that the benefits of additional information from enforced initial waiting were outweighed by the cost of waiting. In other words, strategic waiting alone is insufficient and must be paired with adjustments in order batching and sequencing based on the anticipation of future orders.

Observation 5 (Makespan: Strategic Waiting)**.** *For makespan objective, strategic waiting becomes more important with larger carts due to increased batching opportunities. Only the total waiting time matters, as it can be positioned at the start. However, to leverage strategic waiting, future order anticipation and adjustments in order batching and sequencing are necessary.*

3.6.2.4 Further forms of anticipation in CIOSs

In addition to strategic waiting, CIOSs employ other forms of anticipation. The first involves anticipatory adjustments in order batching, while the second involves anticipatory adjustments in routing.

To evaluate anticipatory batching, we compared the batching decisions in CIOSs with those in a purely myopic reoptimization policy (Reopt), by repeating the studies of Table 3.7 for the savings of two-order batches in *Reopt* solutions, see Table 3.12. Surprisingly, the savings of the *Reopt* batches are almost equivalent, especially for the *makespan* objective, where the overall mean batch savings have the equal value of 0.35, and where on average 69% of the two-order batch savings exceeded the random median savings (compared to 70% in CIOS). For the *turnover* objective, the *Reopt* two-order batches even have relatively better savings compared to CIOS, which is also due to *Reopt* solutions forming *fewer* two-order batches.

For the *makespan* objective, one type of situation, where batching in CIOS explicitly relied on *future* orders to perform better-quality batches, is the following: the picker returns a batch of β orders, while at this moment, there are exactly q other available

Table 3.12: Quality of batches in *Reopt*: Savings of two-order batches in settings with batching capacity $c = 2$

Obj.	Setting	n^o	Mean Reopt batch savings	# Reopt batch savings > random median	Mean random batch savings
Makesp.	*LargeOrders_c2_r200*	15	0.36	4.7 (71%)	0.28
	SmallOrders_c2_r200	15	0.33	3.8 (66%)	0.26
	SmallOrders_c2_r250	15	0.36	4.4 (70%)	0.27
	Overall	15	0.35	4.3 (69%)	0.27
Turnover	*LargeOrders_c2_r200*	15	0.39	3.5 (79%)	0.28
	SmallOrders_c2_r200	15	0.40	2.8 (92%)	0.26
	SmallOrders_c2_r250	15	0.46	3.2 (87%)	0.27
	Overall	15	0.42	3.2 (86%)	0.27

Note. Columns 4, 6,8: Average of the mean savings of the formed 2-order batches per instance. *Columns 5,7*: Number of 2-order batches whose savings are larger than the median of the random 2-order batches of this instance, averaged over all instances.

Table 3.13: Anticipative batching – strategic batch omission in CIOS for the *makespan* objective

	Setting	n^o	# batch omission
Analysis	*LargeOrders_c2_r200*	15	0.2 (2%)
	SmallOrders_c2_r200	15	0.2 (2%)
	Smallorders_c2_r250	15	0.5 (5%)
	Smallorders_c4_r250	15	0.5 (9%)
	Overall	15	0.3 (4%)
Validation	*SmallOrders_c2_r200*	15	2%
	SmallOrders_c2_r200	18	2%
	SmallOrders_c2_r200	21	4%
	Kruskal Wallis test p-value		0.80

Note. Average number of batches per instance with batch omission.

orders in the queue, that could have fitted within the same batch $0 < q \leq c - \beta$. In this CIOS decision, the batching of those q orders is omitted for a better fit in a future batch. A myopic policy on the other hand, would have formed the larger batch because of the subadditivity of the batching process. We dub this type of anticipative batching *strategic batch omission*.

Table 3.7 reports the occurrence of strategic batch omission in CIOSs. Although this effect is observable, it affects less than 5% of batches, for small cart capacities. The importance increases with the cart capacity.

Observation 6 (Anticipatory Batching)**.** *CIOSs seldom rely on future orders to achieve higher-quality batches. A myopic but optimal batching policy presents a valid alternative in terms of batch quality.*

While CIOSs optimize the picker routing for a given batch, there is a notable difference from common warehousing heuristics. In CIOSs, the picker uses idle time to move towards the next picking location for an upcoming order – a practice we term *strategic relocation*. This occurs before the first pick for 39-60% of orders in the *makespan* objective and 41-62% of orders in the *turnover* objective (see Table 3.14), accounting for 8-19% of the total completion time in both objectives.

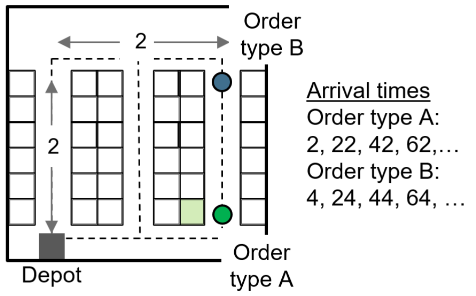
For the *makespan* objective, time savings from strategic relocation are constrained by the warehouse's dimensions (the time to traverse the warehouse's half-perimeter $(W + L)$), as discussed in Lemma 7. For the *turnover* objective, strategic relocation can significantly improve the turnover time of individual orders, leading to a noticeable enhancement of the overall objective (see example in Figure 3.8).

Table 3.14: Anticipative routing – strategic relocation in CIOS

	Setting	n^o	Makespan objective Time	Makespan objective # Orders	Turnover objective Time	Turnover objective # Orders
Analysis	*LargeOrders_c2_r200*	15	213 (8%)	5.9 (39%)	215 (8%)	6.1 (41%)
	SmallOrders_c2_r200	15	381 (15%)	9.0 (60%)	386 (15%)	9.3 (62%)
	SmallOrders_c2_r250	15	269 (14%)	7.1 (47%)	274 (13%)	7.6 (50%)
	SmallOrders_c4_r250	15	356 (19%)	8.8 (58%)	314 (17%)	8.0 (53%)
	Overall	15	305 (14%)	7.7 (51%)	297 (13%)	7.7 (52%)
Validation	*SmallOrders_c2_r200*	15	15%	60%		
	SmallOrders_c2_r200	18	15%	54%		
	SmallOrders_c2_r200	21	16%	60%		
	Kruskal Wallis test p-values		0.72	0.77		

Note. Average time picker spends with strategic relocation per instance, and average number of orders with strategic relocation before first pick.

Figure 3.8: Example of strategic relocation for the *turnover* objective



Note. Consider $c = 2$ and unit speed. Average turnover in CIOS is 10 (e.g. by picking pairs of A- and B-orders in batches). For any policy without strategic relocation, the average turnover cannot be better than 14.

Observation 7 (Anticipatory Routing). *CIOSs make extensive use of strategic relocation, where the picker moves to the next order's picking location during idle time.*

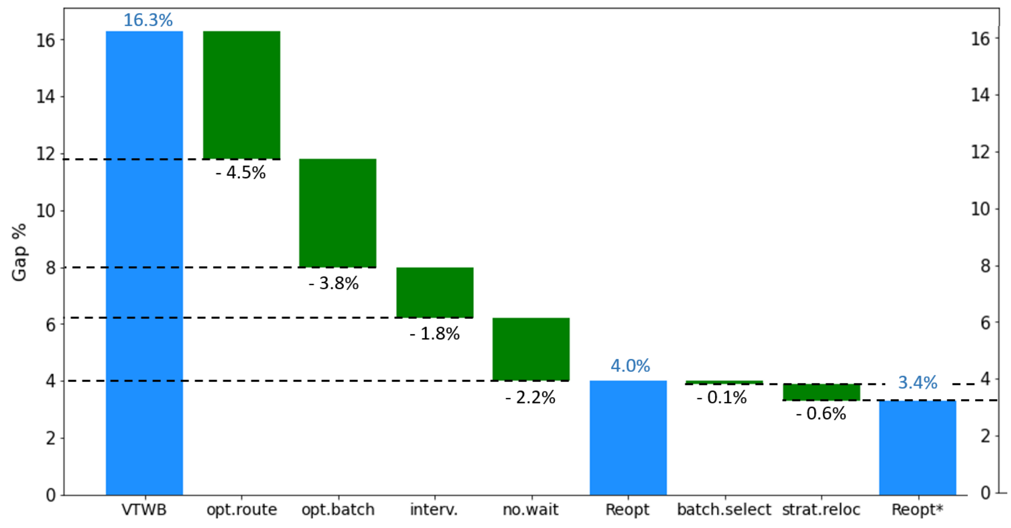
3.6.3 Discussion: Design of good online policies

In this section, we build upon the analysis from Section 3.6.2 to design effective online policies and estimate the potential for advanced anticipatory techniques.

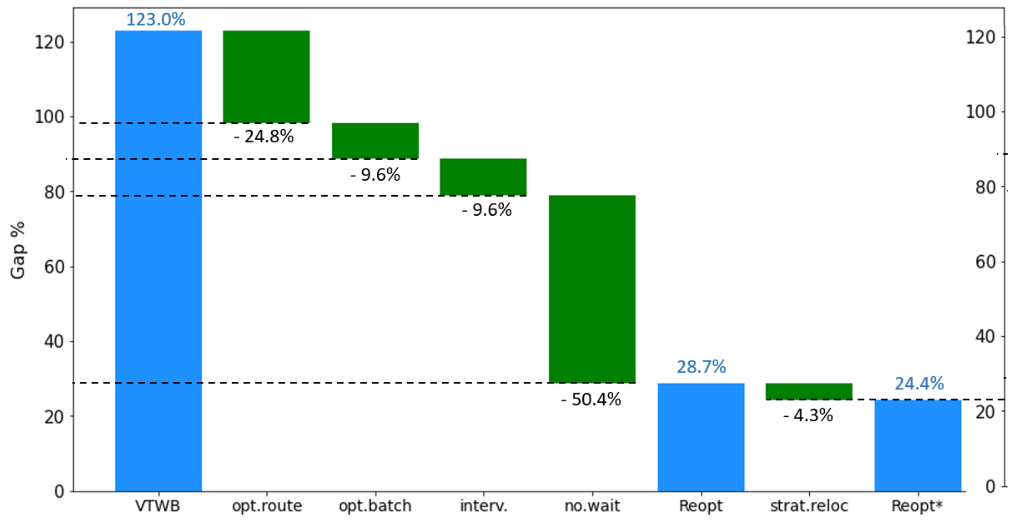
Starting with the well-known *Variable Time Window Batching (VTWB)* policy, which was extensively studied in the literature (Gil-Borrás et al., 2024; Van Nieuwenhuyse & De Koster, 2009), we progressively refine its algorithmic elements based on the insights from the previous sections (see Figure 3.9 and Table 3.15). VTWB is a non-interventionist policy that employs simple FIFO-batching, S-shape routing, and waits at the depot until at least $c_o = 2$ orders arrive (note that the value of $c_o = 2$ equals the optimal amount of waiting in the examined warehouse based on the queuing-theoretical formulas of Le-Duc and De Koster (2007) applied to our *Basis* setting). First, we replace the S-shape routing with *optimal routing* (cf. Section 3.6.2.2), then introduce *optimal batching* in place of FIFO batching (cf. Section 3.6.2.1). We further incorporate intervention, resulting in a myopic reoptimization policy with intervention (cf. Dauod & Won, 2022; Giannikas et al., 2017, and Chapter 2). Given our findings on the high opportunity cost of strategic waiting and its dependence on anticipatory batching (Section 3.6.2.3), we *eliminate waiting*.

For the *makespan* objective, we add a *batch selection rule*, prioritizing batches with the largest number of orders and, in case of ties, those offering the highest savings (cf. Section 3.6.2.4). No such rule is applied for the *turnover* objective, as reoptimization

Figure 3.9: Development of the overall average gap to the perfect anticipation result of online policies with algorithmic elements



(a) Makespan minimization



(b) Average turnover minimization

inherently selects batches to minimize additive order completion times. Finally, we allow the picker to move to the warehouse center during idle time, inspired by the concept of *strategic relocation* (cf. Section 3.6.2.4). The resulting policy is termed *Reopt**.

We evaluate the impact of each algorithmic element by calculating the *gap to the perfect anticipation result*. The gap of an algorithm ALG on instance I compares $z^{\text{ALG}}(I)$ (the objective value achieved by ALG) to $z^*(I)$ (the optimal CIOS objective value) and is defined as:

$$\text{gap}(\text{ALG}, I) = \frac{z^{\text{ALG}}(I) - z^*(I)}{z^*(I)}. \tag{3.20}$$

Observe that for *turnover* objective, we compare the gaps of ALG's average turnover time $z^{\text{turnover; ALG}}(I)$ to the *upper bound* of $z^*(I)$, computed using the heuristic DP approach from Section 3.4.

The introduced algorithmic elements led to significant improvements in the basic VTWB policy. On average, gaps were reduced from 16.3% to 3.4% for the *makespan* objective and from 123.0% to 24.4% for the *turnover* objective across all settings, see Figure 3.9.

While the positive effects of optimal routing and batching have been well-documented in the literature, the benefits of *no-wait*, *intervention*, and *strategic relocation* have largely gone unnoticed (cf. Pardo et al., 2024). Additionally, for the *makespan* objective, tailored batch selection offers moderate improvements, especially for higher order arrival rates and large orders, with a 0.1% average improvement across instances and up to 2% in specific cases.

Only Dauod and Won (2022) and Giannikas et al. (2017) applied intervention in their policies, although its positive effect is stable across a large range of policies. While in Figure 3.9 and Table 3.15 we already documented this effect for a *Reopt* policy with VTW-waiting for $c_o = 2$ orders, Table 3.16 extend the findings to *Reopt* with no waiting, and the FIFO policy, which performs optimal routing, FIFO-batching, and no waiting. For both policies, intervention *reduced all average* gaps with the *makespan* objective by impressive 1.7-4.0 percentage points, with the *turnover* objective the reduction was even 6.0-20.8 percentage points.

3.6.4 Discussion: Quantifying the potential of advanced anticipation techniques

The last column of Table 3.15 highlights the potential benefits of advanced anticipation, comparing the enhanced myopic reoptimization algorithm Reopt* with the perfect anticipation algorithm. Advanced anticipation could reduce the average *makespan* by 2.7–3.9%, with higher gains for larger order arrival rates and larger orders, where the potential for greater savings from batching is higher. For the *turnover* objective, relative gains from advanced anticipation are more substantial, ranging from 19.2% to 26.4%, particularly for small orders. However, notice the comparatively small denominator (average order turnover) for these improvements. Overall, as outlined in Section 3.6.3, simple adjustments can lead to significant improvements in basic myopic heuristics, even without the use of anticipation techniques.

Table 3.15: Development of average- (worst-) gap (%) to the perfect anticipation result of online policies with algorithmic elements

Obj.	Setting ($n^o = 15$)	VTWB	Effect of algorithmic elements						Reopt*
			opt.route	opt.batch	interv.	no.wait	batch.select	strat.reloc	
Makesp.	*LargeOrders_c2_r200*	19.9 (34.4)	-9.5 (-15.1)	-2.9 (-5.1)	-1.0 (+1.3)	-3.0 (-8.0)	-0.5 (-1.6)	0.0 (0.0)	3.0 (6.0)
	SmallOrders_c2_r200	10.9 (23.5)	-2.1 (-4.6)	-2.6 (-5.4)	-0.2 (-3.1)	-2.8 (-1.3)	+0.2 (-0.1)	-0.5 (-1.0)	2.7 (8.0)
	SmallOrders_c2_r250	12.4 (28.4)	-2.6 (-10.5)	-1.4 (-1.4)	-1.1 (+0.1)	-2.7 (-5.9)	0.0 (-1.9)	-0.7 (0.0)	3.9 (8.8)
	SmallOrders_c4_r250	22.1 (65.5)	-3.9 (-12.6)	-8.3 (-34.5)	-4.7 (-8.2)	-0.4 (+3.4)	0.0 (-1.2)	-0.9 (0.0)	3.9 (12.5)
Turnover	*LargeOrders_c2_r200*	113.2 (151.5)	-42.7 (-27.9)	-5.1 (-17.5)	-8.7 (-9.5)	-34.2 (-56.9)	– (–)	-3.2 (-10.6)	19.2 (29.1)
	SmallOrders_c2_r200	147.6 (278.4)	-21.5 (-10.3)	-3.6 (-21.2)	-8.3 (-4.5)	-79.9 (-187.6)	– (–)	-8.2 (-14.1)	26.0 (40.5)
	SmallOrders_c2_r250	107.9 (180.3)	-17.2 (-0.8)	-7.1 (-5.0)	-9.2 (+0.1)	-45.2 (-122.1)	– (–)	-3.2 (-4.1)	26.0 (48.8)
	SmallOrders_c4_r250	123.4 (181.8)	-17.8 (-26.0)	-22.5 (-10.6)	-12.5 (-4.9)	-42.2 (-85.9)	– (–)	-2.5 (-1.3)	26.4 (53.0)

Table 3.16: Effect of intervention on average- (worst-) gap (%) to the perfect anticipation result for non-interventionist policies

	Makespan objective				Turnover objective			
	Reopt		FIFO		Reopt		FIFO	
Setting ($n^o = 15$)	Gap N-interv.	interv.	Gap N-interv.	interv.	Gap N-interv.	interv.	Gap N-interv.	interv.
LargeOrders_c2_r200	6.3 (11.2)	-2.8 (-3.6)	7.5 (14.8)	-2.0 (-1.5)	37.7 (54.0)	-13.3 (-14.3)	48.3 (95.9)	-15.5 (-46.9)
SmallOrders_c2_r200	5.1 (15.2)	-2.1 (-6.2)	6.6 (17.2)	-1.7 (-1.7)	46.7 (75.7)	-12.5 (-21.1)	63.1 (93.3)	-10.5 (-5.3)
SmallOrders_c2_r250	8.0 (16.6)	-3.4 (-6.0)	8.8 (17.4)	-2.3 (-1.0)	47.3 (101.1)	-18.1 (-48.6)	61.6 (106.1)	-13.1 (-23.0)
SmallOrders_c4_r250	8.6 (20.2)	-3.8 (-6.4)	9.4 (25.9)	-4.0 (-5.6)	49.7 (93.2)	-20.8 (-38.3)	64.1 (97.1)	-6.4 (-5.0)
Overall	7.0 (20.2)	-3.0 (-6.5)	8.1 (25.9)	-2.5 (-5.6)	45.3 (101.1)	-16.6 (-46.5)	59.3 (106.1)	-6.0 (-14.0)

Note. Columns 2 & 6 (4 & 8) Gap N-interv.: Gap gaps of *Reopt* (FIFO) policy without intervention. *Columns 3 & 7 (5 & 9) interv.:* Effect on the gap of allowing intervention in *Reopt* (FIFO).

3.7 Conclusion

This paper investigates perfect-information policies for warehouse picking operations with dynamically arriving orders. The goal is to identify decision patterns that reduce order-picking costs and average order turnover, which can be integrated into practical warehousing policies. Additionally, the study uses perfect-information policies to assess the potential benefits of advanced anticipation techniques, such as contextual information and machine learning approaches.

To achieve this, we developed a customized dynamic programming algorithm (DP) for the *Order Batching, Sequencing, and picker Routing Problem with Release times (OBSRP-R)*, representing perfect-information operations in picker-to-parts warehouses. For the *makespan* objective, which focuses on the picker's time and associated wage costs, the DP is exact, making it the first exact solution approach for OBSRP-R in the literature. For the *turnover* objective, the DP serves as an efficient heuristic and becomes exact if no waiting times are involved. Commercial solvers were incapable of solving even small instances optimally, even with extensive computational resources for parallelization, highlighting the necessity of our algorithms.

Our findings confirm the importance of using optimal batching and routing strategies over simpler heuristic rules like first-come-first-serve (FIFO) or S-shape routing. More importantly, our analysis generates actionable insights on how to improve both the makespan and the turnover objectives *simultaneously*.

Insight 1. Implement intervention. Intervention involves dynamically adjusting picking plans while the picker is 'in the field', such as adding a new order to the cart or replacing an order whose items have not yet been picked with another. Although it requires investment in data and communication systems for real-time updates of the picker's instructions, these adjustments can yield operational benefits. Intervention fully utilizes incoming order information reducing costs and speeding up operations. Despite being highlighted by Giannikas et al. (2017) for its benefits, intervention has received little attention, notwithstanding its consistent positive impact across various heuristic approaches (see Section 3.6.3).

Insight 2. Eliminate waiting. The perceived importance of strategic waiting has been influenced by compelling examples like Figure 3.2 (Section 3.1). While brief periods of strategic waiting have shown some positive effects in FIFO policies (Bukchin et al., 2012; Le-Duc & De Koster, 2007; Van Nieuwenhuyse & De Koster, 2009), attempts to incorporate it into more advanced planning heuristics have largely failed (cf. Gil-Borrás et al., 2024; Henn, 2012). In fact, when intervention is allowed, strategic waiting becomes detrimental to both the makespan and the average order turnover. For the *turnover* objective, eliminating strategic waiting yields the most significant improvement, cutting the average optimality gap by 50% across all instances, with reductions as high as 188% in one case.

Our analysis in Section 3.6.2.3 explains these findings, showing that strategic waiting is effective only when it is brief, well-timed, and based on accurate order anticipation. This makes it challenging to integrate into non-anticipatory online policies. Ultimately, strategic waiting resembles an "all-in" gamble, with high opportunity costs if the waiting period is too long.

Insight 3. Relocate the picker to the 'gravity center' of anticipated future orders during idle time. In practice, space constraints and proximity to sorting and packaging areas often prevent the depot from being centrally located, unlike in idealized warehouse designs. Allowing the picker to move to the 'order gravity center' during idle time can help offset this limitation. Similar strategies have been observed in other dynamic routing problems in the literature (D. J. Bertsimas & van Ryzin, 1993).

The benefits of *no-wait*, *intervention*, and *strategic relocation* have largely gone unnoticed in the warehousing literature (cf. Pardo et al., 2024), despite their positive impact on both the *makespan* and *turnover* objectives.

Insight 4. Advanced anticipation offers limited leverage in picking operations – focus data science efforts on upstream planning. Significant improvements in warehouse picking can be achieved using classic optimization methods, with only modest gains left for advanced anticipation, as shown by the perfect-information benchmark. For example, classic optimization can improve the picker's time and wages by about 13 percentage points, leaving a smaller 3-4% improvement potential from advanced anticipation. Additionally, as discussed in Sections 3.6.2.3-3.6.2.4, the margin for error in anticipation is likely small in picking operations. Higher-impact decisions with more tolerance for anticipation errors are found in upstream areas like daily staffing, inventory levels, item repositioning, and zoning.

Perfect-information benchmarks offer useful insights on the limits and promises of advanced anticipation techniques. Future research is needed to investigate anticipation potential across further decisions in e-commerce warehousing. Future studies may improve the scalability of online algorithms, especially those that allow for time-consuming optimal batching and routing.

3.8 Supplemental material

3.8.1 Supplement: Mixed-integer linear programming formulations for OBSRP-R

In this section, we present *mixed-integer programming (MIP)* formulations for OBSRP-R with the makespan (Section 3.8.1.1) and turnover (Section 3.8.1.2) objective.

In addition to the previous notation summarized in Table 3.1, we introduce in the MILP formulations a $n^i \times n^o$ *adjacency matrix* $A = (a_{sj})_{s \in S, j \in [n^o]}$ for which each entry $a_{sj} \in A$ takes the value 1 if item s belongs to order o_j, and the value 0, else. Let K denote an upper bound for the number of batches in an optimal solution.

For both objectives, we use a three-index formulation similar to in T.-L. Chen et al. (2015) and Van Gils et al. (2019) for the order batching, sequencing, and picker routing problem without release times.

We use the following decision variables:

- For each batch index $k \in [K]$, and each pair of locations $i \in S \cup \{l_d\}, l \in S \cup \{l_d\}, i \neq l$, in the of set picking locations or the depot, the binary variables y_{kil} indicate if location l is visited within the kth batch, and if it is visited directly after location i in this batch (then, $y_{kil} = 1$), or not (then, $y_{kil} = 0$).
- For each batch index $k \in [K]$ and every $j \in [n^o]$, the binary variable x_{kj} indicates if order o_j is in the kth batch (then, $x_{kj} = 1$), or not (then, $x_{kj} = 0$).
- For each item $s \in S$, the continuous variable t_s represents the completion time $C(s)$ of item s.
- For each batch index $k \in [K + 1]$, the continuous variable $t^k_{l_d}$ represents the time the picker leaves the depot l_d to start the kth batch.

3.8.1.1 MIP formulation for the makespan objective

For the makespan objective, we can use a customized upper bound K for the number of batches, which is given by Proposition 11.

Proposition 11. *For every instance of OBSRP-R with the objective of minimizing the makespan,there exists an optimal solution that forms at most K batches, with*

$$K := \lfloor n^o \cdot \frac{2}{c+1} \rfloor \tag{3.21}$$

Proof. Consider a solution $\hat{\sigma}$ with $\pi^{\text{batches}}(\hat{\sigma}) = (\hat{B}_1, ..., \hat{B}_l, \hat{B}_{l+1}, ..., \hat{B}_f)$, such that for two *consecutive* batches $|\hat{B}_l| + |\hat{B}_{l+1}| \leq c$. Then $\hat{\sigma}$ is weakly dominated by the solution $\tilde{\sigma}$ that conserves the visiting sequence of items $\pi(\hat{\sigma})$, and the sequence of batches $\pi^{\text{batches}}(\hat{\sigma})$, except from replacing $\hat{B}_l, \hat{B}_{l+1}$ by a single batch $\tilde{B}$, with $\pi^{\tilde{B}} = (\pi^{\hat{B}_l}, \pi^{\hat{B}_{l+1}})$. This fact follows from the triangle inequality of the considered distance metric: transitioning directly from the last item of $\pi^{\hat{B}_l}$ to the first item of $\pi^{\hat{B}_{l+1}}$ is at least as fast as taking the detour through the depot. Thus, even with release times for the orders, at most the waiting time for one of the items in B_{l+1} may increase, which has no consequences on the solution's completion time, and the overall makespan of the solution can not deteriorate.

By definition, an optimal solution with the smallest number of batches $\tilde{K}$ cannot be weakly dominated by a solution with fewer batches, and from the above, we conclude that each pair of consecutive batches in such a solution may count together at least $c+1$ orders. In case $2n^o \mod (c+1) = 0$, every solution having this property has at most as many batches as a solution that partitions the n^o orders into pairs of batches with *exactly* $c+1$ orders. There are $\frac{n^o}{c+1}$ such pairs of batches which can be formed, thus $\frac{2 \cdot n^o}{c+1}$ individual batches. In the case where $2n^o \mod (c+1) > 0$, there must be even one pair of consecutive batches with more than $c+1$ orders, and the maximum amount of formed batches is K as in (3.21). □

We propose the following MILP formulation:

$$\text{Minimize} \max_{k \in [K+1]} \{t^k_{l_d}\} \tag{3.22}$$

s.t.

$$\sum_{k=1}^{K} \sum_{i \in S \cup \{l_d\}, i \neq l} y_{kis} = 1 \qquad \forall s \in S \tag{3.23}$$

$$\sum_{i \in S} y_{kil_d} \leq 1 \qquad \forall k \in [K] \tag{3.24}$$

$$\sum_{i \in S \cup \{l_d\}, i \neq s} y_{kis} \leq \sum_{i \in S \cup \{l_d\}, i \neq l} y_{ksi} \qquad \forall s \in S, \forall k \in [K] \tag{3.25}$$

$$\sum_{k=1}^{K} \sum_{s \in S} y_{ksl_d} = \sum_{k=1}^{K} \sum_{s \in S} y_{kl_ds} \tag{3.26}$$

$$\sum_{s \in S} y_{kl_ds} \leq \sum_{s \in S} y_{(k-1)l_ds} \qquad \forall k \in \{2, 3, ...K\} \tag{3.27}$$

$$x_{kj} \geq a_{sj} \sum_{i \in S \cup \{l_d\}} y_{kis} \qquad \forall j \in [n^o], \forall k \in [K], \forall s \in S \tag{3.28}$$

$$\sum_{k=1}^{K} x_{kj} \leq 1 \qquad \forall j \in [n^o] \tag{3.29}$$

$$\sum_{j=1}^{n^o} x_{kj} \leq c \qquad \forall k \in [K] \tag{3.30}$$

$$t_s \geq t_i - M(1 - \sum_{k=1}^{K} y_{kis}) + \frac{1}{v} \cdot d(i,s) + t^p \qquad \forall s \in S, \forall i \in S, s \neq i \tag{3.31}$$

$$t_s \geq t^k_{l_d} - M(1 - y_{kl_ds}) + \frac{1}{v} \cdot d(l_d, s) + t^p \qquad \forall s \in S, \forall k \in [K] \tag{3.32}$$

$$t^k_{l_d} \geq t_s - M(1 - y_{(k-1)sl_d}) + \frac{1}{v} \cdot d(s, l_d) \qquad \forall s \in S, \forall k \in \{2, 3, ..., K+1\} \tag{3.33}$$

$$t_s \geq r_j \cdot a_{sj} + t^p \qquad \forall s \in S, \forall j \in [n^o] \tag{3.34}$$

$$y_{kil} \in \{0, 1\} \qquad \forall k \in [K], \forall i, l \in S \cup \{l_d\}, i \neq l \tag{3.35}$$

$$x_{kj} \in \{0, 1\} \qquad \forall k \in [K], \forall j \in [n^o] \tag{3.36}$$

$$t_i \geq 0 \qquad \forall i \in S \tag{3.37}$$

$$t^k_{l_d} \geq 0 \qquad \forall k \in [K+1] \tag{3.38}$$

The objective function (3.22) minimizes the total completion time. The constraints (3.23) - (3.26) are flow constraints, which also ensure that every picking location is visited. Constraints (3.27) are symmetry-breaking constraints. Constraints (3.28) and (3.29) assign orders to batches, such that all items that belong to the same order are placed in the same batch. Constraints (3.30) define the batch capacities. Constraints (3.31)-(3.33) are *Miller-Trucker-Zemlin (MTZ)* subtour elimination constraints and at the same time define the completion time of the items. The following large parameter M is used within these constraints:

$$M := r_{n^o} + \frac{1}{v} \cdot (2 \cdot L + (a+1) \cdot W) \cdot \lceil \frac{n^o}{c} \rceil + n^i \cdot t^p + \frac{1}{v} \cdot (L + W) \tag{3.39}$$

as it is an upper bound for the expressions $t_i + \frac{1}{v}d(i, s) + t^p$ and $t^k_{l_d} + \frac{1}{v}d(l_d, s) + t^p$ for all $s \in S \cup \{l_d\}$ and thus guarantees that the right-hand side of the constraints (3.31)-(3.33) are non-positive in case M is multiplied by 1. To see this, note that the problem turns into a static problem after the arrival of the last order r_{n^o}, whose optimal completion time is bounded from above by the time to pick $\lceil \frac{n^o}{c} \rceil$ arbitrarily formed batches by traversing the warehouse completely in an S-shape motion of at most $\frac{1}{v} \cdot (2 \cdot L + (a+1) \cdot W)$ time, see also Lemma 1 in Chapter 2. Thus the first two summands of M form an upper bound for t_i and $t^k_{l_d}$ in any optimal solution, and similarly, the last summand of M bounds from above the traversal time $\frac{1}{v}d(i, s)$ for all $i, s \in S \cup \{l_d\}$. Constraints (3.34) make sure that the release times are respected and (3.35)-(3.38) define the nature of the variables.

3.8.1.2 MIP formulation for the turnover objective

In the case of the turnover objective, one additional group of variables is needed:

- For each order $o_j, j \in [n^o]$, the continuous variable t^o_j represents the completion time $C(o_j)$ of order o_j.

Furthermore, the upper bound on the number of batches K from Proposition 11 does not hold in this case, thus, $K := n^o$ is used as an upper bound.

The objective in this case writes as follows:

$$\text{Minimize} \sum_{j=1}^{n^o} \frac{1}{n^o} t^o_j - \sum_{j=1}^{n^o} \frac{1}{n^o} r_j, \tag{3.40}$$

where we remember that $\sum\limits_{j=1}^{n^o} \frac{1}{n^o} r_j$ is a constant.

In addition to all constraints (3.23)-(3.38) from the previous section, the following must hold to define the completion times of the orders:

$$t_j^o \geq t_{l_d}^{k+1} - N(1 - x_{kj}) \qquad \forall j \in [n^o], \forall k \in [K+1] \tag{3.41}$$

$$t_j^o \geq 0 \qquad \forall j \in [n^o] \tag{3.42}$$

Similarly to (3.39) the large parameter N receives the value

$$N := r_{n^o} + \frac{1}{v} \cdot (2 \cdot L + (a+1) \cdot W) \cdot \lceil \frac{n^o}{c} \rceil + n^i \cdot t^p. \tag{3.43}$$

3.8.2 Supplement: Proof of dominance rules

In this section, we consider an arbitrary instance I for OBSRP-R and show that the dominance rule claimed in Section 3.4.3 are valid for both objective functions - makespan and turnover minimization.

3.8.2.1 Proof of Proposition 8

We use Lemma 8 to define a set D_1 of feasible solutions for OBSRP-R instance I that are weakly dominated by other feasible solutions. In the second step, Lemma 9 states that every path from the initial state to the terminal state in the constructed state graph - refered to as a *complete path* in the following, which involves a transition $x_k, k \in [n^i - 1]$, from a label of state Θ_k with time value Ω^{time} as described by Proposition 8, is associated to a solution in set D_1.

Lemma 8. *Consider any feasible solution $\hat{\sigma}$ with a visiting sequence of picking locations $\hat{\pi}$ and the respective sequence of batches $\hat{\pi}^{\text{batches}}$ such that some item $\hat{\pi}[i+1]$ has a late release date:*

$$r(\hat{\pi}[i+1]) \geq C(\hat{\pi}[i]) + \frac{1}{v} \cdot (d(\hat{\pi}[i], s^\dagger) + d(s^\dagger, \hat{\pi}[i+1])) + t^p \tag{3.44}$$

for some fixed $s^\dagger \in o_{\hat{j}}$, such that $s^\dagger = \hat{\pi}[i+k]$ for some $k > 1$, i.e. $s^\dagger$ is picked after $\hat{\pi}[i+1]$ in $\hat{\sigma}$, and, such that the order $o_{\hat{j}}$ is already commenced at the picking of item $\hat{\pi}[i+1]$, i.e. some item of $o_{\hat{j}}$ is picked before $\hat{\pi}[i+1]$.

Let decompose the visiting sequence of $\hat{\sigma}$ as follows: $\hat{\pi} = (\hat{\pi}^I, \hat{\pi}[i+1], \hat{\pi}^{II})$, i.e. $\hat{\pi}^I$ and $\hat{\pi}^{II}$ denote the subsequences before and after the visit of location $\hat{\pi}[i+1]$, respectively.

Then the following solution $\tilde{\sigma}$ weakly dominates solution $\hat{\sigma}$:

- $\pi^{\text{batches}}(\tilde{\sigma}) = \hat{\pi}^{\text{batches}}$
- $\pi(\tilde{\sigma}) = (\hat{\pi}^I, s^\dagger, \hat{\pi}[i+1], \hat{\pi}^{II} \setminus s^\dagger)$, *where $\hat{\pi}^{II} \setminus s^\dagger$ is the sequence $\hat{\pi}^{II}$ after removing item $s^\dagger$.*
  In other words, solution $\tilde{\sigma}$ collects item $s^\dagger$ immediately before item $\hat{\pi}[i+1]$.

Proof. By construction, $\tilde{\sigma}$ is a feasible solution. What remains to show, is that $z^{obj}(\tilde{\sigma}) \leq z^{obj}(\hat{\sigma})$, where $obj \in \{\text{makesp, turnover}\}$, depending on the objective.

Let compute the *schedule* of both solutions $\tilde{\sigma}$ and $\hat{\sigma}$ as described in (3.1)-(3.2). Since some items of order $o_{\hat{j}}$ are picked before item $\hat{\pi}[i+1]$ and since $s^\dagger$ also belongs to order $o_{\hat{j}}$:

$$r(s^\dagger) \leq C(\hat{\pi}[i]) \tag{3.45}$$

In both $\tilde{\sigma}$ and $\hat{\sigma}$, items $\hat{\pi}[i], \hat{\pi}[i+1]$ and $s^{\dagger}$ are in the same batch $B(\hat{\sigma}, o_{\hat{j}})$ and the visiting sequences of the first i items are the same in $\tilde{\sigma}$ and $\hat{\sigma}$. Thus:

$$C(\hat{\pi}[l], \tilde{\sigma}) = C(\hat{\pi}[l], \hat{\sigma}) \text{ for all } l \leq i \tag{3.46}$$

and

$$C(o_j, \tilde{\sigma}) = C(o_j, \hat{\sigma}) \text{ for all } o_j \text{scheduled in batches prior to } B(\hat{\sigma}, o_{\hat{j}}) \text{ in } \hat{\pi}^{\text{batches}} \tag{3.47}$$

By the triangle inequality of the distance metric, we furthermore have: $C(s^{\dagger}, \tilde{\sigma}) \leq C(s^{\dagger}, \hat{\sigma})$.

Observe that in solution $\tilde{\sigma}$, item $\hat{\pi}[i+1]$ is picked immediately after visiting location $s^{\dagger}$. Furthermore, By applying (3.1)-(3.2), the completion time of item $\hat{\pi}[i+1]$ in solution $\tilde{\sigma}$ is:

$$\begin{aligned}
C(\hat{\pi}[i+1], \tilde{\sigma}) =& \max\{C(s^{\dagger}) + \frac{1}{v} d(s^{\dagger}, \hat{\pi}[i+1]), r(\hat{\pi}[i+1])\} + t^p && (3.48)\\
=& \max\{\max\{C(\hat{\pi}[i]) + \frac{1}{v} d(\hat{\pi}[i], s^{\dagger}), r(s^{\dagger})\} + t^p \\
&+ \frac{1}{v} d(s^{\dagger}, \hat{\pi}[i+1]), r(\hat{\pi}[i+1])\} + t^p && (3.49)\\
=& \max\{C(\hat{\pi}[i]) + \frac{1}{v} d(\hat{\pi}[i], s^{\dagger}) \\
&+ t^p + \frac{1}{v} d(s^{\dagger}, \hat{\pi}[i+1]), r(\hat{\pi}[i+1])\} + t^p && (3.50)\\
\leq& r(\hat{\pi}[i+1]) + t^p, && (3.51)
\end{aligned}$$

where the equality (3.49) follows from (3.45) and the subsequent inequality (3.50) follows from (3.44).

Similarly, by applying (3.1) and (3.2), the completion time of item $\hat{\pi}[i+1]$ in solution $\hat{\sigma}$ cannot be smaller, since:

$$\begin{aligned}
C(\hat{\pi}[i+1], \hat{\sigma}) &= \max\{C(\hat{\pi}[i]) + \frac{1}{v} \cdot d(\hat{\pi}[i], \hat{\pi}[i+1]), r(\hat{\pi}[i+1])\} + t^p \\
&\geq r(\hat{\pi}[i+1]) + t^p && (3.52)
\end{aligned}$$

In other words:

$$C(\hat{\pi}[i+1], \tilde{\sigma}) \leq C(\hat{\pi}[i+1], \hat{\sigma}) \tag{3.53}$$

After applying (3.2) to compute the completion times of the remaining items and observing that the distances are metric and that the sequences of the visiting locations after picking $\hat{\pi}[i+1]$ coincide in $\tilde{\sigma}$ and $\hat{\sigma}$, but $\tilde{\sigma}$ excludes the location of the already picked item $s^{\dagger}$, we receive that $C(\tilde{\pi}[l], \tilde{\sigma}) \leq C(\tilde{\pi}[l], \hat{\sigma})$ for all $l \geq i+1$, and thus by (3.4) that $C(o_j, \tilde{\sigma}) \leq C(o_j, \hat{\sigma})$ for all orders o_j in batch $B(\hat{\sigma}, o_{\hat{j}})$ and batches scheduled after this batch. By definitions (3.3) and (3.5) we receive that $z^{\text{makesp}}(\tilde{\sigma}) \leq z^{\text{makesp}}(\hat{\sigma})$ or $z^{\text{turnover}}(\tilde{\sigma}) \leq z^{\text{turnover}}(\hat{\sigma})$, depending on the objective.

□

We denote the set of the weakly dominated feasible solutions $\hat{\sigma}$ described by Lemma 8 as D_1.

In Proposition 3.4.1, we have established the one-to-one correspondence between feasible solutions of an instance I and the complete paths the corresponding state graph.

Furthermore, every label for state Θ_k of the graph with a specific time value $\Omega^{\text{time}}(\Theta_k)$ represents exactly one sub-path to reach Θ_k from the initial state. Lemma 9 uses this relation to complete the proof.

Lemma 9. *Consider a feasible transition $x_k \in X(\Theta_k)$ from a state $\Theta_k = (s_k, m^o, S^{batch}, O^{pend})$ with $|S^{batch}| \geq 1$, and an associated time value $\Omega^{time}(\Theta_k)$, that dictates a next picking location $s_{k+1} \in o_j, o_j \in O^{pend}$ with the following property*

$$r_j \geq \Omega^{time}(\Theta_k) + \frac{1}{v} \cdot (d(s_k, s^\dagger) + d(s^\dagger, s_{k+1})) + t^p \text{ for } s^\dagger \in S^{batch}. \tag{3.54}$$

Then, any complete path that involves x_k through the given label belongs to the class of dominated solutions D_1 described by Lemma 8.

Proof. First, note that the items s_{k+1} an $s^\dagger$ of Lemma 9 correspond to items $\hat{\pi}[i+1]$ an $s^\dagger$ in Lemma 8, respectively, and that $r(s_{k+1}) = r_j$ given that $s_{k+1} \in o_j$. Let P^{sub} be the path of transitions from the initial state to state Θ_k that is associated to the value $\Omega^{\text{time}}(\Theta_k)$. Consider any complete path that uses P^{sub} as a sub-path, followed by x_k, denote by $\hat{\sigma}$ the corresponding feasible solution as by Proposition 3.4.1. The completion time of s_k in $\hat{\sigma}$ is $C(s_k, \hat{\sigma}) = \Omega(\Theta_k)$ by definition. Because of (3.54), property (3.44) holds for $\hat{\sigma}$ and therefore, $\hat{\sigma}$ belongs to the set of weakly dominated solutions D_1.

□

3.8.2.2 Proof of Proposition 9

We proceed along similar lines as in Section 3.8.2.1. For an arbitrary instance I of OBSRP-R, we use Lemma 10 to define the set D_2 of feasible solutions for I that are weakly dominated by another feasible solution. In the second step, Lemma 11 associates every complete path that uses $x_k, k \in [n^i - 1]$ through a label of state Θ_k with time value $\Omega^{\text{time}}(\Theta_k)$ as described by Proposition 9, to a solution in set D_2.

Lemma 10. *Consider any feasible solution $\hat{\sigma}$ with a visiting sequence of picking locations $\hat{\pi}$ and the respective sequence of batches $\hat{\pi}^{batches}$ such that:*

- *Some item $\hat{\pi}[i+1]$ that belongs to some batch $\hat{B}_l \in \hat{\pi}^{batches}$ has a late release date:*

$$r(\hat{\pi}[i+1]) \geq C(\hat{\pi}[i]) + \frac{1}{v} \cdot (d(\hat{\pi}[i], l_d) + d(l_d, \hat{\pi}[i+1])) \tag{3.55}$$

- *This batch $\hat{B}_l = \hat{B}_l^I \cup \hat{B}_l^{II}$ can be partitioned into two sets of orders $\hat{B}_l^I$ and $\hat{B}_l^{II}$ such that*
  - *$\hat{B}_l^I$ and $\hat{B}_l^{II}$ are picked subsequently in $\hat{\pi}^{\hat{B}_l} = (\hat{\pi}^{\hat{B}_l, I}, \hat{\pi}^{\hat{B}_l, II})$,*
  - *item $\hat{\pi}[i+1]$ is the first item in the sequence $\hat{\pi}^{\hat{B}_l, II}$.*

Then the following solution $\tilde{\sigma}$ weakly dominates solution $\hat{\sigma}$:

- *Orders of $\hat{B}_l^I$ and $\hat{B}_l^{II}$ are picked in separate batches, whereas the remaining batches and their sequence remain the same: $\pi^{batches}(\tilde{\sigma}) = (\hat{B}_1, \ldots, \hat{B}_{l-1}, \hat{B}_l^I, \hat{B}_l^{II}, \hat{B}_{l+1}, \ldots)$*
- *$\pi(\tilde{\sigma}) = \hat{\pi}$, i.e. the sequences of picking locations coincide.*

Proof. The proof proceeds along the same lines as the proof of Lemma 8. By construction, $\tilde{\sigma}$ is a feasible solution. What remains to show, is that $z^{obj}(\tilde{\sigma}) \leq z^{obj}(\hat{\sigma})$, where $obj \in \{\text{makesp, turnover}\}$, depending on the objective.

We apply (3.2) and notice that $C(\hat{\pi}[i+1],\tilde{\sigma}) \leq C(\hat{\pi}[i+1],\hat{\sigma}) = r(\hat{\pi}[i+1]) + t^p$. By the recursive nature of equations (3.1)-(3.2), all following items in $\hat{\pi}$ conserve their completion time and by (3.4) order completion times are identical in $\tilde{\sigma}$ and $\hat{\sigma}$. The conclusion of the proof follows by the definition of the objective functions (3.3) and (3.5). □

Let denote the set of weakly dominated solutions $\hat{\sigma}$ introduced in Lemma 10 as D_2. Lemma 11 associates each path through the state graph that involves a transition x_k described by Proposition 9 to a weakly dominated solution and thereby closes the proof.

Lemma 11. *Consider a feasible transition $x_k \in X(\Theta_k)$ from a state $\Theta_k = (s_k, m^o, \{\}, O^{pend})$ with $m^o \geq 1$, that dictates a next picking location $s_{k+1} \in o_j, o_j \in O^{pend}$ with the following property*

$$r_j \geq \Omega^{time}(\Theta_k) + \frac{1}{v}(d(s_k, l_d) + d(l_d, s_{k+1})) \tag{3.56}$$

Then, any complete path that involves x_k through the given label belongs to the class of dominated solutions D_2 described by Lemma 8.

Proof. First, note that the item s_{k+1} of Lemma 11 corresponds to item $\hat{\pi}[i+1]$ in Lemma 10, and that $r(s_{k+1}) = r_j$ since $s_{k+1} \in o_j$. Given that $S^{\text{batch}} = \{\}$ and $m^o \geq 1$ in state Θ_k, item s_{k+1} belongs to batch $\hat{B}_l$ that can be decomposed as follows: $\hat{B}_l = \hat{B}_l^I \cup \hat{B}_l^{II}$, where $\hat{B}_l^I$ is a set of orders for which all items have already been picked by the time state Θ_k is reached, and $\hat{B}_l^{II}$ is a set of completely unprocessed orders at state Θ_k, one of which includes item s_{k+1}.

Using the same arguments as in the proof of Lemma 9, each complete path of the state graph that involves transition $x_k \in X(\Theta_k)$ from the label of state Θ_k with time value $\Omega^{\text{time}}(\Theta_k)$ completes the picking of item s_k at time $C(s_k, \hat{\sigma}) = \Omega(\Theta_k)$, thus property (3.56) translates to property (3.55), and the path is associated with a weakly dominated solution in the set D_2. □

3.8.2.3 Proof of Proposition 10

Again, the proof of Proposition 10 follows the same lines as the one presented in Section 3.8.2.1. Consider an instance I for OBSRP-R. Lemma 12 identifies a set D_3 of feasible solutions, that are weakly dominated by another feasible solution.

Lemma 12. *Consider any feasible solution $\hat{\sigma}$ with a visiting sequence of picking locations $\hat{\pi} = (\hat{\pi}^I, \hat{\pi}^{II})$ and the respective sequence of batches $\hat{\pi}^{\text{batches}}, |\hat{\pi}^{\text{batches}}| = \hat{f} \in \mathbb{N}$ such that:*

- *The second subsequence $\hat{\pi}^{II}$ contains the batches $\hat{B}_l \cup \ldots \cup \hat{B}_{\hat{f}}$. for some $l \in [\hat{f}]$. Let denote the first location of the second subsequence as $\hat{\pi}[i+1]$.*
- *There exists an order $o_{\underline{j}} \in \hat{B}_l \cup \ldots \cup \hat{B}_{\hat{f}}$, whose items are picked in the second subsequence $\hat{\pi}^{II}$ in $\hat{\sigma}$, which satisfies the following relation:*

$$r(\hat{\pi}[i+1]) \geq \max\{C^i, r_{\underline{j}}\} + \chi(o_{\underline{j}}) + \frac{1}{v} \cdot d(l_d, \hat{\pi}[i+1]), \tag{3.57}$$

where $C^i := C(\hat{\pi}[i], \hat{\sigma}) + \frac{1}{v} \cdot d(\hat{\pi}[i], l_d)$ if $i \neq 0$ and $C^i := 0$ else.

Then the following solution $\tilde{\sigma}$ is feasible and weakly dominates $\hat{\sigma}$:

- *$\pi(\tilde{\sigma}) = (\hat{\pi}^I, \pi^{\underline{j}}, \pi^{-\underline{j}})$ with $\pi^{\underline{j}}$ representing a visiting sequence of the picking locations in order $o_{\underline{j}}$ of travel distance $\chi(o_{\underline{j}})$, and $\pi^{-\underline{j}}|\hat{\pi}^{II}$ representing the subsequence of $\hat{\pi}^{II}$ after removing all the items of order $o_{\underline{j}}$. In other words, solution $\tilde{\sigma}$ picks the items of order $o_{\underline{j}}$ directly after $\hat{\pi}^I$ and then resumes visiting the remaining locations in the same order as in $\hat{\sigma}$.*
- *$\pi^{batches}(\tilde{\sigma}) = (\hat{B}_1, \dots, \hat{B}_{l-1}, \{o_{\underline{j}}\}, \hat{B}_l \setminus \{o_{\underline{j}}\}, \dots \hat{B}_{\hat{f}} \setminus \{o_{\underline{j}}\})$, i.e., in solution $\tilde{\sigma}$, order $o_{\underline{j}}$ is collected as a separate batch directly after $\hat{\pi}^I$.*

Proof. By construction, $\tilde{\sigma}$ is a feasible solution, for instance, $\pi^{\text{batches}}(\tilde{\sigma})$ represents a mutually disjoint partition of the orders into batches and each batch contains at most c orders. What remains to show, is that $z^{obj}(\tilde{\sigma}) \leq z^{obj}(\hat{\sigma})$, where $obj \in \{$makesp, turnover$\}$, depending on the objective.

Let compute the schedule of both solutions $\tilde{\sigma}$ and $\hat{\sigma}$ as described in (3.1)-(3.2). Observe that the first i items in $\tilde{\sigma}$ and $\hat{\sigma}$ are the same, thus all items and orders in $\hat{\pi}^I$ have the same completion times in both solutions. By definition, $C(o_{\underline{j}}, \tilde{\sigma}) \leq C(o_{\underline{j}}, \hat{\sigma})$. In solution $\tilde{\sigma}$, item $\hat{\pi}[i+1]$ is picked after all the items of order $o_{\underline{j}}$ are collected in some sequence $\pi^{\underline{j}}$. Following the assumptions, and by applying (3.1)-(3.2) the completion time of item $\hat{\pi}[i+1]$ in solution $\tilde{\sigma}$ is:

$$C(\hat{\pi}[i+1], \tilde{\sigma}) \leq \max\{\max\{C^i, r_{\underline{j}}\} + \chi(o_{\underline{j}}) + \frac{1}{v} d(l_d, \hat{\pi}[i+1]); r(\hat{\pi}[i+1])\} + t^p \tag{3.58}$$

$$= r(\hat{\pi}[i+1]) + t^p, \tag{3.59}$$

The completion time of item $\hat{\pi}[i+1])$ in solution $\hat{\sigma}$ cannot be smaller by (3.1)-(3.2),in other words:

$$C(\hat{\pi}[i+1]), \tilde{\sigma}) \leq C(\hat{\pi}[i+1]), \hat{\sigma}) \tag{3.60}$$

After applying (3.2) to compute the completion times of the remaining items and observing that the distances are metric and that the sequences of the visiting locations after picking $\hat{\pi}[i+1]$ coincide in $\tilde{\sigma}$ and $\hat{\sigma}$, but $\tilde{\sigma}$ excludes the locations of the items from the already picked order $o_{\underline{j}}$, we receive that all $C(s, \tilde{\sigma}) \leq C(s, \hat{\sigma})$ and $C(o_j, \tilde{\sigma}) \leq C(o_j, \hat{\sigma})$, for all items s and orders o_j completed after $\hat{\pi}[i+1]$ in $\tilde{\sigma}$. By definition of the objective functions, (3.3) and (3.5), this completes the proof. □

Let denote by D_3 the set of weakly dominated solutions $\hat{\sigma}$ described by Lemma 12. Again, the following Lemma 13 uses the one-to-one correspondence between feasible solutions for an instance and complete paths in the corresponding state graph (see Proposition 3.4.1), to associate any transition x_k from a label of state Θ_k with time value Ω^{time} described by Proposition 10 to a dominated solution in D_3.

Lemma 13. *Consider a feasible transition $x_k \in X(\Theta_k)$ from a state $\Theta_k = (s_k, 0, \{\}, O^{pend})$ and time value Ω^{time} that dictates a next picking location $s_{k+1} \in o_j, o_j \in O^{pend}$ with the following property*

$$r_j \geq \max\{\Omega^{time}(\Theta_k); r_{\underline{j}}\} + \chi(o_{\underline{j}}) + \frac{1}{v} \cdot d(l_d, s_{k+1}) \tag{3.61}$$

for some $o_{\underline{j}} \in O^{pend}$. Then, any closed path in the state graph that involves this transition belongs to the set of dominated solutions D_3 described by Lemma 12.

Proof. Note that the item s_{k+1} of Lemma 13 corresponds to item $\hat{\pi}[i+1]$ in Lemma 12. Since $m^o = 0$ in $Theta_k$, a new batch is initiated by the pick of s_{k+1}. Using the same arguments as in the proof of Lemma 9, each complete path that involves transition $x_k \in X(\Theta_k)$ through the label of Θ_k with time value $\Omega^{\text{time}}(\Theta_k)$ completes the picking of item s_k at time $\Omega^{\text{time}}(\Theta_k) = C^i - \frac{1}{v} \cdot d(s_k, l_d)$. Therefore, property (3.61) translates to property (3.57), and thus the path is associated with a weakly dominated solution in the class D_3. □

3.8.3 Supplement: Proof of analytical properties

3.8.3.1 Proof of Lemma 6

The lemma follows directly from formulas (3.1) and (3.2). Specifically, (3.2) can be reformulated as follows:

$$C^*(\pi^*[i+1]) = C^*(\pi^*[i]) + walk(\pi^*[i], \pi^*[i+1]) + w(\pi^*[i+1]) + t^p \quad \forall i \in [n^i - 1] \tag{3.62}$$

where $walk(\pi^*[i], \pi^*[i+1])$ is the picker's walking time between picking locations $\pi^*[i]$ and $\pi^*[i+1]$:

$$walk(\pi^*[i], \pi^*[i+1]) = \begin{cases} \frac{1}{v} \cdot d(\pi^*[i], \pi^*[i+1]) \\ \text{if } \pi^*[i], \pi^*[i+1] \text{ belong to the same batch} \\ \frac{1}{v} \cdot d(\pi^*[i], l_d) + \frac{1}{v} \cdot d(l_d, \pi^*[i+1]) \\ \text{if } \pi^*[i], \pi^*[i+1] \text{ belong to distinct batches} \end{cases} \tag{3.63}$$

and $w(\pi^*[i+1])$ is the waiting- or idle time at picking location $\pi^*[i+1]$:

$$w(\pi^*[i+1]) = \max\{0, r(\pi^*[i+1]) - C(\pi^*[i]) - walk(\pi^*[i], \pi^*[i+1])\}. \tag{3.64}$$

Recursively, from (3.62) the following formula follows for the completion time of the last picking item in both policies:

$$C^*(\pi^*[n^i]) = C^*(\pi^*[1]) + \sum_{i=1}^{n^i-1} w(\pi^*[i+1]) + \sum_{i=1}^{n^i-1} walk(\pi^*[i], \pi^*[i+1]) + (n^i - 1) \cdot t^p \tag{3.65}$$

$$= \frac{1}{v} \cdot d(l_d, \pi^*[1]) + w(\pi^*[1]) + t^p + \sum_{i=1}^{n^i-1} w(\pi^*[i+1]) \tag{3.66}$$

$$+ \sum_{i=1}^{n^i-1} walk(\pi^*[i], \pi^*[i+1]) + (n^i - 1) \cdot t^p \tag{3.67}$$

$$= w^*(I) + \frac{1}{v} \cdot d(l_d, \pi^*[1]) + t^p + \sum_{i=1}^{n^i-1} walk(\pi^*[i], \pi^*[i+1]) + (n^i - 1) \cdot t^p \tag{3.68}$$

$$= \tilde{C}^*(\pi^*[1]) + \sum_{i=1}^{n^i-1} walk(\pi^*[i], \pi^*[i+1]) + (n^i - 1) \cdot t^p \tag{3.69}$$

$$= \tilde{C}^*(\pi^*[n^i]) \tag{3.70}$$

This closes the proof, as the makespan of the alternate policy with waiting is $\tilde{z}^* = \tilde{C}^*(\pi^*[n^i]) + d(l_d, \pi^*[n^i])$ and equivalently, the makespan of the departing CIOS was $z^* = C^*(\pi^*[n^i]) + d(l_d, \pi^*[n^i])$. □

3.8.3.2 Proof of Lemma 7

For simplicity, let consider a picking time of $t^p = 0$. The proof for non-zero picking times follows the same logic. Let $s = (s_1, s_2, ..., s_l)$ be the visiting sequence of picking locations *and the depot*, of CIOS and ALG, starting with the first picking location. Let $done^{\mathsf{CIOS}}(s_k)$ and $done^{\mathsf{ALG}}(s_k)$ denote the time when CIOS and ALG, respectively, *finish* their task (visit for the depot and picking for a picking location) at location s_k, $k \in [l]$. Let $dep^{\mathsf{ALG}}(s_k)$ be the time the picker in ALG *departs* from location $s_k, k \in [l]$. Recall that $r(s_k)$ for $s_k \in S$ denotes the arrival time of picking location s_k. We extend this notation by writing $r(s_k) = 0$ if $s_k = l_d$.

Induction over $k \in [l]$.

Initialization; $k = 1$. For the first picking location s_1, we have:

$$done^{\mathsf{ALG}}(s_1) - done^{\mathsf{CIOS}}(s_1) \leq r(s_1) + d(l_d, s_1) - r(s_1) \leq \max_{i,j \in S \cup \{l_d\}} \{d(i,j)\} \tag{3.71}$$

Inductive hypothesis. We suppose that for fixed $k \in [l]$

$$done^{\mathsf{ALG}}(s_k) - done^{\mathsf{CIOS}}(s_k) \leq \max_{i,j \in S \cup \{l_d\}} \{d(i,j)\} \tag{3.72}$$

Inductive step, $k \to k+1$.

We distinguish two cases.

Case 1: $dep^{\mathsf{ALG}}(s_k) = done^{\mathsf{ALG}}(s_k)$, i.e. the picker in ALG departs immediately towards s_{k+1} after she reaches and terminates at location s_k, since $r(s_{k+1}) \leq done^{\mathsf{ALG}}(s_k)$. Then:

$$\begin{aligned}
& done^{\mathsf{ALG}}(s_{k+1}) - done^{\mathsf{CIOS}}(s_{k+1}) \\
= \; & done^{\mathsf{ALG}}(s_k) + d(s_k, s_{k+1}) - \max\{r(s_{k+1}); done^{\mathsf{CIOS}}(s_k) + d(s_k, s_{k+1})\} && (3.73) \\
\leq \; & done^{\mathsf{ALG}}(s_k) + d(s_k, s_{k+1}) - done^{\mathsf{CIOS}}(s_k) - d(s_k, s_{k+1}) && (3.74) \\
\leq \; & \max_{i,j \in S \cup \{l_d\}} \{d(i,j)\} && (3.75)
\end{aligned}$$

by the inductive hypothesis.

Case 2: $dep^{\mathsf{ALG}}(s_k) = r(s_{k+1})$, i.e. the picker in ALG must wait for the arrival of s_{k+1} after she terminates at location s_k. Then:

$$\begin{aligned}
& done^{\mathsf{ALG}}(s_{k+1}) - done^{\mathsf{CIOS}}(s_{k+1}) \\
= \; & r(s_{k+1}) + d(s_k, s_{k+1}) - \max\{r(s_{k+1}); done^{\mathsf{CIOS}}(s_k) + d(s_k, s_{k+1})\} && (3.76) \\
\leq \; & r(s_{k+1}) + d(s_k, s_{k+1}) - r(s_{k+1}) && (3.77) \\
\leq \; & \max_{i,j \in S \cup \{l_d\}} \{d(i,j)\} && (3.78)
\end{aligned}$$

□

Chapter 4

On delivery policies for a truck-and-drone tandem in disaster relief

Alena Otto, Bruce Golden, Catherine Lorenz, Yuchen Luo, Erwin Pesch, Luis Aurelio Rocha

Abstract. This paper introduces the traveling salesman problem with a truck and a drone under incomplete information (TSP-DI). TSP-DI is motivated by the deliveries of emergency supplies under unknown road conditions in the immediate aftermath of a disastrous event. The urgency may force the immediate dispatch of relief vehicles, such that road damages blocking the truck's planned route are detected 'on-the-fly'. The relief transport must schedule deliveries anticipating possible unplanned truck detours, enforce (planned) drone detours for early checking of key road segments, and consider the dynamic nature of road condition information. In this paper, we perform a competitive analysis of a widely used delivery policy for TSP-DI in practice – the online reoptimization policy (Reopt) – and compare it to several alternative delivery strategies. Competitive analysis examines the worst-case performance of the strategies and is particularly important in the context of disaster relief, where worst-case outcomes must be avoided. Our analysis shows that *Reopt* is dominated by alternative delivery policies in terms of the competitive ratio even at a medium level of damage on the road. It also underscores the importance of surveillance detours performed by the drone, even if the surveillance delays the start of the deliveries.

4.1 Introduction

Driven by climate change, the destructive force of weather-related disasters has intensified, and the frequency of those events has increased by a factor of five over the past 50 years (WMO, 2021). During or in the aftermath of a natural disaster, timely response is absolutely essential. A response includes distribution of emergency supplies, such as medication, water, toolkits, and communication devices, to the impacted individuals. However, road infrastructure may be severely damaged and some roads may become impassable for vehicles. Aggravating this situation, dense cloud coverage, smoke, and vegetation may make satellite pictures of the terrain uninformative, and the state of many roads may remain unknown for a prolonged period of time. As a result, available supplies may not be distributed in time, leading to severe shortages (American Red Cross, 2015; Farzaneh et al., 2023). Emergency deliveries can be improved by using *unmanned aerial vehicles* (*UAVs*), or *drones* (see Figures 4.1–4.2).

In this article, we examine the delivery of medical, food, and other supplies by a truck and a drone as part of the immediate response phase. This means that the vehicles can depart immediately, tolerating the uncertainty on the traversability of the truck's scheduled route. What comes out is an online problem, since the condition of single road segments is revealed dynamically during the mission as the truck or the drone approaches the impassable segment. This requires from the developed delivery policies a real-time adjustment of routing decisions at each edge blockage.

We formulate the delivery problem as *the Traveling Salesman Problem with a truck and a drone under incomplete information* (*TSP-DI*). To provide the first impression of TSP-DI, Figure 4.3 illustrates an instance and a possible delivery plan for the *complete-information counterpart* of TSP-DI, i.e., for the problem in which the status of each edge is known. TSP-DI resembles *the Traveling Salesman Problem with a drone (TSP-D)* (Agatz et al., 2018), because both the truck and the drone can perform deliveries, the drone can carry one package at a time, the drone has to meet the truck periodically at one of the admissible rendezvous nodes to pick up packages, and the drone can be retrieved by the truck at the same node where it is launched. However, there are two main differences between our problem and TSP-D. Firstly, we allow the drone to take off and land on the truck at nodes that are not required to be visited. Secondly, the information on the edge status is incomplete.

Figure 4.1: A drone transports a lunchbox to an elderly person in a remote village cut off by road damage in Penela, Portugal

CC BY-SA 4.0 (Paulo Novais)

Figure 4.2: A drone delivers prescription medication to a remote hospital in Texas

CC BY-SA 3.0 (Tim C. Cox)

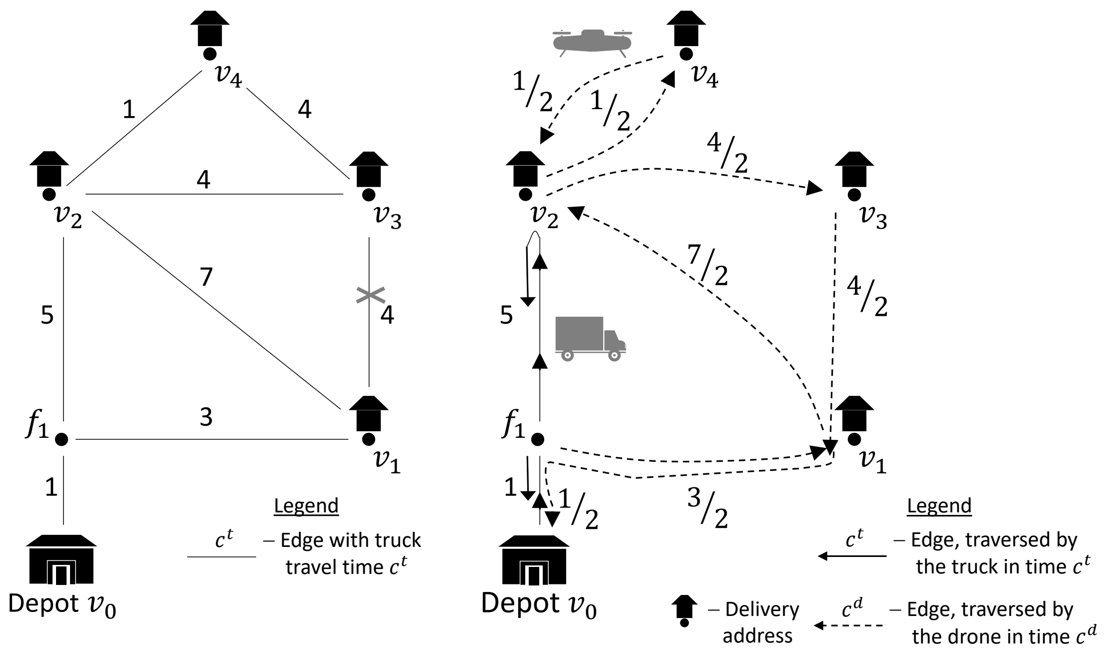
We examine several important routing policies for TSP-DI and provide their *competitive ratios*. Consider an online minimization problem, i.e., a problem with incomplete information, where some of this information can be revealed in the future. Let A be an online approximation algorithm. The *competitive ratio* $\sigma(A)$ is the worst-case ratio of the online algorithm's cost to the cost of an optimal offline algorithm, where all data are known a priori (Jaillet & Wagner, 2008a). If $A(I)$ is the objective value of algorithm A on instance I, I^* is the respective instance with complete information, and $OPT(I^*) > 0$ is the optimal objective value of I^*, then

$$\sigma(A) = \sup_I \frac{A(I)}{OPT(I^*)}.$$

The exact value of $\sigma(A)$ provides crucial insights for decision making.

These insights cannot be gleaned by running simulations. Firstly, unlike simulations, it provides a performance guarantee for the algorithm in any possible scenario. Further, if delivery times in some disaster relief operation turned out to be long, the competitive ratio indicates how much of the delivery time is attributed to the severity of the disaster and how much can be potentially shortened by improving algorithms. For example, if the competitive ratio is close to one, then no algorithm is able to achieve a better result; so, if the resulting objective value is bad, it can be attributed to the severity of the disaster.

Figure 4.3: Example of a *complete-information counterpart* of TSP-DI with one damaged edge (v_1, v_3)



Note. Left figure: An instance with $\alpha = 2$, i.e., the drone is two times faster than the truck.
Right figure: An optimal solution for this instance with makespan 13. The drone leaves v_0 on the roof of the truck; it launches in f_1 to deliver to v_1 and returns to the truck in v_2. The truck delivers to v_2. While the truck remains in v_2, the drone makes a circular sortie to deliver to v_4. Finally, the drone launches in v_2 to deliver to v_3, and both vehicles return to the depot.

Our contributions are as follows. Firstly, we calculate and prove competitive ratios of several important routing policies for the truck and the drone in TSP-DI. To the best of our knowledge, we are the first ones to perform the competitive ratio analysis in the context of truck-and-drone routing. The conducted competitive analysis is *parametric*.

Secondly, based on our analysis , we formulate nontrivial managerial insights. For instance, our analysis underscores the role of information collection: In many cases, the drone should take a detour to examine the status of some pivotal road segments in advance. Furthermore, our analysis also reveals that a straightforward online reoptimization policy has a bad competitive ratio and is outperformed by alternative policies.

We proceed with the literature review in Section 4.2 and a problem description in Section 4.3. Section 4.4 summarizes the main results of this paper – the competitive ratio results for three important policies. This section can be understood without going into the technical details of the proofs provided in Section 4.5. Section 4.6 presents computational experiments and an improved planning policy. We conclude with a discussion and an outlook in Section 4.7.

4.2 Literature review

Studies on humanitarian logistics, including routing of supply vehicles in the aftermath of a disaster, have recently attracted a considerable amount of attention, see the reviews of Farahani et al. (2020) and Özdamar and Ertem (2015) and Kundu et al. (2022). Several papers investigate the routing of repair crews to restore the damaged edges under *complete information* (e.g., Faiz et al., 2024; Moreno et al., 2020; Shin et al., 2019).

Much less attention is paid to a more realistic setting of *incomplete information* on the state of the infrastructure. The available research focuses on two-stage algorithms for relief distribution performed by *(homogeneous) trucks*, in which uncertainty impacts only certain long-term decisions, such as the location of depots or fleet sizing, and the information on the road condition is revealed in the second stage, when the delivery routes of the vehicles are planned (e.g., Ahmadi et al., 2015; Moreno et al., 2018; Rath et al., 2016). Among these articles, only a few investigate cases where relief distribution is performed by trucks *and drones* (Chowdhury et al., 2017; Dukkanci et al., 2023; Golabi et al., 2017), see also the surveys of Otto, Agatz, et al. (2018) and Yucesoy et al. (2024). Several studies investigate a pure monitoring or assessment of damage by off-road vehicles such as drones or motorcycles without considering the *following* relief *distribution* (e.g., Oruc & Kara, 2018; Reyes-Rubiano et al., 2021, 2022; G. Zhang et al., 2023; G. Zhang et al., 2021).

In contrast, similar to our study, Van Steenbergen et al. (2023) consider *dynamic* routing decisions for post-disaster relief distribution with trucks and drones, without exploiting the latter for damage assessment. They address travel time uncertainty for trucks due to infrastructure damage, excluding full road blockages. As stochastic travel times are realized during the arc's traversal, drones benefit from short, deterministic travel times but are unable to conduct surveillance. Two deep reinforcement learning approaches are used to solve the formulated dynamic stochastic program. To the best of our knowledge, only Macias et al. (2020) and Farzaneh et al. (2023) consider simultaneous network damage assessment *by drones* and relief distribution. However, in both articles, only trucks perform deliveries of emergency supplies; i.e., the advantage of the drone's capability to fly over road debris is ignored. In Macias et al. (2020), the damaged edges remain passable for the trucks, just the truck traversal times increase, which is different from our model. Macias et al. (2020) consider the normal distribution for the truck traversal times on damaged edges and propose a greedy-based heuristic and a genetic algorithm for the formulated problem. Farzaneh et al. (2023) consider an integrative heuristic planning framework, which includes the road damage assessment by drones, road recovery and relief distribution by trucks. However, the drone routing for damage assessment *does not anticipate* the impact of the gained information on the relief distribution; the objective is to scout the impacted region completely at minimum cost. Furthermore, the relief distribution framework is different: Trucks attend only one customer (demand node) per trip in a direct way there and back from the depot; they can only use roads, which are known to be intact, and disrupted roads can be repaired after a certain amount of time by recovery teams. To the best of our knowledge, no studies on truck-drone routing policies for simultaneous network damage assessment and deliveries in case of impassable road blockages have been performed so far. Moreover, the competitive ratio analysis for truck-and-drone missions in disaster relief that we present is new.

A separate thread of the literature studies competitive ratios of *online algorithms* in routing applications with incomplete information. Ausiello et al. (2001) were the first ones to consider an online variant of the TSP, where the requests (i.e, the nodes to be visited) appear dynamically over time. From the analytical point of view, the aforementioned TSP problem with node uncertainties deviate significantly from the setting studied in this paper. Liao and Huang (2014) were the first to study an online TSP with uncertainty about edge traversability. They called the problem Covering Canadian Traveler Problem (CCTP) and proposed a touring policy with an attractive competitive ratio for one traveler who dynamically encounters blocked edges on her pass, in case the graph remains connected. H. Zhang et al. (2015) added Steiner nodes to this formulation, i.e., nodes that can be visited once, more than once, or never, and proposed an exponential-time algorithm with the best-possible competitive ratio and a well-performing polynomial-time algorithm. Further variations of the CCTP and the online Steiner Traveling Salesman Problem with edge blockages have been studied in recent years, for example, with multiple agents, advanced information on blocked edges, or with the minimum latency objective (Akbari & Shiri, 2021; Akbari & Shiri, 2022; Büttner & Krumke, 2016; H. Zhang et al., 2016, 2019; Y. Zhang et al., 2022). TSP-DI is different from the studied problems as it involves a drone, which is able to fly over the blocked edges. Additionally, nontrivial *synchronization* of the delivery tours of the truck and the drone invalidates the results of competitive analysis for the truck-only case.

To the best of our knowledge, we are the first ones to perform competitive analysis for truck-and-drone routing in disaster relief. Moreover, ours is the first study on online delivery policies for the truck and the drone in case of incomplete information on impassable road blockages. In the preliminary work of the authors, several heuristic algorithms are presented for the problem variant of TSP-DI with given probabilities of arc damages (Otto, Poikonen, & Golden, 2018).

4.3 Problem statement

Before we state TSP-DI in Section 4.3.2, we formulate its *complete-information counterpart* in Section 4.3.1 first, which we mark with * as TSP-DI*. Section 4.3.3 discusses assumptions.

4.3.1 Problem statement under complete information

We consider a truck and a drone. The drone can deliver packages, but can carry only one package at a time. In contrast, the truck always carries a sufficient number of packages. The street network is described as a undirected graph $G = \left(L, E, c^t\right)$, where L is the set of nodes, E defines the set of edges, and c^t are non-negative edge weights. The set L consists of subsets of nodes $D \subseteq L$ and $V \subseteq D$ with additional properties.

The subset $D \subseteq L$ includes locations where the drone can safely take off from the truck or land on the truck, such as parking lots. The drone lands on the truck to pick up the next package. The drone may also traverse an edge while parked safely on the roof of the truck. The set D includes the depot v_0.

The subset $V \subseteq D$ refers to *delivery addresses*, each of which demands *one package* to be delivered by either the drone or the truck.

Observe that the nodes in V are *required* to be visited *at least* once either by the truck or by the drone. The remaining nodes are so-called *Steiner nodes* since they can be skipped or visited any number of times by the vehicles.

In the following, we write f_i for nodes in $D \setminus V$ (pure parking lots) and l_i for nodes in $L \setminus D$ (we will elaborate on these nodes in Section 4.3.2).

The set of undirected edges $E \subseteq \{(i,j) | i,j \in L\}$ describes road segments between nodes $i,j \in L$. We define E such that the graph G is connected, but not necessarily complete. For convenience, we define $c^t(i,i) := 0 \; \forall i \in L$. Some subset $\mathcal{E} = \{(i,j) | i \neq j\} \subseteq E$ of these edges is damaged. Edge weights $c^t(i,j) \geq 0$ represent travel times. We normalize the truck speed to be 1 and the drone speed to be $\alpha > 0$, i.e., $c^t(i,j)$ is the travel time needed for the truck to traverse $(i,j) \in E$, and by $c^d(i,j) = \frac{1}{\alpha} \cdot c^t(i,j)$ we denote the drone flight time for this edge. We call a set of delivery addresses that share the same location an *agglomeration*. Formally, if some $v_i, v_j \in V$ belong to the same agglomeration, then $c^t(v_i, v_j) = c^d(v_i, v_j) = 0$, and the status of edge (v_i, v_j) is known and it is not damaged.

We define a *walk* of a vehicle as a sequence of nodes in G consisting of launch, return, and delivery nodes. We mark launch, return, and delivery nodes with superscripts ^{L}, ^{R}, and with the overline $^-$, respectively. If there are several launches and returns to the same node, then this node is listed in the walk the respective number of times – once for each launch and once for each return (e.g., for a sortie that starts and ends at this node – as v_i^L and v_i^R). If the drone departs from the depot on the roof of the truck, we treat this node v_0^R as a return node for convenience. Consider the example in Figure 4.3, which contains one damaged edge (v_1, v_3). We emphasize that v_1, v_2, v_3, and v_4 denote delivery addresses. In the case of *complete information*, the damage of edge (v_1, v_3) is known; in the illustrated solution, the drone travels on the truck that departs from v_0 to f_1, then it launches to perform delivery to v_1 and returns back to the truck at node v_2; thereafter, it performs a cyclic operation from v_2 to deliver to v_4, and finally, it launches at v_2 to deliver to v_3 and returns along the shortest route to the depot v_0. The resulting drone walk is sequence $\pi^d = (v_0^R, f_1^L, \overline{v}_1, v_2^R, v_2^L, \overline{v}_4, v_2^R, v_2^L, \overline{v}_3, v_1, f_1, v_0^R)$ and the truck walk is $\pi^t = (v_0^R, f_1^L, \overline{v}_2^R, v_2^L, v_2^R, v_2^L, f_1, v_0^R)$.

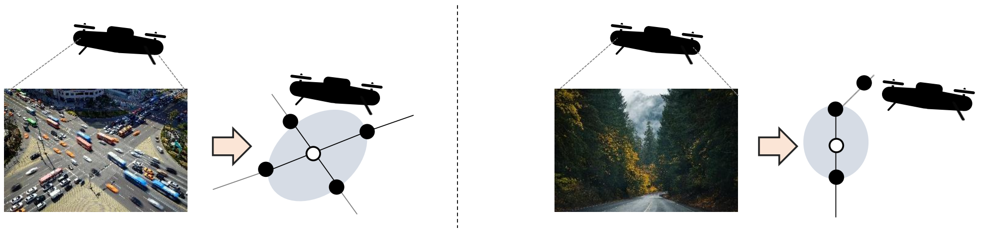
A *solution* of TSP-DI* consists of a *truck walk* π^t and a *drone walk* π^d such that:

- Both walks start and end at the depot.
- The truck walk contains *no* damaged edges.The drone walk can contain damaged edges.
- All launch and return nodes are eligible parking lots, i.e. they belong to subset D. In the walk of any vehicle, each launch node is followed by a return node as well as each return node (unless this is the first node in the walk) is preceded by a launch node.
- Each delivery address $v_i \in V$ is visited at least once either in the truck walk (truck delivery) or in the drone walk (drone delivery). Since the drone can carry only one package at a time, there is a return and a launch node between any two deliveries in the drone walk. We assume that the delivery takes no time.
- The walks of the truck and the drone are interrelated, since they physically meet each other for launches and returns, therefore the subsequences of launch and return nodes are identical in the truck walk and in the drone walk.

In order to verify the conditions above, we denote a solution s of TSP-DI as *a sequence of operations and travel legs*. Each *operation* is associated with exactly one delivery performed by the drone, and consists of two sequences of nodes: The truck walk and the drone walk that start at the preceding launch node and end at the subsequent return node of this delivery. Each *travel leg* describes that the drone travels on the roof of the truck, and is denoted as the respective (one) sequence of nodes. The solution in Figure 4.3 consists of one travel leg (v_0, f_1) and three operations $((f_1, v_2)(f_1, v_1, v_2))$, $((v_2, v_2)(v_2, v_4, v_2))$, and $((v_2, f_1, v_0)(v_2, v_3, v_1, f_1, v_0))$.

The objective is to find a solution s of TSP-DI* which minimizes the duration or *makespan* $T(s)$, i.e., the time until both the truck and the drone return to the depot after having completed the required deliveries. Makespan $T(s)$ of solution s equals the sum of the durations of its travel legs and operations. In a given solution s, the duration $\tau_{o(s)}$ of operation $o(s)$ is the maximum of the truck travel time $\tau_{o(s)}^t$ and the drone travel time $\tau_{o(s)}^d$ in this operation: $\tau_{o(s)} = \max\{\tau_{o(s)}^t, \tau_{o(s)}^d\}$. Here and in the following, we simplify notation and skip the reference to solution s, whenever this solution is clear from the context. For example, we write τ_o instead of $\tau_{o(s)}$. The duration of the travel leg depends on the truck travel time only. The duration of travel leg (v_0, f_1) in the solution of Figure 4.3 is 1. The travel times of the truck and the drone in operation $((v_2, v_2)(v_2, v_4, v_2))$ are 0 and 1, so that the operation's duration is $\max\{0, 1\} = 1$. The durations of the remaining two operations are 5 and 6, so that the makespan of this solution equals 13. It is an optimal solution for the instance in Figure 4.3.

Figure 4.4: Illustrative example for modeling the visibility of the roads



©Adobe Stock on the integrated fotographies

Note. The truck and the drone only see the travel conditions of an edge after reaching one of its defining nodes. The white node in the gray area marks the current position of the vehicle.

We denote an instance with the complete information on edge damages as I^*, the makespan of the optimal solution for instance I^* as $T^* = OPT(I^*)$, and an optimal solution of I^* as s^*.

4.3.2 Problem statement under incomplete information

An instance I of TSP-DI shares all characteristics with its complete information counterpart (instance) I^*, except that we do not know initially whether certain edges in E are damaged. Observe that in problems under incomplete information, we do not talk about solutions, but about *policies*. Policies specify, how the truck and the drone should act – such as "wait", "move from a node v_i along an edge (v_i, v_j) in G" – in each possible information state of the system. Luckily, since we analyze particular given policies, we do not need to formalize the state and action space completely, which would result in some cumbersome notation.

We introduce the information acquisition below and state policies informally in Section 4.4 and formally in Section 4.5.

Edges with unknown status are identified as "damaged" or "intact" only after either the drone or the truck have visited one of its adjacent nodes. Therefore, we call *all* the nodes in L *lookout* nodes. Lookout nodes model locations from which the travel condition of the adjacent road segments can be observed, see Figure 4.4. A curvy road in a dense forest or in a dense urban terrain has very poor visibility and can be modeled by a path in G with many lookout nodes connecting several edges which represent short road segments. On the other hand, a straight road segment in flat terrain with a clear view can be represented by only one edge and its two adjacent lookout nodes. A similar information acquisition process is used, for instance, in the Canadian traveler problem (Aksakalli et al., 2016; Papadimitriou & Yannakakis, 1991). We assume that $\mathcal{E}$ is *initially unknown*, except for those edges adjacent to the depot v_0.

To sum up, in this article, we examine competitive ratios of certain policies (algorithms A) for any instance I of TSP-DI. The policies A, which we examine in this article, always result in a feasible solution for the complete-information counterpart I^* and the value of the objective function (makespan) of this solution can be calculated as defined in Section 4.3.1.

4.3.3 Discussion of assumptions

We make several assumptions and leave the study of the respective extensions for future research:

- *Immediate information transmission. No repairs and no further damages.* As soon as either truck or drone has an update on the status of a road segment, the information immediately reaches the other vehicle. The set $\mathcal{E}$ of damaged edges does not change.

- *Unlimited truck capacity.* The truck carries a sufficient number of identical packages.
- *Sufficient energy.* We assume that the drone energy is sufficient to cover the longest drone sortie in the examined policies. For instance, in case of Reopt, the drone energy is sufficient to reach the next rendezvous location if an edge damage in the truck walk has been discovered. Section 4.6.3 compares the duration of drone sorties in different policies.
- *The same fields of view of the truck and the drone.* For simplicity of exposition, we assume the set of lookout nodes L to be the same for the truck and the drone, i.e., the fields of view of the truck and the drone are identical. Computational experiments in Section 4.6.2 illustrate the case of a larger field of view for the drone.

To simplify the exposition and *w.l.o.g.*, in the following, we discuss only the case when both vehicles use the *same road network*. Observe that the stated lower bounds on the competitive ratios remain valid also for the generalized case, because the presented worst-case instances are applicable for this generalization. Furthermore, it is straightforward to see that the proofs of the stated upper bounds on the competitive ratios remain valid as well, as long as the drone is *allowed* to follow the same edges as the truck. This is, for instance, the case when the drone is allowed to take 'crow-fly' *shortcuts* additionally to using the existing edges E of the street network, meaning it can fly directly between *any* two points.

4.4 Competitive ratios of several important policies

In this article, we analyze the competitive ratio for the widely used *optimistic online reoptimization* policy *(Reopt)* and two alternative more conservative policies – *the conservative delivery* policy *(CD)* and *the surveillance-first* policy *(SF)*. SF uses drone surveillance since other surveillance alternatives are costlier and often unavailable. For example, dense vegetation and smoke may hinder informative satellite imagery and pose risks for helicopter deployments.

Optimistic online reoptimization. In Reopt, the planner assumes all the edges with unknown status are intact. Therefore, the truck and the drone start performing deliveries based on an optimal solution of the resulting TSP-DI instance with no edge damages. In the course of their walks, the truck or the drone might discover damaged edges that affect the currently planned truck route. Each time such a damage is discovered by either of the vehicles, the delivery walks of both vehicles are reoptimized given their current positions and assuming all the edges with yet unknown status are intact (see Section 4.5.1 for details). *Reopt* is considered attractive by practitioners, because deliveries start immediately and are performed in the shortest possible time, if there are no damaged edges on the truck walk. Also observe that, until recently, economically viable and fast surveillance options, such as drones, simply did not exist.

Conservative delivery. In CD, deliveries are performed only by the drone, which collects a package at the depot v_0 each time and makes a return flight to a customer along the shortest trajectory. This policy is the best possible algorithm if all the edges are damaged.

Surveillance first. In SF, the initial truck and drone delivery walks are constructed under the assumption that all edges with unknown status are intact. But before the truck departs from the depot, there is a surveillance phase during which the drone clarifies the status of all the edges of the planned truck walk starting from the depot. Each time the drone discovers that an edge of the truck walk is damaged, a new delivery tour for the truck and the drone is calculated assuming all the edges with yet unknown status are intact. Neither vehicle starts deliveries before the drone has returned to the depot and has confirmed that all the edges in the truck walk of the currently planned delivery solution are intact (see Section 4.5.3 for details). In disaster relief, the decision to pause deliveries until more information on road status is collected is always difficult because each minute counts.

In our analysis, we parametrize the competitive ratio. Observe that in TSP-DI, no policy may have a constant competitive ratio. It is in line with a similar observation for the truck-only case of the Canadian Traveler Problem (cf. H. Zhang et al., 2015). We provide the value for the competitive ratio for each combination of the following parameter values: the number of damaged edges $k \in \mathbb{N} \cup \{0\}$, the ratio α between the travel time of the truck and the drone, and the number of required deliveries $|V| \in \mathbb{N}$.

Table 4.1 reveals significant differences in the competitive ratios of the described policies, proven in the subsequent sections. For instance, with a typical speed ratio of drone to truck $\alpha = 2$, conservative strategies CD and SF notably outperform *Reopt* in an illustrative example. Next, we summarize managerial insights based on our analysis of the competitive ratios.

Table 4.1: Parametric competitive ratios of some policies

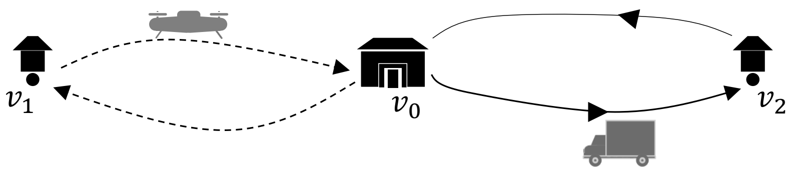
Optimistic online reoptimization	Conservative delivery	Surveillance first
$\begin{cases} 1 & \text{if } \lvert V\rvert = 1, \alpha > 1, \\ \geq 2^k & \text{otherwise.} \end{cases}$	$\begin{cases} 1 + \frac{\lvert V\rvert - 1}{\alpha} & \text{if } \alpha \geq 1, \\ \frac{\lvert V\rvert}{\alpha} & \text{if } \alpha < 1. \end{cases}$	$\begin{cases} 1 & \text{if } \lvert V\rvert = 1, \alpha > 1, \\ 1 + \frac{k+1}{\alpha} & \text{otherwise.} \end{cases}$
Illustration for $k = 5$, $\lvert V\rvert = 30$, and $\alpha = 2$		
≥ 32	15.5	4

Reopt is optimal whenever the truck encounters no damages in its walk ($k = 0$). But it is a costly gamble. If several damaged edges are encountered, *Reopt* may end up generating extremely poor solutions: Myopic truck drivers select risky routes based on the slightest promise of shorter delivery times and may have difficulties in returning to the depot. This behavior is typical and is known in psychology as risk seeking in the domain of losses (Kahneman & Tversky, 2000). The worst-case for *Reopt* tends to occur when there are multiple impassable consecutive pathways between a pair of delivery addresses, which eventually causes a long backtrack. Since damaged edges are often clustered in highly impacted regions, such blocking of multiple alternative pathways can reasonably occur. For example, flooding or storm surge could damage clusters of road segments leading to massive backtracking, especially if an island or coastal area is impacted. If the drone is faster than the truck ($\alpha > 1$), the more conservative policy SF dominates *Reopt* already for $k \geq 2$ in terms of the competitive ratio. As expected, CD may become very attractive if the drone is fast and the number of addresses is rather small. For example, if $\alpha > 1$, it dominates SF when $|V| < (k+2)$.

SF performs surprisingly well if measured by the competitive ratio. It may outperform *Reopt* even if the drone is slow ($\alpha < 1$), the advantage grows quickly with each damaged edge in the truck walk. To sum up, an optimal policy should include surveillance in one form or another as soon as the truck should deliver to some of the addresses.

Observe that competitive analysis makes no statement about the average performance of the algorithms. Even if the worst-case performance of some policy is bad, it may

Figure 4.5: Illustration for Lemma 14: An instance with $V = \{v_1, v_2\}$



Note. The figure visualizes only a portion of the graph, the k damaged edges are located somewhere in the not-visualized part of the graph. Distances are selected such that the drone delivers to v_1, and the truck delivers to v_2; both vehicles return at the same time T^* to the depot in a complete information optimal solution (CIOS).

perform quite well on average. Yet, in disaster relief, the worst outcomes are to be avoided. Therefore, the knowledge of the competitive ratios should complement the simulation results as an assessment tool for the selection of a suitable policy.

4.5 Proofs of competitive ratios

In this section, we provide the proofs for the competitive ratios, which were stated in Section 4.4.

In the following, we refer to the truck and the drone solving instance I as *a policy solution* (PS), and to those optimally solving instance I^* as *a complete information optimal solution* (CIOS). To streamline our exposition, we make the following two assumptions. First, we have to decide how to proceed with the cases of ties. We assume that in case of ties, a policy selects the *worst possible* decision for the examined instance. For example, if $|V| = 1$ and the truck and drone are equally fast with $\alpha = 1$, we assume that the Reopt-planner sends the truck to perform this only delivery. An alternative tie-breaker would not change the general direction of our conclusions. Nevertheless, we carefully mark the cases, where this technical assumption is used as well as highlight its consequences. Second, if we have to construct an unfavorable instance with many delivery addresses $|V|$, we usually place most of them in one location as an agglomeration. Not only does this serve as a valuable technical tool, but agglomerations are also quite common in disaster relief. For instance, a village in a sparsely populated area or a dormitory suburb of a city can be modeled as agglomerations. We use $[a, b] := \{a, \ldots, b\}$ to denote the set of integers between a and b, $a, b \in \mathbb{Z}$. We use the word *loop* in its *mundane* meaning to descriptively refer to loop-like formations of edges that form a cycle with a fixed start point. We start with Lemma 14 (see Figure 4.5), which we use throughout the section. Lemma 14 is proven in the Supplement 4.9.1

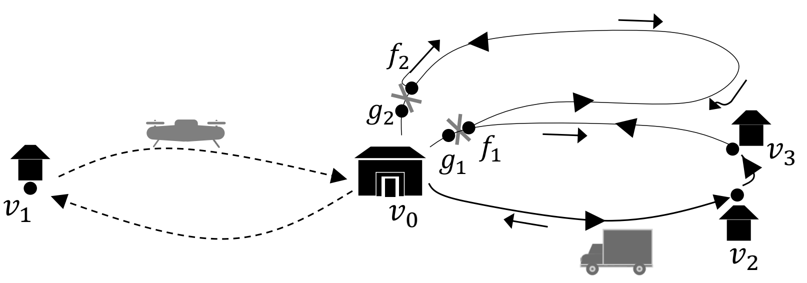
Lemma 14. *For any $|V| \geq 2$, $\alpha > 0$ and $k \in \mathbb{N} \cup \{0\}$ we can always construct an instance where the set V is divided into two subsets V_1 and V_2 and the damaged edges are such, that in CIOS the drone delivers to all $v \in V_1$, the truck to all $v \in V_2$, and both finish at the same time T^*.*

In the following proofs, we first find an *unfavorable* instance I' with $\frac{A(I')}{OPT(I'^*)} = w$ for some $w \in \mathbb{R}^+$. Then, we prove the *non-existence* of instance I with $\frac{A(I)}{OPT(I^*)} > w$.

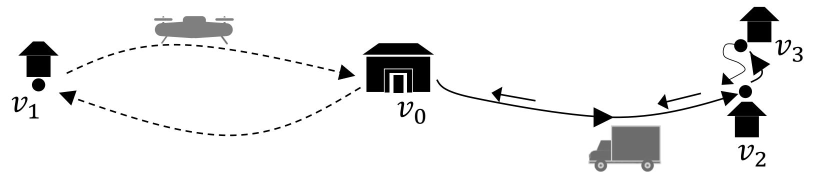
4.5.1 Analysis of the optimistic online reoptimization policy

In Reopt, the planner starts with an initial policy solution $s^{\diamond,0}$ of makespan $T^{\diamond,0} := T(s^{\diamond,0})$. Recall that the initial policy solution $s^{\diamond,0}$ finds an optimal delivery tour under the

Figure 4.6: An unfortunate example for *Reopt* with $k = 2$ and $|V| = 3$



(a) Policy solution (PS)



(b) Complete information optimal solution (CIOS)

Note. The truck in PS delivers to address v_2 and v_3 along the lower *loop* to the right of the depot. It encounters damaged edge (f_1, g_1) at the end of its travel and decides to drive to the depot along the upper *loop*. But because of another damaged edge (f_2, g_2), it has to travel all the way back.

assumption that all the edges with unknown status are intact. Observe that $T^{\diamond,0} \leq T^*$. At each occurrence i, when either the truck or the drone discovers a damaged edge in the current truck walk, *Reopt* immediately updates the current truck-and-drone tour given the current positions of the vehicles; thereby one of the vehicles may still be traversing its edge. Denote the time passed from the beginning of the truck-and-drone tour to the moment of the ith such update as θ_i. We somewhat compromise on notation and denote $s^{\diamond,i}$ as an optimal solution of the following ('remaining') TSP-DI instance: At θ_i, given the current positions of the truck and the drone and assuming all edges with currently unknown status intact, start delivering packages to the remaining delivery addresses while the part of schedule $s^{\diamond,i}$ traversed up to time θ_i remains fixed. We denote the makespan of this solution as $T^{\diamond,i} := T(s^{\diamond,i})$. Obviously, $T^{\diamond,i+1} \geq T^{\diamond,i}$.

Theorem 4. *The competitive ratio of the optimistic online reoptimization policy is*

$$\sigma(A^{Reopt}) = \begin{cases} 1 & \textit{if } |V| = 1, \alpha > 1, \\ \geq 2^k & \textit{otherwise.} \end{cases}$$

Proof. For $|V| = 1$ and $\alpha > 1$, the policy sends the drone to deliver, since the drone is α times faster than the truck: If the drone's shortest driving time to the delivery address is T^*, then the truck would need the larger time $T^* \cdot \alpha$. The competitive ratio is 1 since PS and CIOS coincide.

As for the remaining cases, we start with examining the case of $|V| = 3$ and $\alpha > 0$. Figure 4.6a illustrates PS for an unfavorable instance with $V = \{v_1, v_2, v_3\}$ and $k = 2$.

Several *loops* start and end at the depot v_0. Node v_1 is positioned in the middle of the left *loop* of length $\alpha \cdot T'$. Nodes v_2 and v_3 are located on the right lower *loop* from the depot, on opposite sides of the *loop's* midpoint at a distance of ξ from the midpoint, where $\xi > 0$ is a *very small number*. The length of this *loop* is T', which equals the truck travel time as the truck speed is normalized to 1.

In total, there are k *loops* on the right side of the depot, each having a damaged edge $(f_i, g_i), i \in [1, k]$, located close to depot v_0, such that the truck travel time from f_i over g_i to v_0 would be ξ_i in case of no damage. We assume that the distances to the depot decrease in a geometric progression and define $\xi_i := 2^{1-i}\xi$ for $i \geq 1$. The ith *loop* shares the part from depot v_0 via g_{i-1} to f_{i-1} with *loop* $i-1$ and leads over f_i and g_i to the depot, for all $i \geq 2$. Further assume that the truck travel time on each *loop* $i \geq 2$, starting and finishing in the depot if all the edges were intact, is $2^{i-2}T' - \delta_i$ and $\delta_i > 0$ are monotonically increasing. Recall that the truck travel time is T' for the first *loop*.

If ξ is sufficiently small (see Supplement 4.9.3), then in CIOS, the drone delivers to v_1 and the truck delivers to v_2 and to v_3, taking the lower *half-loop* there and back (Figure 4.6b). The makespan of CIOS equals the truck driving time: $T^* = T' + 2\xi$.

Since *Reopt* treats all the unvisited edges as intact, the truck in PS, which has incomplete information, will decide to take the lower *loop* on the right side of the depot with a travel time T'. However, it encounters a damaged edge (f_1, g_1) at the end of its travel. Facing two alternatives, to return back taking time $T' - \xi$ or taking the next upper *loop* (*loop* $i = 2$) from f_1 via (f_2, g_2) to v_0 of duration $T' - \xi - \delta_2$, the naive truck in PS will choose the latter. Indeed, at the time of this decision, the truck is aware of the damaged edge (f_1, g_1), but does not know that edge (f_2, g_2) is damaged. At each point f_i, $1 \leq i < k$, the truck chooses *loop* $i + 1 \leq k$ to return to the depot. The travel time from f_i via f_{i+1} to v_0 is $2^{i-1}T' - \delta_{i+1} - \xi_i$, which is the length of *loop* $i + 1$ minus ξ_i. The travel time of the truck from f_i back to the depot using *loops* $(i, i-1, \ldots 1)$ is

$$\sum_{j=2}^{i} 2^{j-2}T' + T' - \sum_{j=2}^{i} \delta_j - \left(2\sum_{j=1}^{i-1} \xi_j + \xi_i\right) = 2^{i-1}T' - \sum_{j=2}^{i} \delta_j - \left(2\sum_{j=1}^{i-1} \xi_j + \xi_i\right). \quad (4.1)$$

If we define $\delta_2 = \xi$ and $\delta_{i+1} = \sum_{j=2}^{i} \delta_j + 2\sum_{j=1}^{i-1} \xi_j + \xi$ for $i \geq 2$, (observe that we added term ξ to avoid ties), then in PS the truck needs more time (by ξ) to drive the whole way back than to take the next upper *loop* to reach the depot and, thus, prefers the latter.

By replacing δ_i in the recursive definition of the ith *loop*'s length, we obtain for $i \geq 2$:

$$2^{i-2}T' - \delta_i = 2^{i-2}T' - \sum_{j=1}^{i-2} 2^{i-j-1}\xi_j - 2^{i-2}\xi. \quad (4.2)$$

Recall that $\xi_i = 2^{1-i}\xi$ and that $\sum_{j=1}^{i-2} 2^{i-j-1}\xi_j$ converges to the infinite sum of a geometric progression. We get

$$\sum_{j=1}^{i-2} 2^{i-j-1}\xi_j = \sum_{j=1}^{i-2} 2^{i-j-1+1-j}\xi = 2^i\xi \sum_{j=1}^{i-2} 2^{-2j} = 2^i\xi \sum_{j=1}^{i-2} \frac{1}{4^j} \leq 2^i\xi\frac{4}{3} = 2^{i-2}\xi\frac{16}{3}, \quad (4.3)$$

which leads to a lower bound on the length of the ith *loop*, see (4.2):

$$2^{i-2}T' - \delta_i \geq 2^{i-2}T' - 2^{i-2}\frac{16}{3}\xi - 2^{i-2}\xi = 2^{i-2}\left(T' - \frac{19}{3}\xi\right). \quad (4.4)$$

In the worst case, in PS the total truck travel time to reach f_k from depot v_0 taking *loops* $1, 2, \ldots, k$, and to travel all the way back to the depot as stated in (4.1), adds up to:

$$2 \cdot \left(2^{k-1}T' - \sum_{j=2}^{k} \delta_j - \left(2\sum_{j=1}^{k-1} \xi_j + \xi_k\right)\right) = 2 \cdot \left(2^{k-1}T' - \delta_{k+1} + \xi(1 - 2^{1-k})\right) \quad (4.5)$$
$$\geq 2^k \left(T' - \frac{19}{3}\xi\right),$$

therefore we used the definition of δ_{i+1} and of ξ_i for $i = k$ for the first transformation and the lower bound on the ith *loop*'s length computed in (4.4). Recalling $T^* = T' + 2\xi$, observe that this bound approaches $2^k T^*$ for ξ converging to 0. We conclude that $\sigma(A^{Reopt}) \geq 2^k$, since *Reopt* may perform worse in some other instances. Also observe, that the bound is valid both for the cases of a fast drone with $\alpha \geq 1$ and for the cases of a slow drone with $0 \leq \alpha < 1$.

If $|V| > 3$ (and any $\alpha > 0$), we can construct a similar instance by putting nodes $v_j, j > 3$, in the location of v_2 as an agglomeration.

If $|V| = 2$ and $0 \leq \alpha < 1$, we can straightforwardly adjust the above instance and drop v_1.

In the remaining cases of ($|V| = 2$ and $\alpha \geq 1$) as well as ($|V| = 1$ and $\alpha \leq 1$), we rely on our technical assumption that in case of ties, PS always makes the worst possible decision. To construct an unfortunate instance, we drop v_3 from the example in Figure 4.6, move v_2 exactly *in the middle* of the right lower *loop* and set $T' = T^*$. In case of $|V| = 1$ and $\alpha \leq 1$, we also remove node v_1; observe that the truck performs only one delivery in PS. When the truck in PS reaches node v_2, it has a choice between two paths to the depot of the same length $\frac{T'}{2}$, either taking the same way back which is known for sure to be intact, or traveling along the uncertain upper half of the lower right *loop*. Because of our technical assumption on the ties, the truck decides on the latter. The remaining proof proceeds along the same lines as above. In Supplement 4.9.2, we establish a lower bound on the competitive ratio that does not rely on the technical assumption about the ties (but for a very special case of $|V| = 1$ and $\alpha = 1$) and show that also this lower bound increases exponentially in k. □

4.5.2 Analysis of the conservative delivery policy

In CD the drone has to deliver to all addresses in V starting from the depot. We refer to the CD algorithm as A^{CD} and to the competitive ratio of the conservative delivery policy as $\sigma(A^{CD})$.

CD performs badly if there is an agglomeration of delivery addresses far away from the depot, while the truck can reach these addresses, as we discuss in Lemma 15 and 16.

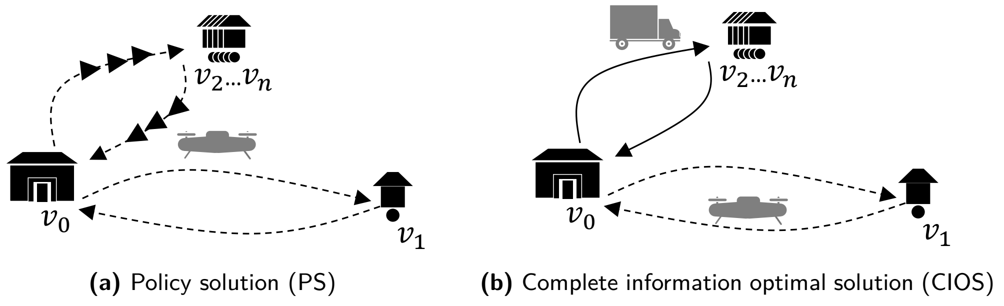
Lemma 15. $\sigma(A^{CD}) \geq 1 + \frac{|V|-1}{\alpha}$.

Proof. The case of $|V| = 1$ is trivial. If $|V| \geq 2$, consider the example of Figure 4.7, where v_1 is located such that in CIOS, the drone delivers to v_1 in a return flight from the depot in T^* time, and the truck delivers to an agglomeration of $|V| - 1$ addresses in T^* time (cf. Lemma 14). In PS, the drone performs all $|V|$ deliveries in a return flight from the depot in $T^* + (|V| - 1) \cdot \frac{T^*}{\alpha}$ time, so that the competitive ratio $\sigma(A^{CD})$ cannot be lower than $1 + \frac{|V|-1}{\alpha}$. □

Lemma 16. $\sigma(A^{CD}) \geq \frac{|V|}{\alpha}$.

Proof. In the example in Figure 4.8, the truck in CIOS delivers to an agglomeration of $|V|$ addresses in T^* time. In PS, the drone performs all $|V|$ deliveries in a return flight from the depot in $|V| \cdot \frac{T^*}{\alpha}$ time, so that the competitive ratio cannot be lower than $\frac{|V|}{\alpha}$. □

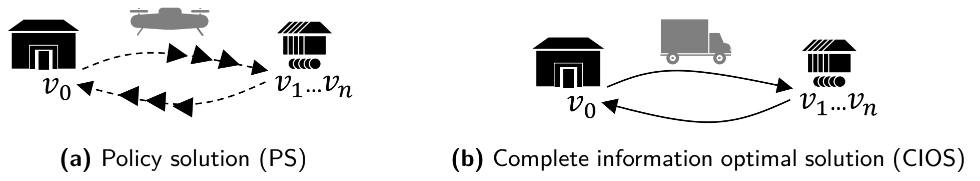
Figure 4.7: An unfortunate example for CD with $|V| = n \geq 2$



(a) Policy solution (PS) **(b)** Complete information optimal solution (CIOS)

Note. The drone in PS delivers to each address in V in a separate return flight from v_0. In CIOS, the truck delivers to the agglomeration of $|V| - 1$ addresses in a single return trip, the drone visits the remaining address v_1.

Figure 4.8: An unfortunate example for CD with $|V| = n \geq 1$



(a) Policy solution (PS) **(b)** Complete information optimal solution (CIOS)

Note. The drone in PS delivers to each address in V in a separate return flight from the depot. The truck in CIOS delivers to the agglomeration of $|V|$ addresses in a single return trip.

Theorem 5. *The competitive ratio of the conservative delivery policy is*

$$\sigma(A^{CD}) = \frac{|V| + \max\{\alpha - 1, 0\}}{\alpha}.$$

Proof. Consider two cases: $\alpha \leq 1$ and $\alpha > 1$.

The case of a slow drone with $\alpha \leq 1$ is quite obvious. Since each address is visited by at least one of the vehicles in CIOS, the drone can follow the complete CIOS tour of this vehicle from the depot to this delivery address and back to the depot. This results in a solution with a makespan of at most $\frac{|V| \cdot T^*}{\alpha}$, since the visit of each delivery address takes at most $\frac{T^*}{\alpha}$ time.

The case of a fast drone with $\alpha > 1$ is more complicated. We will show how to construct a feasible delivery plan s', in which only the drone performs deliveries and the truck never leaves the depot. The makespan of CD obviously cannot be larger than the one of s'. In s' for each $v \in V$, the drone performs one operation that starts and ends at the depot as follows:

- If v is delivered by the truck in CIOS, then the corresponding delivery tour in s' follows the complete truck walk of the CIOS solution s^* in $T^* - \sum_{o=1}^{n_o} (\tau_o - \tau_o^t)$ time, where $n_o := n_o(s^*)$ is the total number of operations in s^* and the expression on the right-hand side sums up the truck waiting time for the drone in all of these operations. We recall that τ_o is the duration of operation o in s^* given as the maximum of the corresponding truck- and drone time respectively: $\tau_o = \max\{\tau_o^t, \tau_o^d\}$. Since the drone is α times faster than the truck, the time it needs to perform this delivery tour for v in s' is

$$\frac{1}{\alpha}(T^* - \sum_{o=1}^{n_o}(\tau_o - \tau_o^t)). \tag{4.6}$$

- If for some operation $o_v := o_v(s^*)$, v is delivered by the drone in CIOS and then the delivery tour for v in s' follows the complete truck walk of s^* like above, except for o_v, where it follows the drone walk of the operation instead of the truck walk and still requires at most τ_o time. Thus, the time of this drone delivery tour for v in s' is at most

$$\frac{1}{\alpha}(T^* - \sum_{o=1}^{n_o}(\tau_o - \tau_o^t)) + \tau_{o_v} - \frac{\tau_{o_v}^t}{\alpha}. \tag{4.7}$$

In the example of Figure 4.3, node v_2 is delivered by the truck in CIOS and the respective drone walk in s' is $(v_0^L, f_1, \overline{v_2}, f_1, v_0^R)$. The delivery node v_1 is covered by the drone in CIOS, thus, the drone walk in s' is $(v_0^L, f_1, \overline{v_1}, v_2, f_1, v_0^R)$.

Considering that the number of operations n_o in s^* corresponds to the number of drone delivery nodes in this solution, the makespan of the solution s' sums up to:

$$T(s') \leq |V| \cdot \frac{1}{\alpha}(T^* - \sum_{o=1}^{n_o}(\tau_o - \tau_o^t)) + \sum_{o=1}^{n_o}(\tau_o - \frac{\tau_o^t}{\alpha}) \tag{4.8}$$

$$\leq \frac{|V|}{\alpha} \cdot T^* - \sum_{o=1}^{n_o}\frac{(\tau_o - \tau_o^t)}{\alpha} + \sum_{o=1}^{n_o}(\tau_o - \frac{\tau_o^t}{\alpha}) \tag{4.9}$$

where the last inequality follows from $|V| \geq 1$. Given that the makespan of CIOS is at least the total duration of its operations, i.e., $T^* \geq \sum_{o=1}^{n_o} \tau_o$, that $\alpha > 1$, and that we can eliminate terms $\sum_{o=1}^{n_o} \frac{\tau_o^t}{\alpha}$ in the last two summands, we conclude that

$$T(s') \leq \frac{|V|}{\alpha} \cdot T^* + \frac{\alpha - 1}{\alpha} T^* = \frac{|V| + \alpha - 1}{\alpha} \cdot T^*. \tag{4.10}$$

Given the makespans of the constructed drone-only delivery plans for both cases of $\alpha \leq 1$ and $\alpha > 1$, as well as given Lemmas 15 and 16, and from the definition of the competitive ratio, we conclude that $\sigma(A^{CD}) = \frac{|V| + \max\{\alpha - 1, 0\}}{\alpha}$. □

4.5.3 Analysis of the surveillance-first policy

A policy solution (PS) of surveillance first (SF) is divided into a surveillance phase and a delivery phase. Consequently, we will note in the following the route and makespan of these phases with superscripts s and d, respectively. In SF, the planner starts with initial delivery plan $s^{\diamond,d,0}$ of makespan $T^{\diamond,d,0} := T(s^{\diamond,d,0})$. Solution $s^{\diamond,d,0}$ is an optimal solution of the respective instance if all the edges with unknown status are intact. Observe that $T^{\diamond,d,0} \leq T^*$. The drone examines all the (not yet surveyed) nodes in the truck walk of $s^{\diamond,d,0}$ and returns to the depot before deliveries can start. If the drone returns to the depot after having discovered no damaged edges at time $T^{\diamond,s,0}$, then the makespan of the policy solution is $T^{\diamond,s,0} + T^{\diamond,d,0}$. At each occurrence $i \in \mathbb{N}$, when the drone discovers a damaged edge in the current truck walk, SF updates the current truck-and-drone tour to $s^{\diamond,d,i}$ and the drone has to inspect the (remaining) nodes with unknown edge status in the truck walk of the new delivery plan $s^{\diamond,d,i}$ in the surveillance phase. Obviously,

$T^{\diamond,d,i+1} \geq T^{\diamond,d,i}$. The overall makespan of PS in SF sums up from the surveillance time after k discovered damaged edges, which is $T^{\diamond,s,k}$, and delivery time $T^{\diamond,d,k}$.

The main idea of SF is to take a short amount of extra time to examine the truck walk with a (usually faster) drone *before* starting the deliveries.

When we think about disaster relief, the decision to pause deliveries until more information on road conditions is collected is always difficult since every minute counts. Thus, the surveillance-first policy seems overly conservative and its benefits are not quite intuitive at first glance. However, in the instance described in Figure 4.12, the surveillance-first policy performs surprisingly well for a fast enough drone. In particular, we benefit greatly from discarding some impassable paths early on because the detour is quite long. For instance, if the truck notices edge damage at node l_1, it has to travel the whole way back to the depot. Overall, our analysis shows that the surveillance time in SF may be quite small if the visibility of the roads is excellent.

Lemma 17 proves that $\sigma(A^{SF}) \geq \frac{k+1+\alpha}{\alpha}$ for $|V| > 2, \alpha > 1$, which is an important scenario with a fast drone and several delivery addresses. Supplement 4.9.4 shows the validity of the lower bound for the remaining cases of $|V| = 2, \alpha > 1$ and $|V| \geq 2, \alpha \leq 1$, if the policy always takes the worst possible decision in case of ties. Theorem 6 proves that this bound is tight.

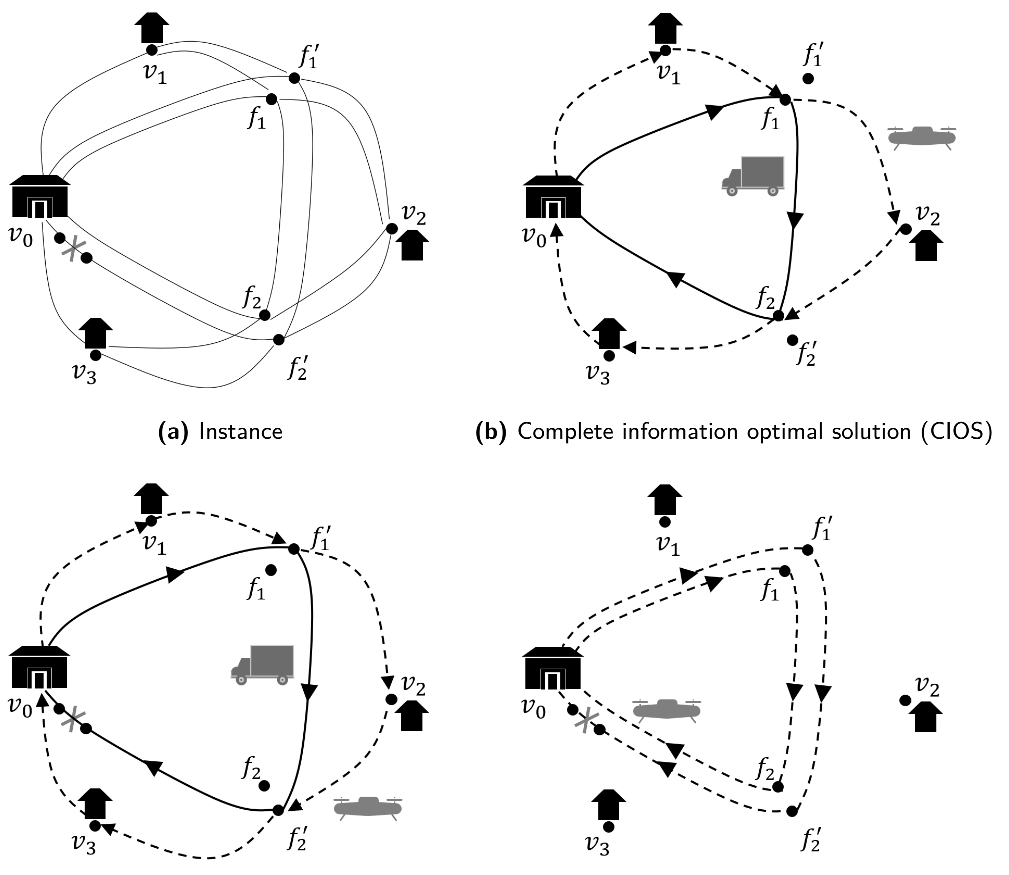
We analyze drone surveillance in more depth in Theorem 7. According to this theorem, any policy that does not include surveillance detours of the drone has a worse competitive ratio than SF for a range of practical parameter values, such as $k > 1$ (several damaged edges), $\alpha > \frac{k+1}{k-1}$ (the drone is faster than the truck) and $|V| > (k+1+\alpha)$ (many delivery addresses).

Lemma 17. *If $|V| \geq 3$ and $\alpha > 1$, the competitive ratio of SF $\sigma(A^{SF}) \geq \frac{k+1+\alpha}{\alpha}$.*

Proof. Consider instances with $|V| \geq 3$ delivery nodes as illustrated in Figure 4.9 for $|V| = 3$. There are $k+1$ *edge- and node-disjoint* cycles, which intersect only at the depot node. In each cycle there are $|V|$ equidistant nodes, namely the depot v_0 and $(|V|-1)$ Steiner nodes from $D \setminus V$. Between any pair of adjacent nodes of a cycle there is a path (*spike*) connecting these adjacent nodes and leading over exactly one of the nodes from V. We call the structure formed by one cycle together with its spikes a *circular path with spikes (CPS)*. Figure 4.9b illustrates one such CPS – it is the CPS which the vehicles traverse in CIOS. Observe, the vehicles do not have to visit the $(|V|-1)$ Steiner nodes of a CPS, but may use them for the drone's launch or landing. In the cycle over the Steiner nodes, starting and ending at the depot of a CPS p (in Figure 4.9b these are edges $(v_0, f_1), (f_1, f_2)$, and (f_2, v_0)), the travel distances between consecutive Steiner nodes $c^t(f^p_{i-1}, f^p_i)$ are all the same for all p and all $i \in [1, |V|]$, where the depot $v_0 \in D$ is denoted as f^p_0 and $f^p_{|V|}$, for all p, for convenience. Recall that the edges connecting Steiner nodes and delivery nodes form *spikes*, such that each delivery node $v_i \in V$ is a neighbor of the two respective successive Steiner nodes f^p_{i-1} and f^p_i of each CPS p. The travel distances are $c^t(f^p_{i-1}, v_i) = c^t(v_i, f^p_i) = \frac{\alpha}{2} \cdot c^t(f^p_{i-1}, f^p_i)$. Figure 4.9 illustrates all $(k+1)$ CPSs for an instance with $|V| = 3$ and $k = 1$. In k of the CPSs, we have a damaged edge *close* to the depot v_0, which cannot be detected from v_0.

Obviously, no feasible delivery tour can have a makespan of less than $T' = |V| \cdot c^t(f_{i-1}, f_i)$, which is the minimum time to visit all the delivery nodes by the fastest vehicle, the drone. Figure 4.9b depicts a CIOS with makespan $T^* = T'$, in which the truck travels the undamaged circular path $\pi^t = (v_0, f_1, f_2, ... f_{|V|-1}, v_0)$ taking time T' and the drone travels along the spikes of this circular path as $\pi^d = (v_0, v_1, f_1, v_2, f_2, ..., v_{|V|}, v_0)$ taking the same time T'. I.e., the drone delivers to all the addresses by picking packages from the truck in the Steiner nodes. Note that the vehicles reach the Steiner nodes at the same time and do not wait for each other.

Figure 4.9: An unfortunate example for SF with $|V| = 3, k = 1$ and $\alpha > 1$



(a) Instance **(b)** Complete information optimal solution (CIOS)

(c) Initial delivery plan $s^{*,0}$ **(d)** Complete surveillance tour in the policy solution

Note. In CIOS, the truck takes the undamaged circular path from the depot and launches the drone from its nodes for each delivery. Since the circular paths are indistinguishable for the online policy, SF proposes at each update of the delivery plan a truck walk through a damaged circular path, until all damages are detected. Since damages are detected only at the end of such path, the drone traverses all circular paths in the surveillance phase of SF, which results in a distance of $(k+1)$ times the truck walk in CIOS.

Since SF does not know about road damages, the initially planned truck walk in $s^{*,d,0}$ takes one of the damaged circular paths. In the surveillance phase, the drone traverses the complete circular path in $\frac{|V| \cdot c^t(f_{i-1}, f_i)}{\alpha} = \frac{T^*}{\alpha}$ time to discover the damage at the end of this walk. At each discovery of a damaged edge at the end of a circular path, SF suggests a new delivery route for the truck along yet another damaged path, until all k damaged edges have been discovered. Thus, in the surveillance phase, the drone traverses all $k+1$ circular paths in $T^{*,s,k} = (k+1) \cdot \frac{T^*}{\alpha}$ time. The makespan of PS sums up this surveillance time and the duration $T^{*,d,k} = T^*$ of the final delivery tour. We conclude that $\sigma(A^{SF}) \geq \frac{k+1+\alpha}{\alpha}$. $\square$

Theorem 6. *The competitive ratio of SF is*

$$\sigma(A^{SF}) = \begin{cases} 1 & \textit{if } |V| = 1, \alpha > 1 \\ \frac{k+1+\alpha}{\alpha} & \textit{otherwise} \end{cases}.$$

Proof. If $|V| = 1$ and $\alpha > 1$, the drone in PS will perform the delivery and $\sigma(A^{SF}) = 1$.

Lemma 17 and Supplement 4.9.4 prove that $\sigma(A^{SF}) \geq \frac{k+1+\alpha}{\alpha}$ if either $|V| > 1$ or $\alpha \leq 1$. It remains to show that $\sigma(A^{SF}) \leq \frac{k+1+\alpha}{\alpha}$.

Observe that the makespan of the delivery schedule in SF cannot exceed the objective value of CIOS: $T^{\diamond,d,i} \leq T^*$, because SF treats all the edges with not yet known status as intact when planning deliveries. Assume the drone in PS would finish surveillance and return to the depot (if all the remaining edges were intact) at time $T^{\diamond,s,i-1}$, but it discovers a damaged edge at time $\theta_i \leq T^{\diamond,s,i-1}$. In SF, the drone selects a shortest flight to investigate the not yet surveyed edges in the truck walk of the updated delivery plan $s^{\diamond,d,i}$. This cannot take more time than the following feasible walk: Approach the depot by completing the current surveillance flight at time $T^{\diamond,s,i-1}$ and follow the updated truck walk of $s^{\diamond,d,i}$ to reach the depot no later than time $T^{\diamond,s,i-1} + \frac{T^*}{\alpha}$. As the drone starts information collection at time 0, its initially planned duration cannot exceed $T^{\diamond,s,0} \leq \frac{T^*}{\alpha}$. Moreover, the drone performs at least 1 and at most $k+1$ such surveillance iterations: A damaged edge may be discovered in each of k consecutive surveillance iterations and the drone will make one more iteration to confirm that all the edges in the currently planned truck walk are intact. Consequently, the drone in PS will complete information collection and return to the depot no later than $T^{\diamond,s,k} \leq (k+1)\frac{T^*}{\alpha}$. Thus, we get an upper bound for $\sigma(A^{SF})$ by summing the received upper bound on the surveillance time $\frac{k+1}{\alpha} \cdot T^*$ and the upper bound on the delivery time T^*, and dividing this sum by T^*. □

Theorem 7. *For $k > 1$, $\alpha > \frac{k+1}{k-1}$, and $|V| > (k+1+\alpha)$, any truck-and-drone delivery policy without drone surveillance has a worse competitive ratio than SF.*

We define “without drone surveillance” as follows: The drone performs deliveries to some addresses $v \in V$ by moving along the *shortest path* from the launch node to address v and returning along the *shortest path* to the return node. See Supplement 4.9.5 for a proof.

4.6 Computational experiments

In this section, we describe the data set (Section 4.6.1) and investigate observed performance ratios of the presented policies in detailed simulations by using settings, common in the TSP-D literature (Section 4.6.2). For instance, we perform sensitivity analysis for the case of a slow drone ($\alpha < 1$), an alternative objective function (minimize last delivery time), nonzero delivery times, hybrid policies with partial initial drone surveillance, and differing fields of view of the vehicles. Since energy expenditure is a central aspect in the deployment of a drone, we examine the duration of drone sorties and, thus, the required energy, in different policies in Section 4.6.3.

4.6.1 Data generation and the hybrid policy

We generate 200 graphs with $|L| = 16$ nodes, by placing the nodes randomly in a 1000×1000 square. Afterward, we randomly assign $|V| = 5$ nodes to be delivery addresses and one node to be the depot. Following Letchford et al. (2013), we create sparse graphs since they most resemble real road networks. We keep the graphs connected to ensure feasibility: There is a road between each delivery address and the depot. We start with a complete graph and gradually reduce the number of edges to $|E| = 36$ (or 30% of $\frac{16 \cdot 15}{2}$) by randomly deleting edges that do not disconnect the graph. For each graph, we randomly generate 20 instances by setting 22 edges (or $\approx 60\%$ of $|E|$) to be damaged. This results in $200 \times 20 = 4000$ instances in total. Observe that *Reopt* and SF are computationally expensive, and require an optimal solution of TSP-DI* at each replanning iteration; this limits the size of the tested instances significantly.

In *the basic setting (Base)*, the drone is twice as fast as the truck ($\alpha = 2$) and $D := V \cup \{v_0\}$, i.e., the remaining nodes serve exclusively as surveillance nodes. We set the truck speed to 1.

We examine *seven* additional settings by changing one factor at a time in *Base*. In four settings, we vary the drone speed ($\alpha = 1$, $\alpha = 3$) and the incidence of the parking lots ($|D| = 11$, $|D| = 16$). Three further settings refer to widespread network characteristics in the TSP-D literature: the Euclidean metric (L_2) for the truck with drone shortcuts as well as the Manhattan metric (L_1) for the truck with and without drone shortcuts. In more detail, the drone flight is always measured by the Euclidean distance, i.e., $c^d(i,j) = \frac{1}{\alpha} \cdot \|(\vec{j} - \vec{i})\|_2$, where $\vec{i}, \vec{j} \in \mathbb{R}^2$ denote the coordinates of two nodes $i, j \in L$. In *Base*, the drone only follows edges $(i,j) \in E$. Alternatively, we allow the drone to travel *shortcuts* (shortest Euclidean distances) between its current position and any node $j \in L$. The truck distance is measured by the Euclidean metric in *Base* for the sake of comparison to the theoretical analysis, where the truck and the drone use the same metric. We examine the Manhattan metric for the truck by setting $c^t(i,j) = \|(\vec{j} - \vec{i})\|_1$ for $(i,j) \in E$.

For every instance, we report the *ratios* $\tilde{\sigma}(A)$ between the result of policy A and the CIOS result. We extend the dynamic programming approach of Bouman et al. (2018) to optimize the delivery phases in *Reopt* and SF. The surveillance phase of SF amounts to the traveling salesman problem, which we solve with a standard MIP with IBM ILOG CPLEX 22.1.1. We run experiments on a Linux server (6248R CPU, 3.00GHz×96 processors, and 754.5 GB RAM).

Our analysis in Section 4.5.3 suggests the benefits of drone surveillance in the case of a fast drone. To test this, we introduce an additional *hybrid policy* $\mathsf{S}^{\text{hybrid}}_{25\%}$, which integrates features of both SF and *Reopt*. Similar to SF, it starts with a surveillance phase; however, this phase is shorter, constrained by a runtime limit $c_{\max}$. To determine the surveillance route, we solve an orienteering problem, where the objective is to maximize the cumulative 'prizes' gathered from visiting nodes from the truck walk of the currently planned delivery tour, within the given time limit. As in SF, whenever a damaged truck edge is encountered during surveillance, provided $c_{\max}$ is not exceeded, the planned delivery tour and the remaining surveillance route are reoptimized. In $\mathsf{S}^{\text{hybrid}}_{25\%}$, the time limit $c_{\max}$ is set to 25% of the minimum required surveillance time to check the status of each edge in the initially planned truck walk by visiting one of its adjacent nodes. Supplement 4.9.6 provides details on hybrid policies with varying $c_{\max}$. Due to the limited surveillance phase, $\mathsf{S}^{\text{hybrid}}_{25\%}$ may still encounter damaged truck edges within the delivery tour, which, similarly to Reopt, leads to a reoptimization of the delivery routes to cover the remaining unvisited locations. A new feature of SF is that during the reoptimization of both surveillance and delivery phases, $\mathsf{S}^{\text{hybrid}}_{25\%}$ penalizes the use of edges with unknown status in the truck walk by applying a penalty factor to account for the increasing makespan due to a potential truck detour. This is because these edges might be damaged, potentially causing additional detour time for the truck. Based on calibration experiments conducted on separate instances, we multiply the length of such edges by 1.3 to account for this uncertainty (see the Supplement 4.9.6). More details and a pseudocode of the hybrid policy $\mathsf{S}^{\text{hybrid}}_{25\%}$ are provided in the Supplement 4.9.6.

4.6.2 Observed average and worst performance of the policies

In terms of the *observed average ratios (Avg)*, *Reopt* dominated SF in all the settings (see Table 4.2). Indeed, in as much as 10% of the instances (397 out of 4000), *Reopt*'s results

Table 4.2: Relative performance of alternative policies

	$\tilde{\sigma}$(SF)		$\tilde{\sigma}$($S^{\text{hybrid}}_{25\%}$)		$\tilde{\sigma}$(Reopt)		$\tilde{\sigma}$(CD)		% $\leq \tilde{\sigma}$(Reopt)	
	Avg	Max	Avg	Max	Avg	Max	Avg	Max	$\tilde{\sigma}$(SF)	$\tilde{\sigma}$($S^{\text{hybrid}}_{25\%}$)
	Influence of the drone speed									
$\alpha = 1.0$	1.9	2.8	1.2	2.4*	1.2	2.5	2.1	**3.7**	1.3	57
$\alpha = 2.0$	1.5	2.1	1.1	2.0*	1.1	2.3	1.6	**2.5**	2.8	80
$\alpha = 3.0$	1.3	1.7*	1.1	1.9	1.1	**2.1**	1.4	1.9	5.4	90
	Influence of the drone shortcuts and alternative distance metrics for truck									
L_2 for truck	1.5	2.1	1.1	2.0*	1.1	2.3	1.6	**2.5**	2.8	80
L_1 for truck	1.3	1.8*	1.1	2.1	1.1	**2.4**	1.5	2.1	4.3	82
L_2 for truck, shortcuts	1.4	1.8*	1.1	2.4	1.1	**2.5**	1.5	2.3	6.9	90
L_1 for truck, shortcuts	1.3	1.8*	1.0	1.8	1.0	**2.5**	1.3	2.0	7.0	95
	Influence of parking possibilities									
$\|D\| = 6$	1.5	2.1	1.1	2.0*	1.1	2.3	1.6	**2.5**	2.8	80
$\|D\| = 11$	1.5	2.2	1.1	2.1*	1.1	2.2	1.7	**2.5**	1.6	75
$\|D\| = 16$	1.5	2.2	1.1	2.0*	1.1	2.2	1.7	**2.5**	0.8	73

Note. The *Base* setting is highlighted in grey. The *maximum* (*minimum*) worst observed ratios to the CIOS result for each setting are marked in **bold** (marked with *).

coincided with that of CIOS, since the truck walk did not contain any damaged edges. The picture reverses, when we examine the *observed worst ratios (Max)*. Here, SF performed better than *Reopt* in five settings out of eight. Only when the drone speed equaled that of the truck ($\alpha = 1$), the observed worst ratio of SF was inferior to that of *Reopt*. In case of edge damages in the truck walk of Reopt, the required replannings were quite costly resulting in up to 2.5 times larger makespan (each replanning caused a 1.4 times increase in the makespan, on average). CD had the worst *observed average* and *worst ratios* in most settings compared to SF and *Reopt*. Supplement 4.9.7 contains sensitivity analysis for the case of non-zero delivery times as well as for an alternative objective – *minimize the time of the last delivery*. The relative performance of the algorithms remained similar. Overall, the ratios obtained by minimizing the time of the last delivery exceeded those achieved with the makespan objective. This trend has also been observed in other routing problems, such as the online TSP with and without the requirement to return to the depot (Ausiello et al., 2001). Overall, *Reopt* has a higher competitive ratio than SF in case several damaged edges are present and the drone is fast (see Section 4.4). Table 4.2 confirms this relation in terms of the observed worst ratios.

Table 4.3: Data generation and the number of relevant* damaged edges in *Reopt* in *Base*

	Share of damaged edges in the generated graph		
	20 %	40 %	60 %
% instances with no damaged edges in the initial truck walk	50	24	10
Avg # of relevant* damaged edges per instance	0.6	1.2	1.6
Max # of relevant* damaged edges per instance	4	7	7

Note. *) Relevant damaged edges refer to the damaged edges in the truck walk discovered during the application of *Reopt*. The share of damaged edges in the generated data set is highlighted in grey.

Observed worst ratios in Table 4.2 were, quite as expected, significantly smaller than *competitive ratios* of the examined policies, which are *theoretically possible* worst ratios. For instance, competitive ratios of CD for $\alpha = 1, 2, 3$ are $1 + \frac{|V|-1}{\alpha} = 5, 3, 2.3$, respectively, compared to the *observed worst ratios* 3.7, 2.5, and 1.9. Interesting are possible interpretations of parameter k – the maximal number of damaged edges – which is pivotal for the computation of competitive ratios for SF and *Reopt*. For this, consider the total number of so-called *relevant damaged edges*, which are the damaged edges in the truck walk discovered during the application of a policy. Recall that only relevant damaged edges cause replannings, since the drone can fly over the damage. As Table 4.3 shows for *Reopt* in *Base*, the number of relevant damaged edges never exceeded 7 (out of 22 damaged edges in the graph) and equaled 1.6 on average. Identifying k as the expected number of relevant damaged edges leads to a closer gap between theoretical and observed worst ratios. For instance, for $\alpha = 1, 2, 3$; if $k := 1.6$, competitive ratios of SF are 3.6, 2.3, 1.9, whereas observed worst ratios were 2.8, 2.1, 1.7; the respective observed worst ratios of *Reopt* were 2.5, 2.3, 2.1, whereas the competitive ratio is $2^k \approx 3.0$. The first columns in Table 4.3 report sensitivity analysis for the instance generation: if the share of damaged edges decreases, the share of instances, in which initial *Reopt* solution is optimal, raises fast.

Hybrid policy $\text{S}_{25\%}^{\text{hybrid}}$ achieved the best performance among all tested policies. Table 4.2 shows that $\text{S}_{25\%}^{\text{hybrid}}$ had low *observed average ratios* similar to Reopt, but significantly improved the *observed worst ratios* in all the settings. Interestingly, $\text{S}_{25\%}^{\text{hybrid}}$ also had a better observed worst ratio than SF in the majority of settings, because of the shorter initial surveillance phase.

In addition, Table 4.4 presents the case of a slow drone, which is uncommon in practice but interesting in its own right. Here, both CD and SF performed worse than *Reopt* both in *observed average* and *worst* ratios. Interestingly, drone surveillance may still pay off, since $\text{S}_{25\%}^{\text{hybrid}}$ moderately improved the *observed worst ratios* of *Reopt* in one setting.

Table 4.4: Relative performance of alternative policies in case of a slow drone

	$\tilde{\sigma}$(SF)		$\tilde{\sigma}(\text{S}_{25\%}^{\text{hybrid}})$		$\tilde{\sigma}$(Reopt)		$\tilde{\sigma}$(CD)		% $\leq \tilde{\sigma}$(Reopt)	
	Avg	Max	Avg	Max	Avg	Max	Avg	Max	$\tilde{\sigma}$(SF)	$\tilde{\sigma}(\text{S}_{25\%}^{\text{hybrid}})$
	Influence of the drone speed, the drone is slower than the truck									
$\alpha = 0.25$	3.5	6.3	1.6	3.3*	1.5	3.6	4.2	**13.0**	0	47
$\alpha = 0.50$	2.6	4.3	1.4	3.3	1.4	3.2*	2.9	**7.2**	1	53
$\alpha = 0.75$	2.2	3.3	1.3	3.2	1.3	2.8*	2.4	**4.8**	1	54

Note. The *maximum* worst observed ratios to the CIOS result for each setting are marked in **bold**. The *minimum* worst observed ratios to the CIOS result are marked with *.

Finally, Table 4.5 presents the case where the truck and the drone have different fields of view. We introduced two additional lookout nodes on each *damaged* edge which inhibit the view of the truck, but have no influence on the field of view of the drone. In other words, the field of view of the truck deteriorates. We vary the field of view of the truck in three different settings: the truck encounters the damage exactly in the middle of the original damaged edge (at 50% of the edge length), the truck encounters the damage after having traversed 25% of the edge (at 25% of the edge length), and the original edge (i, j) is partitioned into three equal segments and we randomly place the damage with probability $\frac{1}{3}$ in one of the segments (uniformly at the edge). The smaller field of view of the truck obviously does not affect SF, which relies on the complete initial drone surveillance. Interestingly, the *observed average* ratios stay stable and increase only slightly compared to the base setting with equal fields of views. The increase in the *observed worst ratios* for both policies can be explained by the longer detours of the truck if it encounters a damaged edge compared to the base setting.

4.6.3 Drone energy requirements: Duration of drone sorties

Table 4.6 reports the duration of the drone sorties both excluding and including the waiting time for the truck. Indeed, depending on the application, the drone may have

Table 4.5: Relative performance of *Reopt* and $S^{hybrid}_{25\%}$ with different fields of view for truck and drone

	$\tilde{\sigma}$(SF)		$\tilde{\sigma}(S^{hybrid}_{25\%})$		$\tilde{\sigma}$(Reopt)		% $\leq \tilde{\sigma}$(Reopt)	
	Avg	Max	Avg	Max	Avg	Max	$\tilde{\sigma}$(SF)	$\tilde{\sigma}(S^{hybrid}_{25\%})$
Equal field of view	1.5	2.1	1.1	2.0*	1.1	2.3	3	80
Uniformly at the edge	1.5	2.1	1.1	2.4	1.1	2.4	6	82
At 25% of the edge length	1.5	2.1	1.1	2.4	1.1	2.4	6	82
At 50% of the edge length	1.5	2.1	1.1	2.6	1.2	**2.6**	8	82

Note. The *Base* setting is highlighted in grey. The *maximum* (*minimum*) worst observed ratios to the CIOS result for each setting are marked in **bold** (marked with *).

Table 4.6: Average and maximum drone flight time in surveillance and per drone sortie

	Initial drone surveillance		Drone travel time per sortie			Drone travel + waiting time per sortie		
	SF	$S^{hybrid}_{25\%}$	SF	*Reopt*	$S^{hybrid}_{25\%}$	SF	*Reopt*	$S^{hybrid}_{25\%}$
Influence of the drone speed								
$\alpha = 1.0$	**3717**	172	1539	**1548**	1534	1635	**1689**	1622
	(9113)	(776)	(4632)	**(8186)**	(5870)	(5536)	**(9750)**	(8335)
$\alpha = 2.0$	**1168**	30	742	**769**	761	790	**835**	813
	(4695)	(253)	(2156)	**(4494)**	(3428)	(2452)	**(5404)**	(5220)
$\alpha = 3.0$	**525**	8	493	501	**503**	539	**558**	552
	(2268)	(153)	(1437)	**(2317)**	(2261)	(2035)	**(3364)**	(2871)
Influence of the drone shortcuts and alternative distance metrics for truck								
L_2 for truck	**1168**	30	742	**769**	761	790	**835**	813
	(4695)	(253)	(2156)	**(4494)**	(3428)	(2452)	**(5404)**	(5220)
L_1 for truck	**936**	24	737	**759**	757	798	**834**	817
	(3998)	(327)	(2156)	**(4154)**	(3418)	(2831)	**(4984)**	**(4984)**
L_2 for truck,	**661**	10	435	**454**	453	482	**511**	506
shortcuts	**(2488)**	(160)	(1075)	**(2944)**	(1972)	(1578)	(3746)	**(3944)**
L_1 for truck,	**533**	5	453	466	**467**	510	**526**	523
shortcuts	**(1782)**	(160)	(1087)	**(3680)**	(1784)	(1682)	**(4992)**	(2730)
Influence of parking possibilities								
$\|D\| = 6$	**1168**	30	742	**769**	761	790	**835**	813
	(**4695**)	(253)	(2156)	(**4494**)	(3428)	(2452)	(**5404**)	(5220)
$\|D\| = 11$	**1150**	33	698	**713**	709	732	**759**	745
	(**3676**)	(294)	(2156)	(**2971**)	(2442)	(2259)	(**4954**)	(3564)
$\|D\| = 16$	**1135**	33	672	**681**	678	702	**714**	703
	(**3554**)	(294)	(2180)	(2248)	(2228)	(2480)	(**4954**)	(3266)

Note. The *Base* setting is highlighted in grey. For each setting, the *first* (*second*) row of values *without with*) parenthesis are the *average* (*maximum*) times. The largest values for each setting are marked in **bold**.

to hover while waiting for the truck to prevent theft. Interestingly, as predicted by the theoretical analysis, drone sorties may turn long in Reopt, when the truck has to take a detour to reach the next landing location. As a result, maximal observed drone sorties in *Reopt* may take even longer than the initial drone surveillance phase in SF, especially if the truck follows the L_1 metric with drone shortcuts (maximum drone travel time per sortie in *Reopt* of 3,680 vs. maximum surveillance time in SF of 1,782). Drone sorties in *Reopt* get longer if the number of rendezvous locations $|D|$ is small, the drone speed is small, or the truck follows the L_1 metric. Overall, drone sorties in the delivery phase are the shortest in SF, which is achieved at the cost of a long initial drone surveillance time. The hybrid policy $S^{hybrid}_{25\%}$ again offers a good compromise for drone energy consumption: It significantly reduces initial drone surveillance time compared to SF, making it compatible with most drones' battery capacities, while the time for drone sorties is generally shorter than in *Reopt*. Since all the examined policies require long drone sorties, efficient energy management is essential. This may involve installation of battery swapping facilities in the field, enforcing a short distance between the drone and the vehicle at all times, as well as a preference for longer-range drones at least for the initial surveillance phase.

4.7 Discussion and outlook

In this paper, we investigate the traveling salesman problem with a truck and a drone under incomplete information (TSP-DI), motivated by deliveries of emergency supplies under unknown road conditions, which are often faced by disaster relief teams. The paper computes *competitive ratios* (CRs) for several policies for TSP-DI, including the widespread online *reoptimization policy* (Reopt). CRs reveal the worst-case performance of these policies. Competitive analysis is especially relevant and important in the context of disaster relief, where the worst outcomes have to be avoided. We conduct a parametric analysis and differentiate problem classes according to the number of damaged edges $k \in \mathbb{N} \cup \{0\}$, the factor by which the drone is faster than the truck $\alpha \in \mathbb{R}^+$, as well as the number of required deliveries $|V| \in \mathbb{N}$.

Our analysis reveals that *Reopt* performs poorly in realistic worst-case instances if several damaged edges are present, moreover, its CR increases exponentially with k. *Reopt* fails if several alternative routes to a required destination exist and all but a few routes are impassable, so that the truck gets easily trapped in a damaged region. For instance, this situation is typical in inundated urban areas. Our analysis underscores the significance of a rather counter-intuitive decision – to delay the start of emergent deliveries, so that the drone can examine the most relevant road segments of the truck tour. One such policy is surveillance first (SF), which has a better CR than *Reopt* if the road network is severely damaged. In experiments on conventional data sets, we illustrate that *worst observed* CRs indeed improve in almost all settings if we use SF. Nevertheless, *Reopt* outperforms SF in all the settings, on *average*. We further show that hybrid policies with limited initial drone surveillance, as in $S_{25\%}^{hybrid}$, may already be sufficient to improve the *worst observed* ratios significantly while maintaining the good average performance on par with *Reopt*. The main limitation of our computational experiments is the small size of the instances caused by computationally intensive replanning iterations in *Reopt* and SF. The validation of the observed trends is a topic for future research. Although the question about a policy with the best possible CR remains open, we narrow its properties analytically and prove that such a policy should include detours for the purpose of surveillance if the drone is fast ($k > 1$ and $\alpha > \frac{k+1}{k-1}$) and the number of the required deliveries is large ($|V| > (k + \alpha + 1)$). We also compute the CR for the conservative delivery policy, in which the drone performs all the deliveries in return flights from the depot.

A number of questions remain for future research. Firstly, both *Reopt* and SF rely on reoptimization, in which an NP-hard truck-and-drone routing problem is solved repeatedly. Therefore, suitable fast heuristics are required. Secondly, the question of an optimal policy remains open. Depending on a specific disaster relief application, the robustness of the policy (such as its worst-case performance, eventually restricted to some interval or a set of probable scenarios) may be more important than its expected average performance. Therefore, respective types of optimality analysis are important for disaster relief management. Thirdly, future research has to investigate extensions of TSP-DI, including the influence of vehicle fleet characteristics (e.g., several drones and/or several trucks), of the limited energy of the drone, of information transmission mechanisms between the vehicles as well as of deadlines/releases and priorities of the deliveries. The authors confirm that the data sets used in the computational experiments are available in the Online Supplement.

4.8 Acknowledgements

The authors would like to acknowledge the comments and suggestions from the Associate Editor and the two anonymous referees whose inputs have substantially improved the quality of the article. This work was supported by the Deutsche Forschungsgemeinschaft (DFG - German Research Foundation) as part of the SuPerPlan project #414225725.

4.9 Supplemental material

4.9.1 Supplement: Proof of Lemma 14

In this section, we repeat the technical Lemma 14 and provide the complete proof.

Lemma 18. *For any $|V| \geq 2$, $\alpha > 0$ and $k \in \mathbb{N} \cup \{0\}$, we can always construct an instance where the set of locations V is divided into two subsets V_1 and V_2 and the damaged arcs are distributed such that in a complete information optimal solution (CIOS) the drone delivers to all $v \in V_1$, the truck to all $v \in V_2$, and both finish at the same time T^*.*

Proof. Figure 4.5 provides an illustration. There are two *loops* on opposite sides of the depot v_0, consisting of two or more *intact* edges each. Address v_1 is located in the middle of the left *loop*, and the remaining addresses $V \setminus \{v_1\}$ are placed in the middle of the right *loop* of total length T_r. Recall that the speed of the truck is normalized to 1 and the speed of the drone equals α. The length of the left *loop* is $T_l = \alpha \cdot T_r$. The rest of the graph is not visualized and can be chosen arbitrarily provided that the left and the right *loops* remain shortest paths from the depot v_0 to v_1 and from the depot v_0 to all addresses $V \setminus \{v_1\}$ in I^*, respectively. All k damaged edges can be located arbitrarily in that hidden portion of the graph. Obviously, the vehicles travel $\frac{T_l}{\alpha} = T_r$ and T_r time, respectively, and return at the same time to the depot. □

4.9.2 Supplement: Further unfortunate examples for Reopt

If either ($|V| = 2$ and $\alpha \geq 1$) or ($|V| = 1$ and $\alpha \leq 1$), the proof of $\sigma(A^{Reopt}) \geq 2^k$ in Theorem 4 relies on our technical assumption that if the truck in the PS faces alternative paths of the same length, it always chooses the worst possible path. In particular, it prefers a not yet completely investigated path to the path of the same length known to be intact. This assumption may appear counterintuitive. Therefore, we design further unfavorable examples for Reopt, which do not rely on the technical assumption above and they also result in a lower bound on the competitive ratio that is exponential in k. The example in Lemma 19 covers the case of ($|V| = 2$ and $\alpha \geq 1$) and the one in Lemma 20 covers the case of ($|V| = 1$ and $\alpha < 1$), respectively.

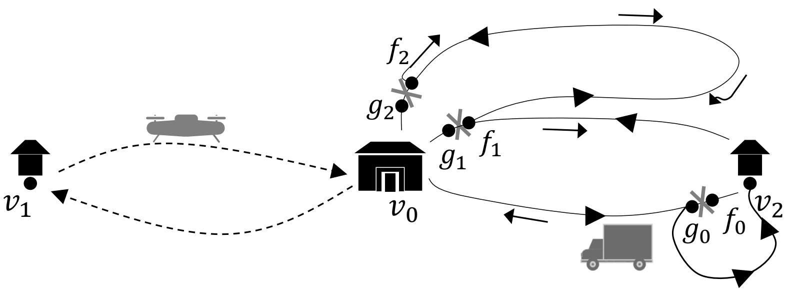
Lemma 19. *For any $|V| = 2$ and $\alpha \geq 1$, the competitive ratio of the optimistic online reoptimization policy is $\sigma(A^{Reopt}) \geq 2^{k-1}$.*

Proof. We *relax* the assumption that PS always chooses the worst possible tie-breaker.

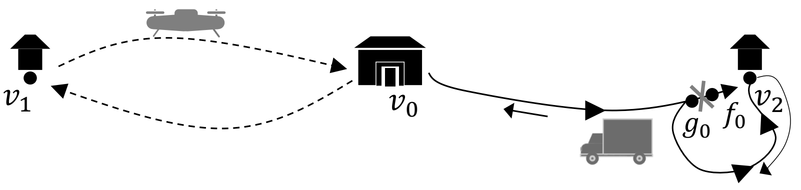
The case of $k = 1$ is trivial. In the following, we will discuss two cases: ($k \geq 2$, $\alpha \neq 1$) and ($k \geq 2$, $\alpha = 1$).

For $k \geq 2$ and $\alpha \neq 1$, consider the instance in Figure 4.10. Nodes v_1 and v_2 are located at the mid-points of two *loops*: the left *loop* (of length $\alpha T'$) and the lower right *loop* (of length T'), respectively. The distances are selected such that the truck and the drone would complete their trips at the same time through the left and lower right *loop*, respectively, if all edges were intact (cf. Lemma 14). There is a damaged edge (g_0, f_0) in

Figure 4.10: An unfortunate example for *Reopt* with $k = 3$ and $|V| = 2$



(a) Policy solution (PS)



(b) Complete information optimal solution (CIOS)

Note. The truck in PS plans to deliver to address v_2 along the lower *loop* via g_0 and f_0 to the right of the depot. It encounters damaged edge (f_0, g_0) close to v_2 and takes a detour. At v_2, the truck decides to return to the depot along the yet unexplored upper part of the right lower *loop* of length $\frac{T'}{2}$, since it would take $\frac{T'}{2} + \xi, \xi > 0, \xi \to 0$, to return the same way back to the depot. At the end of its travel, the truck encounters damaged edge (g_1, f_1) and decides to drive to the depot along the upper *loop*. But because of another damaged edge (f_2, g_2), it has to travel all the way back.

the vicinity of v_2 and there is an additional intact direct edge between g_0 and v_2, which allows for a detour. Let the truck travel time between depot v_0 via g_0 over the detour to v_2 be $\frac{T'}{2} + \xi$, where $\xi > 0$ is very small.

The remaining edges in the instance are constructed as explained in Theorem 4 with the difference that there are just $(k-1)$ *loops* on the right side of the depot, including the lower right-hand *loop*. Similar to Theorem 4, each such *loop* has a damaged edge $(f_i, g_i), i \in \{1, \ldots, k-1\}$, located close to depot v_0, such that the truck travel time from f_i over g_i to v_0 equals ξ. Also here, the ith *loop* shares the path from depot v_0 via g_{i-1} to f_{i-1} with *loop* $(i-1)$ and is connected via the edges (f_i, g_i) and (g_i, v_0) to the depot, for all $i \geq 2$.

Obviously, the makespan of CIOS T^* is bounded by the following interval: $T' \leq T^* \leq T' + 2\xi$. Observe that $T' \leq T^*$, since the makespan of CIOS cannot be smaller than the time it takes the fastest vehicle to deliver to v_1 and v_2 and return to v_0 assuming all the edges are intact (cf. Lemma 14). Moreover, in a feasible solution with makespan $T' + 2\xi$, the drone delivers to v_1 and, in parallel, the truck delivers to v_2 taking the lower *loop* via the direct detour-edge (g_0, v_2) there and back.

In PS, the truck tries to deliver to v_2 traveling the walk $(v_0, g_0, f_0, \overline{v_2}, f_1, g_1, v_0)$, since it does not know that specific edges are damaged. Because of the damage at (g_0, f_0), the truck takes the detour via the direct edge (g_0, v_2), provided ξ is sufficiently small ($\xi \to 0$). At v_2, it has a choice to travel all the way (v_2, g_0, v_0) back or to proceed along (v_2, f_1, g_1, v_0). The latter path takes $\frac{T'}{2}$ and is shorter than the former, which has the duration of $(\frac{T'}{2} + \xi)$. The proof proceeds along the same lines as in Theorem 4.

For $k \geq 2$ and $\alpha = 1$, we have to modify the example. Indeed, otherwise, the Reopt-planner would rather send the truck to the left and the drone to the right, since we already see from the depot that the left *loop* is intact, but we have uncertainty whether all the edges are intact for the right-hand lower *loop*, and both *loops* are of the same length. To prevent this, we must introduce two additional lookout nodes in the left *loop* such that both *loops* are indistinguishable for the ex-ante planner. □

Lemma 20. *For any $|V| = 1$ and $\alpha < 1$, the competitive ratio of the optimistic online reoptimization policy is $\sigma(A^{Reopt}) \geq 2^{k-1}$.*

Proof. If $|V| = 1$ and $\alpha < 1$, we simply remove node v_1 in the example in Figure 4.10. PS remains the same as described in Lemma 19, since initially the truck, as the faster vehicle, will try to deliver to v_2 driving along $(v_0, g_0, f_0, v_2, f_1, g_1, v_0)$.

In CIOS, the makespan remains bounded by the interval $T' \leq T^* \leq T' + 2\xi$. In a feasible solution, the truck delivers to v_2 taking the lower *loop* via the direct detour-edge (g_0, v_2) there and back, hence, $T^* \leq T' + 2\xi$. Further, the makespan of CIOS cannot be smaller than the time it takes the fastest vehicle to deliver to v_2 in a return trip from the depot v_0 assuming all the edges are intact, hence, $T^* \geq \min\{T', \frac{T'}{\alpha}\}\} = T'$. □

4.9.3 Supplement: On the unfortunate example for *Reopt* in Theorem 4

Let's recall the description of the unfavorable instance with $V = \{v_1, v_2, v_3\}$ and $k \in \mathbb{N} \cup \{0\}$ from Theorem 4, illustrated schematically in Figure 4.6. Nodes v_2 and v_3 are located on the lower right-hand *loop* from the depot of length T'. They are located on opposite sides at a distance of ξ from the middle of the *loop*, where ξ is a very small number. Node v_1 is positioned in the center of the left-hand *loop*. This *loop* has the length of $\alpha \cdot T'$. There are, in total, k *loops* to the right side of the depot, each having a damaged edge $(f_i, g_i), i \in [1, k]$, located close to depot v_0, such that the truck travel time from f_i over g_i to v_0 equals ξ. The ith *loop* shares the path from depot v_0 via g_{i-1} to f_{i-1} with *loop* $i - 1$ and connects through the edges (f_i, g_i) and (g_i, v_0) to the depot, for all $i \geq 2$. Further, assume that the truck travel time on each *loop* $i \geq 2$, starting and finishing at the depot if all the edges were intact, is $2^{i-2}T' - \delta_i$ and $\delta_i > 0$ are monotonically increasing. Recall that the truck travel time is T' for the first *loop*.

Lemma 21. *Let an unfavorable instance with $V = \{v_1, v_2, v_3\}$ and $k \in \mathbb{N} \cup \{0\}$ be given as presented in Theorem 4 . Let edges (f_i, g_i) and edges (v_0, g_i) have lengths of $\frac{\xi}{2}$ for $i \in [1, k]$. If $\xi < \min\{\frac{\alpha T'}{4}, \frac{T'}{4\alpha+5}\}$, then the makespan of CIOS is $T' + 2\xi$.*

Proof. Let's prove that the makespan of CIOS equals $T^* = T' + 2\xi$. In one such possible solution, the drone delivers to v_1 and the truck delivers to v_2 and to v_3 taking the lower *half-loop* there and back (see Figure 4.6a).

Let's observe that the drone can launch from the truck at nodes v_0, v_1, v_2, v_3, g_i, or f_i, $i \in [1, k]$.

We will prove by enumerating possible feasible solutions and showing that their makespan cannot be lower than $(T' + 2\xi)$. First, observe that any feasible solution, in which the truck delivers to (or simply visits) node v_3 or node f_i cannot have a lower objective value than $T' + 2\xi$. Similarly, no solution, in which the truck delivers to (or simply visits) *both* nodes v_1 and v_2, can have a better makespan either. Indeed, the truck needs at least as much time as it takes to traverse the minimum duration tour from the depot over nodes v_1 and v_2 back to the depot, which equals $\alpha T' + T' - 2\xi > T' + 2\xi$ because $\xi < \frac{\alpha T'}{4}$.

In the remaining cases, the drone necessarily delivers to v_3, the truck does not visit nodes v_3 and $f_i, i \in [1, k]$, and the truck visits at most one node from v_1 and v_2. Let's enumerate the following possibly overlapping cases:

(i) First, consider complete information solutions $s \in G_1$, in which the truck does not visit v_1. Then, the drone has to deliver to v_1 in an operation that has its start and end at nodes that are distinct from v_1. The makespan of the solutions in G_1 cannot be less than the following computation:

- the drone delivers to v_1 in a return flight from v_0 first, which takes time T' (observe that v_0 is the closest node in which the drone can meet the truck to take another package, moreover, any shortest path of the drone from v_1 to $v \in L \setminus \{v_1\}$ contains v_0),
- then the drone travels there and back to v_3 with the *speed of the fastest vehicle*, which cannot take more time than $\min\{T' - 2\xi, \frac{T'-2\xi}{\alpha}\}$.

We obtain that:

$$T(s \in G_1) \geq T' + \min\{T' - 2\xi, \frac{T' - 2\xi}{\alpha}\} > T' + 2\xi. \tag{4.11}$$

In the last transformation, we used the relation $\xi < \frac{T'}{4\alpha+5}$, which implies that $T' - 2\xi > \xi(4\alpha + 5) - 2\xi$, and the nonnegativity of α and ξ.

(ii) Let the truck visit v_1. If the truck visits v_1, then it must visit v_2 and v_3. In this case, the makespan is the maximum of $\alpha T'$ (the time that the truck needs to visit v_1 in a return trip from v_0) and $\frac{T'-3\xi}{\alpha} + \frac{T'-2\xi}{\alpha}$ (the time, the drone needs to deliver to v_3 and v_2 from the closest possible launch-return points, which are g_1 and v_0, respectively).

Consider two cases:

- If $\alpha \geq \left(1 + \frac{2}{3}\right)$ and using the fact that $\xi < \frac{T'}{(4\alpha+5)} < \frac{T'}{3}$, then the makespan of the resulting solution cannot be smaller than

$$\max\{\alpha T', \frac{T' - 3\xi}{\alpha} + \frac{T' - 2\xi}{\alpha}\} \geq \alpha T' \geq T' + \frac{2T'}{3} > T' + 2\xi. \tag{4.12}$$

- If $\alpha < \left(1 + \frac{2}{3}\right)$ and using the fact that $\xi < \frac{T'}{(4\alpha+5)} < \frac{T'(2-\alpha)}{4\alpha+5}$, then the makespan of the resulting solution cannot be smaller than

$$\max\{\alpha T', \frac{T' - 3\xi}{\alpha} + \frac{T' - 2\xi}{\alpha}\} \geq \frac{T' - 3\xi}{\alpha} + \frac{T' - 2\xi}{\alpha} =$$
$$T' + \frac{T'(2 - \alpha)}{\alpha} - \frac{5\xi}{\alpha} > T' + \frac{(4\alpha + 5)\xi}{\alpha} - \frac{5\xi}{\alpha} > T' + 2\xi. \tag{4.13}$$

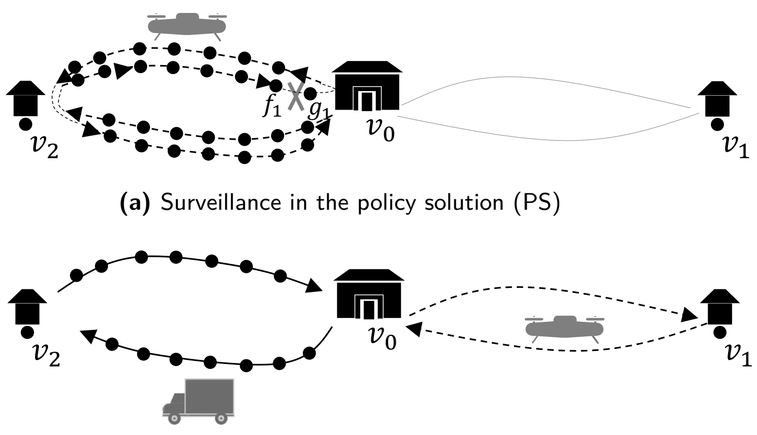
□

4.9.4 Supplement: Further unfortunate examples for SF

In Lemma 17, we provided a lower bound for the competitive ratio of the surveillance-first policy (SF) in the most relevant case, where the drone is faster than the truck and deliveries have to be made to more than two addresses. In the following, we provide additional worst-case instances for SF in the general case of $|V| > 1$ or $\alpha \leq 1$, which rely on the assumption that the policy always chooses the worst possible path, when it faces alternative paths of the same length.

Lemma 22. *If $|V| > 1$ or $\alpha \leq 1$, the competitive ratio of SF $\sigma(A^{SF}) \geq \frac{k+1+\alpha}{\alpha}$.*

Figure 4.11: An unfortunate example for SF with $k = 1$ and $|V| = 2$



(a) Surveillance in the policy solution (PS)

(b) Delivery in a policy solution (PS) and complete information optimal solution (CIOS)

Note. In the initial delivery plan $s^{\diamond,d,0}$ of PS, the truck travels along the inner *loop* to the left of the depot. So the drone in PS has to examine the edges of this inner *loop*, but discovers a damaged edge in the end of this initial surveillance sortie. As a result, the drone in PS has to proceed with surveying the complete replanned truck walk along the outer *loop* to the left of the depot. Afterward, the truck and the drone can deliver packages. The final delivery plan in PS coincides with CIOS.

Proof. If $|V| > 1$, there are instances, such as the one presented in Figure 4.11, where the truck in CIOS performs at least one delivery notwithstanding the value of parameter α (cf. Lemma 14).

We consider the example in Figure 4.11 with $k = 1$ and $|V| = 2$ constructed as described in Lemma 14, in which the drone in CIOS needs T^* units of time to deliver to address v_1, and the truck needs T^* units to deliver to address v_2. Furthermore, there are four paths of equal lengths and very low visibility connecting v_0 and v_2. This is achieved by placing many lookout nodes along the paths. Thus, the drone in PS, basically, has to traverse the whole path to check whether it is passable. Since the four shortest paths between v_0 and v_2 have equal lengths, a possible delivery route of the truck in PS will initially go through the paths in the inner *loop*. Assume there is a damaged edge (f_1, g_1) at the very end of the drone's surveillance flight through the inner *loop*, such that the drone reaches f_1 at time θ_1 close to $\frac{T^*}{\alpha}$. The truck in the re-planned delivery trip travels along the outer *loop*. The total surveillance time, including the time to examine edges in the updated truck walk and the delivery time after the surveillance is completed, converges to $\frac{2T^*}{\alpha} + T^*$. Therefore, $\sigma(A^{SF}) \geq \frac{2+\alpha}{\alpha}$ for $k = 1$.

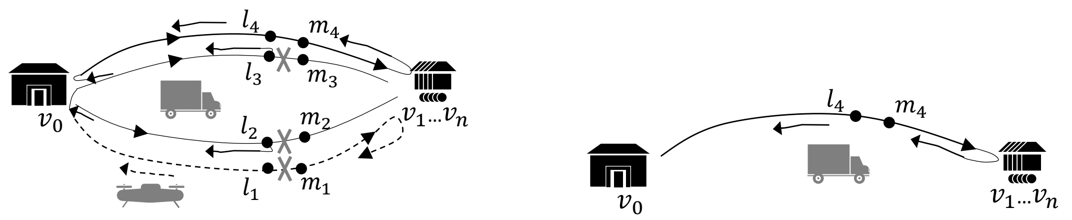
We can construct examples for $k \geq 2$ in a similar manner using $(k+1)$ low-visibility *loops* of the same length that start and end at v_0 and where v_2 is in the middle of these *loops*.

For $|V| = n > 2$, we place all delivery addresses $v \in \{v_2, ..., v_n\}$ at the location of node v_2 as an agglomeration. If $|V| = 1$ and $\alpha \leq 1$, we construct an example where node v_1 is dropped in Figure 4.11. In case the truck and the drone are equally fast ($\alpha = 1$), we use our technical assumption that a possible SF solution will require the truck to perform the only delivery. □

4.9.5 Supplement: Proof of Theorem 7

We proved that the competitive ratio of SF is $1 + \frac{k+1}{\alpha}$ if either $|V| > 1$ or $\alpha \leq 1$. Thus, we look for an instance I with a larger ratio $\frac{A(I)}{OPT(I^*)}$ for any policy (algorithm) A without

Figure 4.12: Example of the benefits of drone surveillance with $k = 3$



(a) A possible policy solution (PS) without surveillance

(b) A possible complete information optimal solution (CIOS)

Note. It is important to discard some impassable paths early on due to the lengthy detours they would otherwise cause for the truck.

drone surveillance.

Proof. For any $k > 1$, $\alpha > \frac{k+1}{k-1}$, and $|V| > (k+1+\alpha)$, we can construct an example similar to that in Figure 4.12 with lookout nodes $l_j \in L \setminus D$ and $m_j \in L \setminus D$, and edges $(v_0, l_j), (l_j, m_j), (m_j, v_i)$ for $j \in [1, k+1], v_i \in V$. All delivery nodes belong to the same agglomeration. Some edge $(l_{k'}, m_{k'}), k' > 1$, is intact and $(k-1)$ remaining edges $(l_j, m_j), k > 1, k \neq k'$, as well as edge (l_1, m_1) are damaged, but this information is initially unknown.

Paths (v_0, l_j, m_j, v_i), $j > 1, v_i \in V$, have equal lengths of $\frac{T'}{2}$, whereas the length of each path (v_0, l_1, m_1, v_i) is $\frac{T'}{2} - \epsilon$. For this purpose, we set the first edge $c^t(v_0, l_1)$ to be infinitesimally smaller by $\epsilon > 0$ than $c^t(v_0, l_j)$ for $j > 1$, whose edge lengths are all equal. As for the remaining edges, we keep the lengths of (m_j, v_i) equal and the lengths of (l_j, m_j) equal for any $j \in [1, k+1], v_i \in V$.

The makespan of CIOS is $T^* = T'$, provided ϵ is sufficiently small to make the drone-only deliveries unattractive, e.g., $\epsilon < T' \frac{k+1}{2(k+1+\alpha)}$. Indeed, in a feasible solution, the truck delivers to all addresses $v_i \in V$, using the undamaged path $(v_0, l_{k'}, m_{k'}, v_1)$ from the depot there and back within the time of $T^* = T'$. Any feasible solution, in which the truck visits one or several nodes $v_i \in V$, has the makespan of at least T'. In the remaining solutions, the drone delivers to all $v_i \in V$ starting from v_0, because no launch is possible in lookout nodes l_j and m_j. The makespan of such drone-only deliveries equals $|V| \cdot \frac{T'-2\epsilon}{\alpha} > (k+1+\alpha) \cdot \frac{T'-2\epsilon}{\alpha} > (k+1+\alpha) \cdot \frac{T'}{\alpha} - T' \cdot \frac{k+1}{\alpha} = T'$; here we used relations $|V| > (k+1+\alpha)$ (from the definition of the theorem) and $\epsilon < T' \frac{k+1}{2(k+1+\alpha)}$.

In PS, the drone may start or end its delivery operations only in nodes v_0 and $v_i \in V$, because l_j's and m_j's are lookout nodes ($l_j, m_j \in L \setminus D, \forall j \in [1, k+1]$). If the drone launches in v_0 to deliver to some node $v_i \in V$, then it always takes the shortest path (v_0, l_1, m_1, v_i). The same is true (in the reverse direction), if the drone returns to the depot v_0 after delivering to v_i. The case is obvious, if the drone delivers to v_i using some node $v_{i'} \in V$, as a launch or return node. To sum up, the drone will not collect information on the damaged edges $(l_j, m_j), j > 1$, in policies without drone surveillance.

First, we consider delivery policies where the truck performs at least one delivery. Since paths (v_0, l_j, m_j, v_i) are identical for $j > 1, v_i \in V$, no delivery policy can distinguish between them. Therefore, for any such delivery policy, we can construct an instance where the truck has to perform $(k-1)$ detours before it can deliver packages using the intact path $(v_0, l_{k'}, m_{k'}, \dots), k' \in [2, k+1]$. We let the length of each detour (v_0, l_j, v_0) be δT^* for some $0 < \delta < 1$ and $j > 1$. Then, the truck may require at least $(\delta(k-1)+1)T^*$ units of time to deliver its package(s). By shifting edges (l_j, m_j) to the right so that $1 > \delta > \frac{k+1}{(k-1)\alpha}$, we obtain a worse ratio than we do for the surveillance-first policy.

Now, consider policies without surveillance, in which the truck does not perform any deliveries and does not visit any $v_i \in V$. As reasoned above, in this case the drone can only launch from node v_0. Since $|V|$ is sufficiently large, i.e., $|V| > (k+1+\alpha)$, the resulting ratio for these drone delivery policies is worse than for the surveillance-first policy, see Theorem 6. □

4.9.6 Supplement: Additional details for the hybrid policy

Policy $\mathsf{S}^{\text{hybrid}}_{25\%}$ is a *hybrid* approach that combines the advantages of both *Reopt* and SF policies. As with SF, the new policy $\mathsf{S}^{\text{hybrid}}_{25\%}$ consists of a surveillance phase and a delivery phase. The basic idea is that the surveillance phase is shorter compared to SF, and the delivery phase begins with some pre-acquired knowledge about the condition of certain roads.

Algorithm 2 provides the pseudocode of $\mathsf{S}^{\text{hybrid}}_{25\%}$. Prior to the surveillance in $\mathsf{S}^{\text{hybrid}}_{25\%}$, an initial delivery plan $s^{\diamond,d,0}$ is generated (line 2), which is an optimal solution under the assumption that all the edges with unknown status are intact. Let $V^{t,0}, E^{t,0}$ be the nodes and edges of the truck walk in the solution $s^{\diamond,d,0}$, respectively. The sets $V^{t,0}, E^{t,0}$ and the matrix $(c^d_{ij})_{i,j\in V^{t,0}}$ are then provided as input to a standard solver, to solve the integer linear programming (ILP) formulation *Adjusted TSP*. Thereby, $(c^d_{ij})_{i,j\in V^{t,0}}$ contains for each pair of nodes $i, j \in V^{t,0}$ of the original graph the distance of the shortest connecting path that the drone is allowed to take, determined by whether the drone is allowed to take shortcuts or not. The binary decision variable x_{ij} equals 1 if the drone travels from $i \in V^{t,0}$ to $j \in V^{t,0}$ (taking a shortest path, as described above), and 0 otherwise. The integer variable $u_i \in \mathbb{N}$ denotes the position of each visited node $i \in V^{t,0}$ in the drone path.

Adjusted TSP

$$\textit{Minimize} \sum_{i\in V^{t,0}} \sum_{j\in V^{t,0}} c^d_{ij} x_{ij} \tag{4.14}$$

subject to

$$\sum_{k\in V^{t,0}, k\neq i} x_{ki} + \sum_{k\in V^{t,0}, k\neq j} x_{kj} \geq 1 \qquad \forall (i,j) \in E^{t,0} \tag{4.15}$$

$$\sum_{j\in V^{t,0}, j\neq v_0} x_{v_0,j} \geq 1 \tag{4.16}$$

$$\sum_{j\in V^{t,0}, j\neq i} x_{ij} = \sum_{j\in V^{t,0}, j\neq i} x_{ji} \qquad \forall i \in V^{t,0} \tag{4.17}$$

$$|V^{t,0}| \cdot x_{ij} + u_i - u_j \leq |V^{t,0}| - 1 \qquad \forall i,j \in V^{t,0}, i \neq j \tag{4.18}$$

$$x_{ij} \in \{0,1\} \qquad \forall i,j \in V^{t,0} \tag{4.19}$$

$$u_i \in \mathbb{N} \qquad \forall i \in V^{t,0} \tag{4.20}$$

The *Adjusted TSP* ILP describes a variant of the classical TSP problem, in which not every node must be visited, but instead, for every arc in $(i,j) \in E^{t,0}$, at least one of its adjacent nodes $i, j \in V^{t,0}$ must be visited, see constraints (4.15), in the shortest possible time (4.14). It is guaranteed that the depot is part of the tour (4.16); (4.17) are flow constraints and (4.18) are the classical Miller-Tucker-Zemlin constraints for subtour elimination. Let $T^{\diamond,d,0}$ be the resulting optimal objective value.

Algorithm 2 $\mathsf{S}^{\text{hybrid}}_{25\%}$

```
 1: # Phase 1: surveillance
 2: # Preprocessing: compute $s^{\diamond,d,0}, V^{t,0}, E^{t,0}, c_{max}$,
 3: $c_{curr} \leftarrow 0, v^0_{curr} \leftarrow$ copy of $v_0, replan \leftarrow True, k \leftarrow 0$
 4: while $replan = True$ and $c_{max} - c_{curr} > 0$ do
 5:     $replan \leftarrow False$
 6:     Solve orienteering problem for given $k$ with output $s^{\diamond,s,k}$
 7:     Start the surveillance tour $s^{\diamond,s,k}$ for the drone, thereby
 8:     for each traversed edge in $s^{\diamond,s,k}$ do
 9:         Update the set of known edges in $E$ and the current time $c_{curr}$
10:         if new damage is observed in the planned truck walk of $s^{\diamond,d,k}$ then
11:             $replan \leftarrow True$, $k \leftarrow k+1$, $v^k_{curr}$ := current position of drone
12:             Set penalty factor $w^k_{ij}$ for all unknown edges in $(i,j) \in E$
13:             Recompute truck-and-drone delivery tour $s^{\diamond,d,k}_{pen}$
14:             Set $V^{t,k}, E^{t,k}$ := nodes and edges in truck tour of $s^{\diamond,d,k}_{pen}$
15:             $c_{max} \leftarrow c_{max} - c_{curr}$, $s^{\diamond,d,k} \leftarrow s^{\diamond,d,k}_{pen}$
16:             break and go to line 4
17:         end if
18:     end for
19: end while
20: # Phase 2: delivery
21: $replan \leftarrow True$, $s^{\diamond,d,0} \leftarrow s^{\diamond,d,k}, k \leftarrow 0$
22: while $replan = True$ do
23:     $replan \leftarrow False$
24:     Start truck-and-drone delivery tour $s^{\diamond,d,k}$, thereby
25:     for each traversed edge in $s^{\diamond,d,k}$ do
26:         Update the set of known edges in $E$
27:         if new damage is observed in the truck walk $s^{\diamond,d,k}$ then
28:             $replan \leftarrow True$, $k \leftarrow k+1$
29:             Set penalty factor $w^k_{ij}$ for all unknown edges in $(i,j) \in E$
30:             Recompute optimal truck-and-drone tour $s^{\diamond,d,k}_{pen}$ given current positions of truck and drone, for the
                currently unvisited customers
31:             $s^{\diamond,d,k} \leftarrow s^{\diamond,d,k}_{pen}$
32:             break and go to line 22
33:         end if
34:     end for
35: end while
```

In lines 3-19 of Algorithm 2, blue$\mathsf{S}_{25\%}^{\mathsf{hybrid}}$ performs surveillance. Unlike SF, which necessitates visiting *all* truck nodes as quickly as possible – effectively modeling the surveillance route construction as a Traveling Salesman Problem (TSP) –, the idea of $\mathsf{S}_{25\%}^{\mathsf{hybrid}}$ is to limit the total surveillance time by a time limit c_{max}, at the cost of not acquiring knowledge of the complete truck tour. This time limit is given as a fixed *portion of the total required time to check all truck edges of the initially planned tour*, $T^{\diamond,d,0}$. On a separate dataset of 100 graphs with 20 different distributions of damaged edges per graph (thus, 2000 instances) of the *Base* setting, constructed as described in Section 4.6.1, we compared the results for different durations of the surveillance tour: Table 4.7 reports the results for the hybrid policy with 25%, 50%, and 75% of $T^{\diamond,d,0}$ as surveillance time. The results reveal that 25% of the full surveillance time is already enough to provide the best results, thus, the name of the main policy, $\mathsf{S}_{25\%}^{\mathsf{hybrid}}$, and the selection of $c_{max} = 0.25 \cdot T^{\diamond,d,0}$.

Within time c_{max}, the *most important* truck nodes should be visited. Given those requirements, the construction of surveillance routes of $\mathsf{S}_{25\%}^{\mathsf{hybrid}}$ can be appropriately modeled as an *orienteering problem*, see (4.21)-(4.27), where the objective is to maximize the cumulative 'prizes' gathered from visiting nodes from the truck walk of the currently planned delivery tour within the time limit. The prize p_i for each node i of the truck walk equals the number of its adjacent edges with unknown status in E.

First, an initial surveillance tour $s^{\diamond,s,0}$ is found by solving the orienteering problem (4.21)-(4.27) for truck nodes $i \in V^{t,0}$ of the initially planned delivery tour $s^{\diamond,d,0}$ with time limit c_{max}. Then the drone traverses $s^{\diamond,s,0}$. The elapsed time c_{curr} is tracked, and the set of unknown edges of the graph is continuously updated. In case the drone encounters no edge damage in the planned truck walk of $s^{\diamond,d,0}$, the surveillance phase terminates after time $T^{\diamond,s,0} := T(s^{\diamond,s,0})$. Otherwise, at each encountered damaged truck edge, the drone stops at its current node v^k_{curr}, and the planned truck-and-drone delivery tour is updated to $s^{\diamond,d,k}_{pen}$. In the computation of $s^{\diamond,d,k}_{pen}$, edges $(i,j) \in E$ of unknown status are penalized by multiplying their truck-traversal time by a *penalty factor* w^k_{ij}. This is because these edges may potentially lead to costly detours if included in the route and are later found to be damaged. In our implementation of $\mathsf{S}_{25\%}^{\mathsf{hybrid}}$, we set the penalty factor for every damaged edge, in every iteration, to the same value of 1.3, determined by calibration on the separate test dataset described above. We refer to Table 4.7 for details on the calibration for the makespan objective. Again, the orienteering problem is solved for computing a surveillance tour starting from v^k_{curr} that maximizes the collected prizes by visiting truck nodes of the new delivery plan $s^{\diamond,d,k}_{pen}$ given the updated prizes for each node and the updated remaining time c_{max}. Then, the drone continues surveillance according to its new route $s^{\diamond,s,k}$, and this procedure is repeated until no more edge damages in the planned truck delivery walk are found. The time of the surveillance phase is then the time c^{curr} as last updated.

In the following, $\mathsf{S}_{25\%}^{\mathsf{hybrid}}$ begins with the truck-and-drone delivery (lines 20-34 in Algorithm 2). Initially, truck and drone follow the delivery tour $s^{\diamond,d,0} := s^{\diamond,d,k}$ that was lastly planned in the surveillance phase. If they encounter no damaged truck edge during the delivery, the time of the delivery phase is $T^{\diamond,d,0} = T(s^{\diamond,d,0})$. Otherwise, at each damaged truck edge that is encountered, a new delivery plan $s^{\diamond,d,k}_{pen}$ is computed, given the current positions of the vehicles. As a result, the truck is encouraged to use edges that are knowingly intact, by applying the same penalty factor as above for the truck travel time of each unknown edge in the computation. After the completion of the delivery phase, the $\mathsf{S}_{25\%}^{\mathsf{hybrid}}$ policy terminates with a total objective value equal to the surveillance time plus the duration of the delivery tour.

Table 4.7: Calibration of penalty factors and surveillance time for hybrid policy, on separate dataset

Penalty	$\tilde{\sigma}(\mathsf{S}^{\text{hybrid}}_{25\%})$			$\tilde{\sigma}(\mathsf{S}^{\text{hybrid}}_{50\%})$			$\tilde{\sigma}(\mathsf{S}^{\text{hybrid}}_{75\%})$		
factor w^k_{ij}	Avg	Max	$\% \leq \tilde{\sigma}$(Reopt)	Avg	Max	$\% \leq \tilde{\sigma}$(Reopt)	Avg	Max	$\% \leq \tilde{\sigma}$(Reopt)
1.0	**1.1**	2.6	**84**	**1.2**	2.8	40	**1.2**	2.8	25
1.1	**1.1**	2.6	81	**1.2**	2.5	41	**1.2**	2.5	26
1.2	**1.1**	2.3	80	**1.2**	2.5	42	**1.2**	2.5	28
1.3	**1.1***	**1.8***	79*	**1.2**	2.0	**42**	**1.2**	**1.9**	28
1.4	**1.1**	**1.8**	79	**1.2**	2.0	42	**1.2**	**1.9**	28
1.5	**1.1**	**1.8**	79	**1.2**	2.0	42	**1.2**	2.1	28
1.6	**1.1**	**1.8**	78	**1.2**	2.0	41	**1.2**	2.1	28
1.7	**1.1**	**1.8**	77	**1.2**	2.0	41	**1.2**	2.1	28
1.8	**1.1**	**1.8**	76	**1.2**	2.0	41	**1.2**	2.0	29
1.9	**1.1**	1.9	76	**1.2**	2.0	41	**1.2**	2.0	29
2.0	**1.1**	1.9	75	**1.2**	**1.9**	41	**1.2**	2.1	**29**
2.1	**1.1**	1.9	74	**1.2**	**1.9**	40	**1.2**	**1.9**	29
2.2	**1.1**	1.9	74	**1.2**	**1.9**	40	**1.2**	**1.9**	29
2.3	**1.1**	1.9	73	**1.2**	**1.9**	40	**1.2**	**1.9**	29
2.4	**1.1**	1.9	73	**1.2**	**1.9**	40	**1.2**	**1.9**	29
2.5	**1.1**	1.9	72	**1.2**	**1.9**	39	**1.2**	**1.9**	29
2.6	**1.1**	1.9	72	**1.2**	**1.9**	39	**1.2**	**1.9**	29
3.0	**1.1**	1.9	70	**1.2**	**1.9**	39	**1.2**	**1.9**	29
4.0	**1.1**	2.1	69	**1.2**	2.1	38	**1.2**	2.1	28

Note. This calibration was performed on a *separate data set* of 2.000 instances, generated along the same line as *Based*. For each hybrid policy with varying surveillance times (25%, 50%, 75%), the best average, worst observed, and the ratio of instances where *Reopt* was outperformed are highlighted in **bold**. The values for the selected parameter setting (25% of full surveillance time, 1.3 penalty factor) are marked with *.

Orienteering problem for the drone surveillance after the k'th damaged edge

$$Maximize \quad \sum_{i \in V^{t,k} \cup \{v_{curr}\}} \sum_{j \in V^{t,k} \cup \{v_{curr}\}, j \neq i} p_i x_{ij} \tag{4.21}$$

subject to

$$\sum_{j \in V^{t,k} \cup \{v_{curr}\}} x_{ij} = \sum_{j \in V^{t,k} \cup \{v_{curr}\}} x_{ji} \leq 1 \qquad \forall i \in V^{t,k} \tag{4.22}$$

$$x_{v_0, v_{curr}} = 1 \tag{4.23}$$

$$\sum_{i \in V^{t,k} \setminus \{v_0\}} \sum_{j \in V^{t,k} \setminus \{v_{curr}\}} c^d_{ij} x_{ij} \leq c_{max} \tag{4.24}$$

$$|V^{t,k} \cup \{v_{curr}\}| \cdot x_{ij} + u_i - u_j \leq |V^{t,k} \cup \{v_{curr}\}| - 1 \qquad \forall i,j \in V^{t,k}, i \neq j \tag{4.25}$$

$$x_{ij} \in \{0,1\} \qquad \forall i,j \in V^{t,k} \cup \{v_{curr}\} \tag{4.26}$$

$$u_i \in \{0, 1, ..., |V^{t,k} \cup \{v_{curr}\}|\} \qquad \forall i \in V^{t,k} \cup \{v_{curr}\} \tag{4.27}$$

The objective (4.21) maximizes the sum of the prizes of visited nodes. Constraints (4.22) are flow constraints, and constraint (4.23) guarantees that the starting point and the depot are visited and connected by an artificial edge of weight zero. Constraint (4.24) limits the duration of the tour to c_{max}. Constraints (4.25) are the Miller-Tucker-Zemlin subtour elimination constraints. Constraints (4.26) and (4.27) specify the domains of the decision variables.

4.9.7 Supplement: Alternative objective and impact of non-zero delivery times

Table 4.8 reports the results of the same set of experiments described in Section 4.6.2 using the alternative objective of minimizing the *time of the last delivery*. Table 4.9 and Table 4.10 report the results of the same set of experiments as described in Section 4.6 with

Table 4.8: Relative performance of alternative policies for the last delivery time objective

	$\tilde{\sigma}$(SF)		$\tilde{\sigma}$($\mathsf{S}^{\text{hybrid}}_{25\%}$)		$\tilde{\sigma}$(Reopt)		$\tilde{\sigma}$(CD)		% ≤ $\tilde{\sigma}$(Reopt)	
	Avg	Max	Avg	Max	Avg	Max	Avg	Max	$\tilde{\sigma}$(SF)	$\tilde{\sigma}$($\mathsf{S}^{\text{hybrid}}_{25\%}$)
Influence of the drone speed										
$\alpha = 1.0$	2.4	3.9	1.4	3.7*	1.4	4.1	2.6	**6.0**	2	68
$\alpha = 2.0$	1.7	2.6*	1.2	2.7	1.2	3.6	1.9	**3.8**	5	80
$\alpha = 3.0$	1.5	2.3*	1.1	2.7	1.1	**3.0**	1.6	2.7	6	89
Influence of the drone shortcuts and alternative distance metrics for truck										
L_2 for truck	1.7	2.6*	1.2	2.7	1.2	3.6	1.9	**3.8**	5	80
L_1 for truck	1.6	2.4*	1.2	**3.3**	1.2	**3.3**	1.7	3.1	5	85
L_2 for truck, shortcuts	1.6	2.5*	1.2	3.1	1.2	3.3	1.8	**3.7**	5	91
L_1 for truck, shortcuts	1.5	2.2*	1.1	3.2	1.1	**3.7**	1.6	2.9	7	94
Influence of parking possibilities										
$\lvert D \rvert = 6$	1.7	2.6*	1.2	2.7	1.2	3.6	1.9	**3.8**	5	80
$\lvert D \rvert = 11$	1.8	2.7	1.2	2.5*	1.2	2.7	2.0	**3.8**	2	78
$\lvert D \rvert = 16$	1.8	2.8	1.2	2.2	1.2	2.1*	2.0	**3.8**	1	77

Note. The *Base* setting is highlighted in grey. The *maximum* (*minimum*) worst observed ratios to the CIOS result for each setting are marked in **bold** (marked with *).

a non-zero delivery time for the makespan and last delivery time objectives, respectively. We set delivery times to 10, which is a conventional parameter for delivery times in the vehicle routing literature (cf., Solomon, 1987). We assume the following sequence of events upon the arrival of a vehicle at a node, where it has to perform a delivery: First, the status of the adjacent edges to his node is revealed, then – in case of the delivery by the truck and if applicable – the drone can dispatch for a sortie from the truck or return from a sortie and land on the truck, afterwards, the delivery commences.

Table 4.9: Relative performance of alternative policies for the makespan objective with a service time of 10

	$\tilde{\sigma}$(SF)		$\tilde{\sigma}$($\mathsf{S}^{\text{hybrid}}_{25\%}$)		$\tilde{\sigma}$(Reopt)		$\tilde{\sigma}$(CD)		% ≤ $\tilde{\sigma}$(Reopt)	
	Avg	Max	Avg	Max	Avg	Max	Avg	Max	$\tilde{\sigma}$(SF)	$\tilde{\sigma}$($\mathsf{S}^{\text{hybrid}}_{25\%}$)
Influence of the drone speed										
$\alpha = 1.0$	1.9	2.7	1.2	2.4*	1.2	2.5	2.1	**3.7**	1	58
$\alpha = 2.0$	1.5	2.1	1.1	2.0*	1.1	2.3	1.6	**2.5**	3	78
$\alpha = 3.0$	1.3	1.7*	1.1	2.0	1.1	**2.2**	1.4	1.9	5	89
Influence of the drone shortcuts and alternative distance metrics for truck										
L_2 for truck	1.5	2.1	1.1	2.0*	1.1	2.3	1.6	**2.5**	3	78
L_1 for truck	1.3	1.8*	1.1	2.0	1.1	**2.4**	1.5	2.1	4	82
L_2 for truck, shortcuts	1.4	1.8*	1.1	**2.5**	1.1	2.5	1.5	2.3	6	90
L_1 for truck, shortcuts	1.3	1.7*	1.0	1.8	1.0	**2.5**	1.4	2.0	7	95
Influence of parking possibilities										
$\lvert D \rvert = 6$	1.5	2.1	1.1	2.0*	1.1	2.3	1.6	**2.5**	3	78
$\lvert D \rvert = 11$	1.5	2.2	1.1	2.1*	1.1	2.2	1.7	**2.5**	2	74
$\lvert D \rvert = 16$	1.5	2.2	1.1	2.0*	1.1	2.0*	1.7	**2.5**	1	73

Note. The *Base* setting is highlighted in grey. The *maximum* (*minimum*) worst observed ratios to the CIOS result for each setting are marked in **bold** (marked with *).

Table 4.10: Relative performance of alternative policies for the last delivery time objective with a service time of 10

	$\tilde{\sigma}$(SF)		$\tilde{\sigma}(\text{S}_{25\%}^{\text{hybrid}})$		$\tilde{\sigma}$(Reopt)		$\tilde{\sigma}$(CD)		% $\leq \tilde{\sigma}$(Reopt)	
	Avg	Max	Avg	Max	Avg	Max	Avg	Max	$\tilde{\sigma}$(SF)	$\tilde{\sigma}(\text{S}_{25\%}^{\text{hybrid}})$
	Influence of the drone speed									
$\alpha = 1.0$	2.4	3.9	1.4	3.7*	1.4	3.9	2.6	**5.9**	2	68
$\alpha = 2.0$	1.7	2.6*	1.2	2.7	1.2	3.6	1.9	**3.8**	5	79
$\alpha = 3.0$	1.5	2.5*	1.1	2.6	1.1	**3.0**	1.6	2.7	6	88
	Influence of the drone shortcuts and alternative distance metrics for truck									
L_2 for truck	1.7	2.6*	1.2	2.7	1.2	3.6	1.9	**3.8**	5	79
L_1 for truck	1.6	2.4*	1.2	**3.3**	1.2	**3.3**	1.7	3.1	5	85
L_2 for truck, shortcuts	1.6	2.5*	1.2	3.1	1.2	3.3	1.8	**3.6**	6	91
L_1 for truck, shortcuts	1.5	2.2*	1.1	3.2	1.1	**3.6**	1.6	2.9	8	92
	Influence of parking possibilities									
$\|D\| = 6$	1.7	2.6*	1.2	2.7	1.2	3.6	1.9	**3.8**	5	79
$\|D\| = 11$	1.8	2.6	1.2	2.5*	1.2	2.7	2.0	**3.8**	2	78
$\|D\| = 16$	1.8	2.8	1.2	2.2*	1.2	2.3	2.0	**3.8**	1	77

Note. The *Base* setting is highlighted in grey. The *maximum* (*minimum*) worst observed ratios to the CIOS result for each setting are marked in **bold** (marked with *).

Chapter 5

Very large-scale neighborhood search for drone routing with energy replenishment

Catherine Lorenz, Nicola Mimmo, Alena Otto, Daniele Vigo

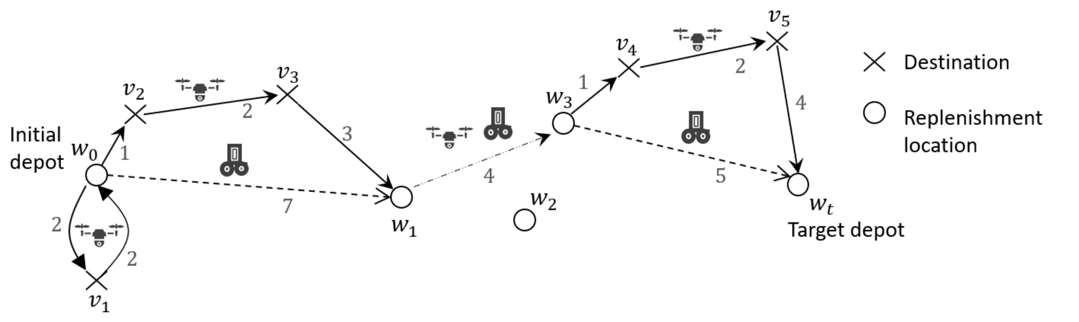
Abstract. The Drone Routing Problem with Energy replenishment (DRP-E) describes a general class of routing problems with intermediate stops and synchronization constraints. In DRP-E, the drone has to visit a set of nodes and routinely requires battery swaps, energy- or payload replenishment from a mobile replenishment station. Thereby, the drone may visit several destinations between two replenishments, and mutual waiting at the rendezvous locations may occur. In this paper, we propose a very large-scale neighborhood that synergetically leverages two large-sized polynomially solvable DRP-E $\underline{s}$ub$\underline{p}$roblems (SP1 and SP2). The number of feasible solutions in the resulting neighborhood is a multiple of those in SP1 and SP2, and, thus, exponential in the input size of the problem. We develop a non-trivial search procedure, VLNS, which examines this neighborhood entirely, in a computational time that remains polynomial in the problem size. The desired trade-off between accuracy and runtime of the proposed two-stage dynamic programming approach can be flexibly adjusted with just a single parameter. For large parameter values, it converts to an exact approach. VLNS is a flexible improvement procedure, which is easy to implement. It can be straightforwardly adapted to many DRP-E variants and may be embedded in any algorithmic scheme, meta- or math-heuristic. In computational tests, we demonstrate that a simple VLNS-based local search heuristic outperforms state-of-the-art heuristics for several DRP-E variants by a significant margin. We furthermore propose a high-performing exact approach for DRP-E.

5.1 Introduction

Aerial drones are a highly promising modern technology. Just in the U.S. alone, around 850,000 drones are registered for commercial and recreational purposes FAA, 2022. One of the main challenges in the usage of commercially attractive small multi-rotor drones is the short battery life, so a large number of applications require periodic landings of the drones for energy replenishment, e.g., by performing a battery swap. Currently available technologies allow the battery swaps within just a few seconds, while the drone may remain fully powered during this time (Mohsan et al., 2022; Schneider et al., 2022).

In this article, we study the scheduling of a *single drone*, in which the drone has to replenish its energy regularly, and possibly refill other capacitated resources, like payload. We formulate a basic optimization problem –*the $\underline{D}$rone $\underline{R}$outing $\underline{P}$roblem with $\underline{E}$nergy*

Figure 5.1: Illustration of a feasible solution for DRP-E: Route of the MRS and the drone



Note. Example for $n_d = 5$ destinations, $n_r = 5$ RLs, and a battery capacity of $e_{max} = 7$ time units; the makespan of the route equals 22. The route contains three operations with makespans $\mathcal{M}(o_1) = \mathcal{M}(w_0, v_1, w_0) = 4$, $\mathcal{M}(o_2) = \mathcal{M}(w_0, v_2, v_3, w_1) = 7$, and $\mathcal{M}(o_3) = \mathcal{M}(w_3, v_4, v_5, w_t) = 7$ as well as one nonempty traveling leg with makespan $\mathcal{M}(r_2) = \mathcal{M}(w_1, w_3) = 4$. The drone tour can be notated as an alternating sequence of operations and traveling legs $\pi_d = (r_0, o_1, r_1, o_2, r_2, o_3, r_3)$, the traveling legs r_0, r_1, r_3 being empty. See Section 5.2 for details.

replenishment (DRP-E) (cf. Figure 5.1), which is relevant for a large number of applications. The drone has to visit a set of *points of interest (destinations)*. It can replenish its energy and other resources at one of the *replenishment locations (RLs)* by meeting a *mobile replenishment station (MRS)*. Specially equipped trucks, replenishment robots, or autonomous platforms are examples of such MRSs. At RLs, the vehicles can wait for each other and perform the replenishment operations safely. The objective is to sequence the visits of destinations by the drone, schedule the drone's replenishment stops, and determine the replenishment locations to minimize the *makespan* subject to the following basic constraints (additional constraints are possible):

- The energy consumption of the drone between two subsequently visited RLs should not exceed the maximal energy capacity of the drone's battery;
- The replenishment starts when both the drone and the MRS have reached the selected RL, in other words, *waiting times* eventually emerge.

The described problem setting, in which a single drone has to visit a set of nodes, emerges in a large number of applications (e.g., Evers et al., 2014; Guerrero & Bestaoui, 2013; Jawhar et al., 2014; K. Li et al., 2016), (see also Otto, Agatz, et al., 2018). For example in disaster management, environmental monitoring, and infrastructure maintenance, the drone has to collect data from sensors. Drones also scan or take photographs of points of interest for maintenance, police surveillance, filming or for the purposes of precision agriculture. Furthermore, the problem setting emerges in the surveillance of non-convex and/or non-connected areas, where area discretization remains a state-of-the-art solution technique (Cabreira et al., 2019; Galceran & Carreras, 2013). DRP-E requires a high degree of synchronization compared to many other routing problems with intermediate stops and synchronization constraints (cf. Section 5.1.1). For such problems, solutions of frequently used construction heuristics of the route-first-split-second-type are notoriously hard to improve, notwithstanding the significant gap to optimality of these solutions in a number of settings. In recent drone-routing literature, metaheuristics, like the *adaptive large neighborhood search (ALNS)*, utilizing rapid destroy- and repair operators have demonstrated considerable effectiveness in diversifying the heterogeneous vehicle's routes. However, complex spacio-temporal synchronization constraints pose substantial challenges in verifying the feasibility of these moves. Moreover, in terms of intensification, these methods often struggle to achieve the necessary adjustments to enhance already promising solutions. Therefore, the focus of this paper being on powerful and flexible improvement procedures for DRP-E appears to be a logical step toward achieving progress within this class of problems.

In this article, we propose *very large-scale neighborhood search (VLNS)* for DRP-E, which can search an extremely large neighborhood *exactly*, whose size is exponential in the input size of the problem, in a computational runtime that grows *polynomialy* in the input size. Problem-specific search procedures for very large neighborhoods achieved remarkable success in a wide range of problems (see Ahuja et al., 2002; Brueggemann & Hurink, 2011; Grangier et al., 2017). Moreover, in contrast to generic heuristic approaches (including ALNS algorithms), these search procedures overcome the critique of *No-free-lunch theorems* (Wolpert & Macready, 1997). For instance, the merit of VLNS has a solid analytical justification in terms of the number of feasible solutions that can be examined within an extremely short computational time (see Theorems 8 and 9).

We design VLNS based on the idea of Balas and Simonetti (2001). We prove that for DRP-E (and similar problems), the very large neighborhood of Balas and Simonetti (2001) can be merged synergetically with another very large neighborhood such that the size of the resulting, 'extremely' large neighborhood is a *multiple* of the initial two. The elaboration of the efficient search procedure for this neighborhood is the key contribution of this paper, as it introduces a tight construction scheme based on novel ideas and observations, which are backed up by non-trivial proofs. For instance, the search procedure of Balas and Simonetti (2001) cannot be straightforwardly adapted, because of the problem-specific *two-stage architecture* of VLNS, which provides the flexibility and tractability to handle the two-echelon structure of DRP-R. Fortunately for future applications, once the main framework is established, the implementation of VLNS is quite straightforward and boils down to simple two-stage dynamic programming with a look-up table.

The VLNS's architecture involves *i)* a *meta state graph* that subsumes the properties of the MRS and *ii)* an *operations state graph* that describes the properties of the drone. This architecture allows for easy accommodation of different drone and MRS characteristics without the need to adjust the overall logic of VLNS, which may be especially attractive for practice in view of the constantly evolving drone technology. For instance, VLNS can be adopted to stationary recharging stations instead of the mobile MRS, cargo-drones, as well as to some alternative objective functions. Furthermore, as a valuable extension of our research, we were able to exploit the same two-stage architecture to propose a simple, high-performing exact solution approach for DRP-E.

In the following, we review related literature in Section 5.1.1 and conclude by stating the contribution of this article and providing the outline of the paper in Section 5.1.2.

5.1.1 Literature review

The formulated DRP-E can be interpreted as a variant of the Traveling Salesman Problem (TSP) which is embedded in a two-echelon network: The drone is a second echelon vehicle that must regularly meet with the first-echelon vehicle, the MRS, for replenishment. In this regard, DRP-E is closely related to the classical Two-Echelon Vehicle Routing- (2E-VRP), the Two-Echelon Location Routing (2E-LRP) and the Truck and Trailer Routing Problems (TTRP) (cf. H. Li et al., 2021; Sluijk et al., 2023, for recent reviews). Single-Truck TTRPs (Accorsi & Vigo, 2020) are direct special cases of DRP-E. However, the synchronization between the heterogeneous vehicles in DRP-E is more complex and routes are much more closely interrelated compared to those classical problems: The drone can be launched in one RL, and rejoin the MRS at another one, which may lead to mutual waiting times. Whereas state-of-the-art approaches for 2E-VRP, 2E-LRP, and TTPR often rely on the separability of the routing sub-problems of the vehicles, this approach doesn't seem reasonable in our case and the solution method developed in this paper constructs the heterogeneous vehicle routes simultaneously. We refer to the surveys of Drexl (2012) and Soares et al. (2023) for a detailed elaboration on different classes of synchronization.

Several other routing problems are DRP-E special cases and could serve as potential application targets for the developed VLNS. For instance, the MRS in DRP-E can be replaced straightforwardly by multiple stationary replenishment stations, to form a special case of DRP-E which belongs to the family of location routing problems with intermediate stops (cf. Schiffer et al., 2019; Tarantilis et al., 2008). Furthermore, the general class of asymmetric TSPs with replenishment arcs (Mak & Boland, 2007; Smith et al., 2012) can be converted to DRP-E using basic arc-node transformations. Also, the TSP with hotel selection with the objective of minimizing the number of nights spent in the hotel (Vansteenwegen et al., 2011) is a DRP-E special case. VLNS can straightforwardly be adapted to stationary replenishment stations (c.f. Section 5.4.1).

Recently, complex spacio-temporal synchronization constraints, similar to DRP-E, have been introduced to the field of Electric and Green Vehicle Routing Problems(see e.g. Marrekchi et al., 2021; Vidal et al., 2020). Although this thread of the literature mostly considers stationary recharging stations to handle the limited operating range of *delivery vehicles (DVs)*, the recent articles by Çatay and Sadati (2023), Hof and Schneider (2021), Raeesi and Zografos (2020, 2022), and Ren et al. (2023) and Xiao et al. (2024) study mobile replenishments with MRSs. However, these problem formulations are motivated by different types of applications and differ from DRP-E in several key aspects. The objective is to minimize the *overall fixed- and variable costs* of a delivery system with *several* DVs and *several* MRSs, while adhering to hard time windows at the customer locations. Therefore, major contributions in the proposed solution methodology focus on the accurate and fast accounting of neighborhood moves between the routes of distinct vehicles, e.g., to reduce the fleet size. DRP-E, on the other side, focuses on reducing the overall makespan. Most notably, the majority of studies significantly simplify the synchronization between DVs and MRSs by *eliminating DV waiting times.* The only exception are studies of Hof and Schneider (2021) and Xiao et al. (2024). Hof and Schneider (2021) propose an *adaptive large neighborhood search (ALNS)* with removal- and insertion operators for *diversification* of different vehicle's routes, complemented with a path relinking method for *intensification*. Xiao et al. (2024) restrict replenishment - and thus, the meeting points DVs and MRSs - to *customer locations only* which significantly simplifies the problem. In DRP-E, the presence of distinct, non-proximate recharging locations transforms the DV's routing problem into a *generalized* one, where visiting additional locations becomes optional. This, in turn, alters the sequencing of customer locations considerably, as *'neighboring' customer locations may become 'distant'* if a remote recharging location must be visited in between (see also discussions in our computational experiments). Xiao et al. (2024) design a customized dynamic-programming-based repair operator for ALNS that inserts the (limited number of) nodes, which have been removed in the previous destruction-operator move. Both studies confirm the importance of customized intensification in ALNS. In this context, the present study introduces an additional intensification operator for EV problem classes. Finally, studies on EV vehicles do not allow for drone-specific features (such as an opportunity to ride on the back of the MRS without consuming energy).

Turning to the vast and rapidly evolving literature on drone-and-truck routing, we restrict ourselves to the main, distinctive properties of DRP-E, and refer the reader to the recent surveys of Chung et al. (2020), Macrina et al. (2020), Moshref-Javadi and Winkenbach (2021), Otto, Agatz, et al. (2018), and Srinivas et al. (2022) for a generic overview on drone routing. In contrast to the classical TSP with Drones, or the Flying

Sidekick TSP (cf. e.g., Agatz et al., 2018; Bouman et al., 2018; Murray & Chu, 2015), the drone in DRP-E is allowed to visit *multiple destinations* in between two replenishments, and it can be recovered at a location different from its launch point, allowing for *mutual waiting* times between the MRS and the drone. In the field of package delivery, several studies have adopted these properties in a system, where *both* a truck (the MRS) and a drone can deliver packages to the customers. In that context, Gao et al. (2023) and Yin et al. (2023) have developed exact approaches for their problem variants with customer time windows, namely a hybrid algorithm combining column generation and Benders decomposition techniques, and a branch-and-price-and-cut algorithm, allowing them to solve instances with up to 25 and 35 customer locations, respectively. Most articles in this field propose metaheuristics (e.g. Variable Neighborhood Descent (VND), ALNS, and simulated annealing (SA)), which, at their core, iteratively apply destroy- and repair operators of different neighborhoods to *one current candidate solution*. The algorithmic contributions include the extension of classical operators like swaps or reinsertion with problem-specific elements (see Jiang et al., 2024; Madani et al., 2024; Meng et al., 2023), procedures to navigate intelligently through the diverse neighborhoods (e.g., through adaptive learning, clustering of neighborhoods or tabo lists, see Madani et al., 2024; Meng et al., 2023), the extension to multiple objectives (P. L. Gonzalez-R et al., 2024) or fast feasibility checks (Meng et al., 2023). In contrast, Masmoudi et al. (2022) aim to balance intensification and diversification by proposing a *population-based* version of the SA, injecting elements from evolutionary-based algorithms, to solve the considered multi-drone and multi-truck package delivery problem. The VLNS proposed in our paper can be perceived as a technique for *intensification* around a promising solution, which explores an *extremely large* number of neighbors around this solution *simultaneously*.

Furthermore, in DRP-E, the drone, as the *primary* vehicle, has to visit *all* destinations, whereas the MRS is a *purely supportive* vehicle. Such kind of *two-echelon* systems frequently occur in diverse applications, like the inspection of infrastructure, search-and-rescue, or surveillance, but as well in package delivery, for instance in urban areas to reduce carbon emissions. In this area, multi-visits and mutual waiting are considered by Karak and Abdelghany (2019), who investigate a pick-up-and-delivery problem with one vehicle and a swarm of drones with the cost minimization objective, where the vehicle can visit each RL at most once. They propose a construction heuristic based on the computation of regrets, which is an adaptation of the Clarke and Wright savings algorithm. In the case of a single drone, the problem formulations of Poikonen and Golden (2020) and Zeng et al. (2022) fall under this category and are, to the best of our knowledge, the closest problem formulations to DRP-E. Zeng et al. (2022) feature a drone-lead surveillance application with additional constraints on the routes (e.g. each vehicle may visit a node at most once). They propose a mixed-integer program and a customized iterated local search. Poikonen and Golden (2020) study DRP-E variant for package delivery. The authors develop a route-first-split-second construction heuristic, dubbed *Route, Transform, Shortest Path (RTS)*, which optimally inserts replenishment stops within a *fixed* sequence of destinations (= 'splits' this sequence). Both articles consider some application-specific generalizing aspects compared to the basic variant of DRP-E, which we explain in Section 5.5. We use the approaches of these articles as a benchmark for VLNS on the respective DRP-E variants in our computational experiments.

Finally, most articles on joint drone- and MRS routing have proposed a mathematical programming formulation for their studied problem variant (see, e.g., Jiang et al., 2024; Meng et al., 2023; Zeng et al., 2022). However, the largest instances that could be solved to optimality was limited to roughly 10 customers locations, so that the gaps to optimality of the developed solution approaches have been provided for small-sized instances only. In this paper, the proposed DP-ILP can solve significantly larger problem sizes across all settings to proven optimality, and can solve specific DRP-E problem settings with up to 149 locations optimally.

5.1.2 Contribution and the outline of the paper

DRP-E describes a general class of problems that are challenging to solve as they involve a close temporal synchronization of two vehicles. Our paper makes the following contributions:

- Our key contribution is the introduction of a non-trivial very large-scale neighborhood for DRP-E that can be searched exactly in pseudo-polynomial time with our developed search procedure VLNS. We perform an extensive formal analysis of the proposed VLNS, including its computational complexity and the number of the examined solutions.
- The designed VLNS approach refers to an *algorithmic scheme*, in the sense that the size of the neighborhood is controlled by a single parameter and any desired trade-off between the number of examined feasible solutions and the computational time can be achieved by setting the value of this parameter respectively. For a large enough parameter value, the resulting algorithm becomes exact.
- We show that VLNS can be straightforwardly adapted to a large range of DRP-E variants (cf. Section 5.4.1). As an example, we apply VLNS to a drone surveillance application incorporating numerous problem-specific features, and a package delivery problem with a payload-dependent energy consumption of the drone in our computational experiments (cf. Section 5.5).
- VLNS is a flexible improvement tool that can be embedded in many exact-, meta- or math-heuristic algorithms. In this paper, we show that a simple VLNS-based local search outperforms state-of-the-art algorithms.
- As a valuable extension of our research, we also propose an exact solution approach (DP-ILP) for DRP-E that incorporates elements of VLNS. This approach is capable of finding new optimal solutions for the studied DRP-E variants from the literature and can solve instances with up to 149 locations in certain settings of the basic DRP-E.

To avoid distracting details in outlining the mechanics of VLNS, we first introduce formally a *basic variant* of DRP-E in Section 5.2. Basic DRP-E fits the main framework of many drone- or general (e.g., electric-) vehicle routing problems with mobile replenishments. Section 5.3 outlines VLNS. In Section 5.4, DRP-E and the proposed VLNS are extended with additional problem features that realistically occur in drone applications, VLNS is embedded in a local search framework, and the exact solution approach DP-ILP is presented. We conclude the paper with an extensive computational study on a tailored DRP-E dataset and DRP-E variants from the literature (see Section 5.5), followed by final remarks and an outlook in Section 5.6.

5.2 The drone routing problem with energy replenishment

DRP-E can be described as follows. Given are

- a set of destinations V_d, $|V_d| = n_d$,
- a set of RLs V_r, $|V_r| = n_r$,

- special RLs $w_0 \in V_r$ (initial depot) and $w_t \in V_r$ (target depot),
- drone flight times $c_d : (V_d \cup V_r) \times (V_d \cup V_r) \rightarrow \mathbb{R}_{\geq 0}$,
- MRS travel times $c_r : V_r \times V_r \rightarrow \mathbb{R}_{\geq 0}$,
- a maximal flight time $e_{max} \in \mathbb{R}_{>0}$; e_{max} depends on the capacity of the drone's battery. To exclude infeasible instances, we assume that e_{max} is enough to visit each destination in a return flight from its nearest RL.

We define c_d and c_r as the *shortest* drone flight and MRS travel times between a pair of locations, respectively, thus the triangular inequalities are valid, i.e. $c_d(i,j) \leq c_d(i,k) + c_d(k,j)\ \forall k \in V_d \cup V_r$ (equivalently for c_r).

We call *operation* a drone sortie starting from a RL, then visiting at least one destination and returning to the same or a different RL (see Figure 5.1). We notate operation o with $k(o)$ visited destinations as an ordered sequence $o = w, v_{O1}, v_{O2}, ...v_{Ok}, w' = wsw'$, with $w, w' \in V_r$ being the starting and ending RLs of the operation, respectively, Oi denoting the index of the ith destination in the operation, and s being the destinations sorted according to their visiting order. We call operation o *feasible*, if the drone flight time $C_d(o)$ of this operation does not exceed e_{max}, i.e.,

$$C_d(o) := c_d(w, v_{O1}) + \sum_{i=1}^{k(o)-1} c_d(v_{Oi}, v_{O(i+1)}) + c_d(v_{Ok}, w') \leq e_{max}.$$

In *basic* DRP-E, we assume the drone may land and turn off when arriving first at an RL, which enables drone waiting without battery consumption. The case of drone waiting times in a *hovering* state is discussed in Section 5.4.1. We call *traveling leg* r a sequence of $k(r)$ consecutive RLs, where the drone travels on the back of the MRS without spending energy, $r = (w_{R1}, \ldots, w_{Rk})$ with $w_{R1}, \ldots, w_{Rk} \in V_r$.

Objective: Determine a feasible *drone tour*, i.e. a sequence of operations and traveling legs, $\pi_d = (r_0, o_1, r_1, o_2, \ldots)$ such that

- each destination is visited, i.e. $v \in \pi_d\ \forall v \in V_d$,
- sequence π_d starts at the initial depot w_0 and ends at the target depot w_t,
- each operation $o \in \pi_d$ is feasible,
- the makespan $\mathcal{M}(\pi_d) = \sum_{o \in \pi_d} \mathcal{M}(o) + \sum_{r \in \pi_d} \mathcal{M}(r)$ is minimized, where
  - $\mathcal{M}(o) = \max\{C_d(o); c_r(w, w')\}$, for $o = wsw'$ (makespan of operation o),
  - $\mathcal{M}(r) = \sum_{i=1}^{k(r)-1} c_r(w_{Ri}, w_{R(i+1)})$ for $r = (w_{R1}, \ldots, w_{Rk})$ (makespan of traveling leg r).

Observe that the makespan of a traveling leg solely depends on the MRS, because the drone travels on its back without spending any energy. On the other hand, the makespan of an operation equals to the maximum of the drone's and the MRS's travel times, because both vehicles have to meet for the battery exchange. See Figure 5.1 for an example. W.l.o.g., we use the following properties of DRP-E in our solution procedure VLNS, which directly follow from the triangular inequalities of the drone flight times and MRS travel times:

1) π_d is an *alternate* sequence of operations and 2-node traveling legs: $k(r) = 2$ $\forall r \in \pi_d$ (trivial traveling legs $r = (w, w)$ for $w \in V^r$ mark consecutive operations).

2) Each destination is visited exactly once.

DRP-E is NP-hard, as its special case with unlimited battery capacities of the drone reduces to a standard traveling salesman problem.

5.3 The very large-scale neighborhood search

As stated above, we propose a very large-scale neighborhood that synergetically unites two polynomially solvable DRP-E *subproblems* (SPs). The first one, SP1, is a TSP over destinations with restrictions on the relative positions of the nodes as explored in the seminal studies of Balas (1999) and Balas and Simonetti (2001). The second one, SP2, is DRP-E with a fixed sequence of destination visits x. We call the designed neighborhood *B̲alas-S̲imonetti neighborhood with R̲eplenishment (BS-R)*. For example, for the sequence of destination visits by the drone from Figure 5.1, which is $x = (v_1, v_2, v_3, v_4, v_5)$, a possible formulation of BS-R examines eight sequences of destinations listed in Table 5.1 as part of SP1 and all possible insertions of RLs $(w_0, w_1, w_2, w_3, w_t)$ into each of these sequences as part of SP2.

To search BS-R, we propose a two-stage dynamic programming procedure VLNS consisting of an *operations state graph (ops graph)* and a *meta state graph (meta graph)*. The meta graph explores how to construct an optimal drone tour in BS-R by combining traveling legs and operations, precomputed in the ops graph. For each subset of destinations and a pair of start- and end-RLs, the ops graph computes a feasible operation for BS-R neighbors with a minimum required drone flight time. Much mathematical effort is devoted to keeping these graphs polynomial in size.

Throughout this section, we assume that x denotes a Hamiltonian path over V_d and that destinations in x are enumerated in the increasing order of their position, i.e. $x = (v_1, v_2, \dots, v_{n_d})$. We also use shortcut $[n]$ for the set of integers $\{1, \dots, n\}, n \in \mathbb{N}$, and similarly $[a_1, b_1]$ for integers $\{a_1, \dots, b_1\}$. The position of destination v_i in a permutation σ of the visiting sequence x is referred to as $\sigma(i)$, while $\sigma^{(-1)}(l)$ returns the destination at the lth position in permutation σ.

We proceed with the formulation of BS-R and its properties in Section 5.3.1. Then, Sections 5.3.2 and 5.3.3 explain the construction of the ops graph and the meta graph, respectively. The resulting VLNS algorithm is quite easy to implement and we provide its pseudocode in Online Supplement 5.8.2. Finally, Section 5.3.4 states the time complexity of the VLNS.

5.3.1 Balas-Simonetti neighborhood with replenishment

To define the proposed BS-R, we first introduce *Balas-Simonetti neighbourhood (BS)*.

For a given Hamiltonian path of destinations $x : (v_1, ..., v_{n_d}), v_i \in V_d$, and a parameter $p \in \mathbb{N}$, the Balas-Simonetti neighbourhood $\mathcal{N}_{\mathcal{BS}}(x, p)$ is a collection of permutations defined by the following precedence constraints:

$$x' = (v'_1, v'_2, ...v'_{n_d}) = (v_{\sigma^{-1}(1)}, v_{\sigma^{-1}(2)}, ...v_{\sigma^{-1}(n^d)}) \in \mathcal{N}_{\mathcal{BS}}(x, p) \text{ iff}$$
$$\forall i, j \in [n_d] \text{ s.t. } i + p \le j \text{ we have } \sigma(j) > \sigma(i) \tag{5.1}$$

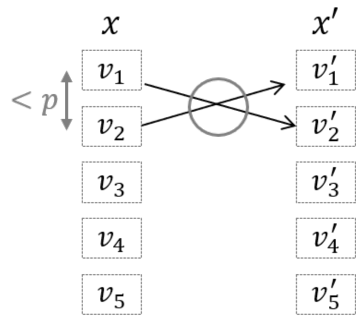
Table 5.1 illustrates how to construct BS-neighbors for $n_d = 5$ and $p = 2$.

Now, we formally define BS-R, which integrates the replenishment decisions.

Definition 1 (Balas-Simonetti neighbourhood with replenishment $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}(x, p)$). *For a given Hamiltonian path of destinations x and a given parameter $p \in \mathbb{N}$, we define the Balas-Simonetti neighbourhood with replenishment $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}(x, p)$ as the subset of all drone tours π_d such that the drone tour restricted to destinations $\pi_d(V_d)$ respects the precedence constraints of $\mathcal{N}_{\mathcal{BS}}(x, p)$:*

$$\pi_d \in \mathcal{N}_{\mathcal{BS}-\mathcal{R}}(x, p) \quad \textit{iff} \quad \pi_d(V_d) \in \mathcal{N}_{\mathcal{BS}}(x, p)$$

Table 5.1: Balas-Simonetti neighbourhood for $x = (v_1, v_2, v_3, v_4, v_5)$ and $p = 2$



Neighbors in $\mathcal{N}_{\mathcal{BS}}(x, 2)$:
$(v_1, v_2, v_3, v_4, v_5)$
$(v_1, v_2, v_3, \boldsymbol{v_5}, \boldsymbol{v_4})$
$(v_1, v_2, \boldsymbol{v_4}, \boldsymbol{v_3}, v_5)$
$(v_1, \boldsymbol{v_3}, \boldsymbol{v_2}, v_4, v_5)$
$(v_1, \boldsymbol{v_3}, \boldsymbol{v_2}, \boldsymbol{v_5}, \boldsymbol{v_4})$
$(\boldsymbol{v_2}, \boldsymbol{v_1}, v_3, v_4, v_5)$
$(\boldsymbol{v_2}, \boldsymbol{v_1}, v_3, \boldsymbol{v_5}, \boldsymbol{v_4})$
$(\boldsymbol{v_2}, \boldsymbol{v_1}, \boldsymbol{v_4}, \boldsymbol{v_3}, v_5)$

Note. The table on the right lists all the neighbours in $\mathcal{N}_{\mathcal{BS}}(x, p)$. The figure on the left illustrates, how to construct these neighbours. So-called matching arcs depict, to which position a node in x moves in x', e.g. v_1 can move to position 1 in x'. To receive a valid $\mathcal{N}_{\mathcal{BS}}(x, p)$-neighbour, only matching arcs that start in nodes $v_i, v_j : |i - j| < p$ may cross.

Observe that we can initialize neighborhood BS-R without having a feasible DRP-E solution, since it is defined just on the visiting sequence of destinations x and parameter $p \in \mathbb{N}$.

Having defined BS-R, we are able to state a lower bound for its complexity.

Theorem 8. *For a given sequence x of destinations and a given parameter p, the number of feasible drone tours that belong to $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}(x, p)$ for DRP-E grows exponentially with the input size n_d (number of destinations).*

If the maximal flight time is unlimited, $e_{max} \to \infty$,

$$|\mathcal{N}_{\mathcal{BS}-\mathcal{R}}(x, p)| \geq (n_r)^{2n_d} \cdot \left(\frac{p-1}{e}\right)^{n_d - 1} \tag{5.2}$$

Otherwise, for $e_{max} \in \mathbb{R}$,

$$|\mathcal{N}_{\mathcal{BS}-\mathcal{R}}(x, p)| \geq \left(\frac{p-1}{e}\right)^{n_d - 1} \tag{5.3}$$

Proof. Consider the case of $e_{max} \to \infty$ first. By Proposition 1 in Balas and Simonetti (2001), $\mathcal{N}_{\mathcal{BS}}(x, p)$ contains at least $\left(\frac{p-1}{e}\right)^{n_d - 1}$ sequences of destinations. For each sequence of destinations x', VLNS implicitly examines all possible replenishment schedules and inserts replenishment stops optimally. By Property 1) in Section 5.2, it is enough to examine two-node traveling legs between each pair of operations. I.e., after each destination in x', we can either fly to the next destination or end the current operation and insert a traveling leg, and there are n_r^2 possible traveling legs, except for the last one that should end in w_t. We add n_r variants for the initial traveling leg that starts in w_0. This results in $n_r \cdot (n_r^2 + 1)^{(n_d - 1)} \cdot n_r$, or $O\left(n_r^{2n_d}\right)$, replenishment schemes for each examined sequence of destinations. Lower bound (5.2) follows immediately.

If e_{max} is limited, then we can construct at least one feasible drone tour for each sequence of destinations $x' = (v_{\sigma^{-1}(1)}, v_{\sigma^{-1}(2)}, ... v_{\sigma^{-1}(n_d)}) \in \mathcal{N}_{\mathcal{BS}}(x, p)$. Indeed, in Section 5.2 we assumed that the maximal flight time e_{max} is large enough to visit each destination in a return flight from its nearest RL. Therefore at least the tour, where the drone replenishes energy after each visited destination in the closest-by RL, is feasible for each sequence of destinations x'. Thus, the number of examined feasible drone tours is not less than the number of neighbors in $\mathcal{N}_{\mathcal{BS}}(x, p)$. $\square$

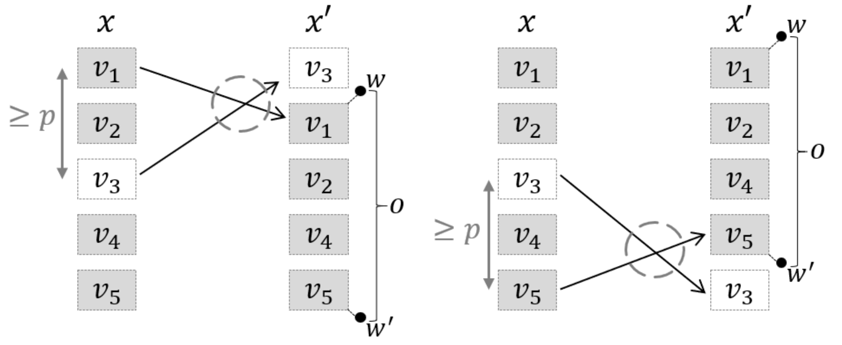
We note that the bound in Theorem 8 is overly pessimistic, since the actual number of the examined drone tours is many times larger than this bound and rapidly grows with n_d even at such low values of p as $p = 2$, as we show in computational experiments in Section 5.5.3.

We proceed with the main contribution of this paper, which is the development of the search procedure VLNS to search the extremely large BS-R entirely while keeping its runtime complexity polynomial in the instance size (but exponential in the parameter p). The construction is non-trivial, as it does *not* straightforwardly extend from the search procedure of BS presented by Balas (1999). In the proposed two-stage approach, which is necessary for the flexible integration of replenishment into the drone sequences of DRP-E, the BS-precedence constraints (5.1) must be checked efficiently by VLNS for *disconnected sub-sequences* - the operations - in the ops graph, invalidating the techniques of Balas (1999) for the *connected* TSP. Furthermore, different from Balas (1999), the arcs of the meta graph represent operations and traveling legs, rather than direct transitions between destinations, necessitating novel validation techniques for the precedence constraints.

5.3.2 Operations state graph

In the ops graph, we generate a collection of operations by computing the best way to fly over any subset of destinations $S \subseteq V_d$ starting and ending in any pair of RLs w and w', respectively. The challenge is to consider only those operations which are relevant for the BS-R neighbors of x given neighborhood $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}(x, p)$, i.e. only those which can be found in at least one neighboring drone tour π_d of $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}(x, p)$. We call such operations *$\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid.*

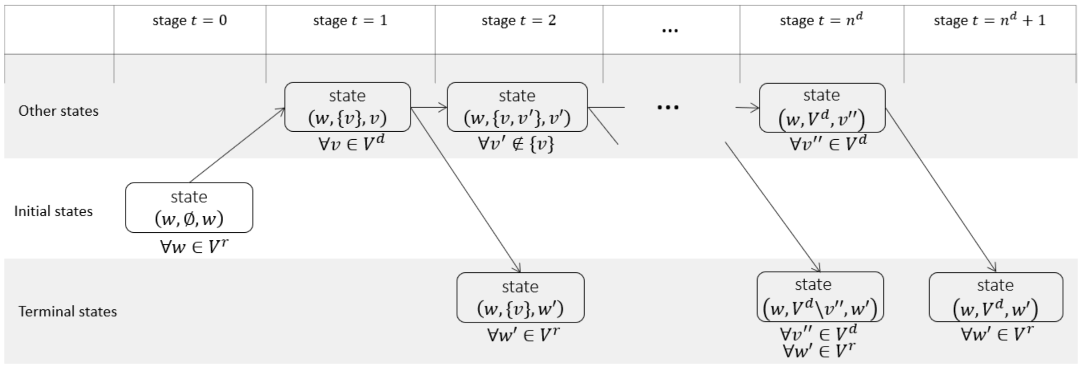
Figure 5.2: Example of a $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-invalid operation $o = wsw', s = (v_1, v_2, v_4, v_5)$ for $p = 2$



Note. Placing destination $v_3 \notin s$ before operation o in a drone tour π_d forces the crossing of the matching arrows from v_3 and v_1, with $|3-1| \geq p$. Placing destination $v_3 \notin s$ after operation o in a drone tour π^d forces the crossing of the matching arrows from v_3 and v_5, with $|3-5| \geq p$. This implies a violation of (5.1).

We have to recognize $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid operations efficiently already during the construction of the ops graph, to ensure that the algorithm remains polynomial. However, it is a nontrivial task, since the sequence of destinations s of some operation wsw' may force *the remaining operations* to violate the Balas-Simonetti precedence relations (5.1). For instance in Figure 5.2, sequence of destinations $s = (v_1, v_2, v_4, v_5)$ does not violate (5.1) for $p = 2$ itself. Nevertheless, operation $o = wsw'$ is not $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid, i.e. no BS-R neighbor may contain this operation. This is because node v_3, which is excluded from o, cannot be assigned to any other operation without violating (5.1). Note that, the search procedure introduced in Balas (1999) and Balas and Simonetti (2001) does not provide any hints on detecting efficiently the precedence violations in this case.

Figure 5.3: A schematic illustration of the ops graph


stage $t = 0$
stage $t = 1$
stage $t = 2$
...
stage $t = n^d$
stage $t = n^d + 1$
Other states
Initial states
Terminal states
state $(w, \emptyset, w)$ $\forall w \in V^r$
state $(w, \{v\}, v)$ $\forall v \in V^d$
state $(w, \{v, v'\}, v')$ $\forall v' \notin \{v\}$
state (w, V^d, v'') $\forall v'' \in V^d$
state $(w, \{v\}, w')$ $\forall w' \in V^r$
state $(w, V^d \backslash v'', w')$ $\forall v'' \in V^d$ $\forall w' \in V^r$
state (w, V^d, w') $\forall w' \in V^r$

For the sake of clarity, we explain the ops graph in case of $e_{max} \to \infty$ first and discuss its complexity (Sections 5.3.2.1-5.3.2.2). Afterward, we show how to respect limited e_{max} in Section 5.3.2.3.

5.3.2.1 Architecture of the ops graph for unlimited flight time

States in ops graph $\mathcal{O}$ are arranged in stages $t \in \{0, 1, ..., n_d + 1\}$ and have the form (w, S, v) with $w \in V_r$ being the initial RL, $S \subseteq V_d$ being the set of destinations of the current operation visited so far and $v \in S \cup V_r$ referring to the current position of the drone (see Figure 5.3). Stage 0 consists of m initial states $(w, \emptyset, w), w \in V_r$ indicating the initial RL of the drone. State (w, S, v) with $v \in S$ belongs to stage $|S|$. Terminal states (w, S, w') with $w, w' \in V_r$ and $S \subseteq V_d, S \neq \emptyset$ correspond to completed operations and belong to stage $|S| + 1$.

By analogy with $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid operations, we define $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid states, which are

- all initial states,
- terminal states (w, S, w') such that there exists at least one associated $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid operation $o = wsw'$ with $\{s\} = S$,
- and all other states (w, S, v), for which there exists at least one associated $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid operation $o = wsw'$, $\{s\} = S$, with a visiting sequence of destinations s terminating in $v \in S$.

We construct the ops graph such that $\mathcal{O}$ contains *only* $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid states.

Transition arcs in $\mathcal{O}$ connect states between consecutive stages. Outgoing arcs from initial states $(w, \emptyset, w)$ income in states $(w, \{v\}, v)$ with $v \in V_d$ and have the cost of $c_d(w, v)$. Non-terminal states (w, S, v) with $v \in S$ are adjacent to the following groups of states:

- $(w, S \cup \{v'\}, v')$ with $v' \in V_d \setminus S$, and $v' \notin S$; the cost of the corresponding arcs equals $c_d(v, v')$ (the prolongation of the drone flight by one not yet visited destination),
- (w, S, w') with $w' \in V_r$; the cost of the respective arcs is $c_d(v, w')$ (closure of the operation).

Figure 5.3 depicts the overall architecture of the described state graph.

In Lemma 23, we explain, how to construct outgoing transition arcs from some $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid state efficiently, such that the head state of the transition arc is an $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid state as well. Since all the initial states $(w, \emptyset, w), w \in V^r$, as well as states on the first stage are $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid by definition, this lemma essentially explains, how to construct the ops graph efficiently in the forward direction.

Lemma 23. *For the BS-R neighborhood $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}(x,p)$ with $p \in \mathbb{N}$, consider non-terminal $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid state $(w,S,v) \in \mathcal{O}$ with $|S| \geq 1$. Consider numbers $m := \min\{j : v_j \in S\}$ and $M := \max\{j : v_j \in S\}$. Then, a corresponding state $(w, S \cup \{v'\}, v' = v_i)$ with $v' \in V_d \setminus S$ is $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid iff*

$$\begin{cases} i \in [M-p+1, \max\{M-1, m+2p-1\}] & \textit{or} \\ i \in [\max\{M+1, m+2p\}, M+p] \textit{ and } (v_j \in S \ \forall j \in [m+p, i-p]) & \end{cases} \tag{5.4}$$

Each corresponding terminal state (w,S,w') with $w' \in V_r$ is $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid.

Proof. See Supplement 5.8.1.1. □

We compute all shortest paths between initial and terminal states in $\mathcal{O}$ by applying the following Bellman's equation in the forward induction manner for states $(w,S,v) \in \mathcal{O}$, $v \in S \cup V_r$:

$$\zeta(w,S,v) = \begin{cases} c_d(w,v) & \text{if } |S| \in \{0,1\} \\ \min\limits_{\substack{v' \in S\setminus\{v\}; \\ (w,S\setminus\{v\},v') \in \mathcal{O}}} \{\zeta(w, S\setminus\{v\}, v') + c_d(v',v)\} & \text{otherwise} \end{cases} \tag{5.5}$$

By construction, for any w, w', and $S \subseteq V_d$, the shortest path in $\mathcal{O}$ from the respective initial state $(w, \emptyset, w)$ to terminal state (w,S,w') represents an $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid operation $o = wsw'$ with $\{s\} = S$ (thus the visiting order) that has the shortest drone flight time, if such an operation exits.

5.3.2.2 Computational complexity of the ops graph for unlimited flight time

Proposition 12. *Consider the BS-R neighborhood $\mathcal{N}_{BS-R}(x,p)$. The construction of the ops graph is polynomial in the input size of the considered DRP-E instance (but exponential in parameter p), and requires a worst-case computational time of $O\left(4^p\left(n_r^2 n_d^2 p + n_r n_d^2 p^2\right)\right)$.*

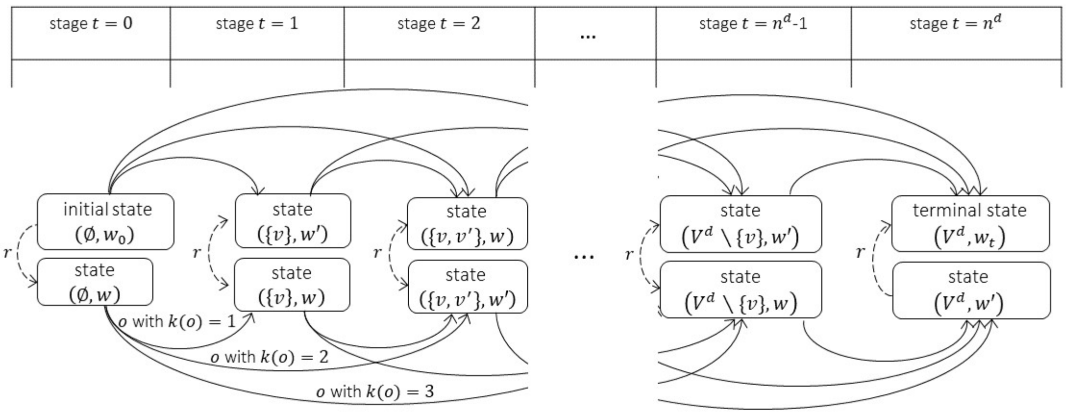
Proof. The described dynamic programming approach implied by equations (5.5) has linear time complexity in the number of transition arcs in ops graph $\mathcal{O}$. There are no outgoing arcs from terminal states. Lemma 23 bounds the number of transition arcs from each non-terminal state by $n_r + 2p$: There are n_r transitions to terminal states and at most $2p-1$ transitions to non-terminal states. Lemma 26 from Supplement 5.8.1.2 reports that the total number of non-terminal states in the ops graph is $O(n_d^2 n_r p \cdot 4^p)$. Putting this together, we get the stated complexity. □

5.3.2.3 Ops graph with energy constraints

In the case of limited maximal flight time e_{max}, we prohibit transitions that lead to (energy-)infeasible operations. We use triangular inequalities to perform this in a forward-looking manner. For each non-terminal state $(w,S,v) \in \mathcal{O}$, we consider the transition to a corresponding non-terminal state $(w, S \cup \{v'\}, v' = v_i)$ with $v' \in V_d \setminus S$ only if the flight time does not exceed e_{max}:

$$\zeta(w,S,v) + c_d(v,v') + \min_{w'' \in V_r}\{c_d(v',w'')\} \leq e_{max} \tag{5.6}$$

Figure 5.4: A schematic illustration of the meta state graph



The (energy-)feasibility check of the transition arcs to terminal states (w, S, w') is straightforward.

Note that the verification of the energy constraints does not change the time complexity stated in Proposition 12. The distance to the closest RL $\min_{w \in V_r}\{c_d(v, w)\}$ can be pre-computed for every destination $v \in V_d$ in runtime $O(n_d n_r)$, such that feasibility checks require $O(1)$ for each transition.

5.3.3 Meta state graph

Based on the ops graph, we construct the meta graph to evaluate feasible drone tours $\pi_d \in \mathcal{N}_{\mathcal{BS}-\mathcal{R}}(x, p)$. Let *operation set* $O = wSw', S \subseteq V_d$ denote the set of all feasible, $\mathcal{N}_{BS-R}$-valid operations $o = wsw'$ with $S = \{s\}$ for given neighborhood $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}(x, p)$. Let $C_d(O)$ denote the shortest drone flight time over all operations in O, if such exist. By construction, the costs $C_d(O), O = wSw'$, correspond to the costs of the terminal states (w, S, w') of the ops graph. We use these costs from the ops graph as an input to compute $\mathcal{OP}^*_{BS-R}$ – the collection of all feasible $\mathcal{N}_{BS-R}$-valid operation sets associated with their makespan $\mathcal{M}(O) = \max\{c_p(w, w'), C_d(O)\}$ for $O = wSw'$. Let also denote $\mathcal{TL}$ as the collection of all traveling legs $r \in V_r \times V_r$ associated with their makespans. Then the meta graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ enumerates all sequences of operations in $\mathcal{OP}^*_{BS-R}$ and traveling legs in $\mathcal{TL}$ that form feasible drone tours $\pi^d \in \mathcal{N}_{\mathcal{BS}-\mathcal{R}}(x, p)$.

Similarly to the ops graph, the main challenge in the construction of the meta graph is to examine only those states, which are relevant for the construction of BS-R-neighbors of x. We call such states $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-*valid* and limit the state set $\mathcal{V}$ to $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid metastates only. In order to efficiently discard $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-*in*valid metastates during the construction procedure, we use a special encoding for the states. This encoding was originally proposed by Balas (1999) for the traveling salesman problem (TSP), but we extend it to our problem. Before we introduce this encoding, we outline the general architecture of the meta graph $\mathcal{G}$ with an intuitive straightforward description of the states in Section 5.3.3.1. In Section 5.3.3.2, we show how to formulate states in an equivalent way by using the mentioned encoding and state the time complexity of the meta graph.

5.3.3.1 Architecture of the meta graph

States in $\mathcal{G}$ are arranged in stages $k \in \{0, 1, 2, ..., n_d\}$. Each state (S, w) at stage $k := |S|$ describes the set of visited destinations $S \subseteq V_d$ and the drone's position at some RL $w \in V_r$, where it meets the MRS. Stage 0 consists of states $(\emptyset, w), w \in V_r$, where the *initial state* $(\emptyset, w_0)$ describes the start of each drone tour. At stage n_d, we find states $(V_d, w), w \in V_r$, including *terminal* state (V_d, w_t) that marks the end of each drone tour.

Transition arcs $\mathcal{A}$ in $\mathcal{G}$ do not necessarily connect metastates at the neighboring stages, but may span over several stages. The overall structure of $\mathcal{G}$ is depicted in Figure 5.4. Given state $(S, w) \in \mathcal{V}$, there is a transition to $(T, w') \in \mathcal{V}$ if and only if one of the cases below holds:

- $S = T$. Then the transition corresponds to a traveling leg $r = ww'$ and its weight is $\mathcal{M}(r)$.
- $S \subsetneq T$ and the respective operation set $O = wT \setminus Sw'$ is in $\mathcal{OP}^*_{BS-R}$. Then the transition weight is $\mathcal{M}(O)$.

By construction, each path in $\mathcal{G}$ corresponds to a feasible drone tour $\pi_d \in \mathcal{N}_{\mathcal{BS}-\mathcal{R}}(x, p)$, and the shortest of these paths corresponds to the best drone tour π_d^* in the considered BS-R neighborhood. We compute the shortest path with the Bellman's equations (5.7). Recall that it is sufficient to consider drone tours formed as an alternate sequence of operations and (possibly trivial) two-node traveling legs (see Section 5.2). This allows us to iterate through the stages of $\mathcal{G}$ in a forward induction manner, combining incoming operation arcs with exactly one two-node traveling leg:

$$\zeta(S, w) = \begin{cases} c_r(w_0, w) & \text{if } S = \emptyset \\ \min\limits_{w' \in V_r} \{\epsilon(S, w') + \mathcal{M}(r) \mid \text{s.t. } r := w'w \in \mathcal{TL}\} & \text{otherwise} \end{cases} \tag{5.7}$$

$$\text{with } \epsilon(S, w') = \min_{\substack{T \subsetneq S; \\ w'' \in V_r}} \{\zeta(T, w'') + \mathcal{M}(O) \mid \text{s.t. } (T, w'') \in \mathcal{V} \tag{5.8}$$

$$\text{and } O = w''\{S \setminus T\}w' \in \mathcal{OP}^*_{BS-R}\}$$

5.3.3.2 Alternative representation of states and complexity of the meta graph

In the following, we explain the encoding, which enables to construct $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid metastates only, so that the meta graph can be constructed in polynomial time. The transition arcs and the Bellman's equation remain as stated in Section 5.3.3.1.

The key idea is that we can uniquely describe set S of the $k = |S|$ *first* visited destinations of any BS-R neighbor π_d of destination sequence x as follows:

$$S = \{v_j : j \in ([k] \setminus S_k^+) \cup S_k^-\}, \text{ with} \tag{5.9}$$

$$S_k^- := \{l \in [n_d] : l \geq k+1, v_l \in S\} \text{ and } S_k^+ := \{h \in [n_d] : h \leq k, v_h \notin S\} \tag{5.10}$$

In other words, S_k^- refers to the destinations that are at positions later than k in x and are moved up (before or at position k) in $x' = \pi_d(V_d)$ of neighboring tour π_d. S_k^+ refers to the destinations that are at positions no later than k in x and are moved down (after position k) in $x' = \pi_d(V_d)$.

Example 4. *Consider the drone tour* $\pi_d = (w_0, w_2, v_4, v_1, v_5, w_1, w_1, v_2, v_3, w_2, w_t) \in \mathcal{N}_{\mathcal{BS}-\mathcal{R}}(x, 4)$, *with* $x = (v_1, v_2, v_3, v_4, v_5)$. *Then the set of* $k = 3$ *first visited destinations in* π_d *is* $S = \{v_4, v_1, v_5\}$, *which can be uniquely described by the index sets* $S_3^- = \{4, 5\}$, $S_3^+ = \{2, 3\}$.

With this new notation, we represent metastates as (S_k^-, S_k^+, w), where k refers to the stage number. In this encoding, $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid metastates and valid transitions between them have *the same structure* for all the stages, which solely depends on parameter p and not on the instance-specific parameters (see Propositions 13 and 14). As the result, these can be computed as a lookup table in the preprocessing, see Table 5.2.

Proposition 13. *For the BS-R neighborhood $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}(x,p)$, the encoded state (S_k^-, S_k^+, w) is a $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid metastate at stage $k \in [n_d]$ if and only if the following conditions hold:*

$$- \quad |S_k^-| = |S_k^+| \leq \frac{p}{2} \tag{5.11}$$

$$- \quad S_k^- \subseteq \{k+1, \ldots, k+p-1\},\ S_k^+ \subseteq \{k-p+2, \ldots, k\} \tag{5.12}$$

$$- \quad \max\{l : l \in S_k^-\} - \min\{h : h \in S_k^+\} < p \tag{5.13}$$

Proof. See Supplement 5.8.1.3. □

Different from the state graph presented in Balas (1999) and Balas and Simonetti (2001), the meta graph in VLNS allows not only transitions between states from *consecutive stages*, but allows transitions from states on one stage towards *any* larger stages, as long as there are associated feasible transitions. Thus, Proposition 14 introduces new transition rules for VLNS.

Proposition 14. *There is an arc between metastate $(S_k^-, S_k^+, w) \in \mathcal{V}$ at stage k and metastate $(S_{k+h}^-, S_{k+h}^+, w') \in \mathcal{V}$ at stage $k+h, h \geq 0$ if and only if one of the cases below holds:*

- *$h = 0$ and $S_{k+h}^- = S_k^-$, $S_{k+h}^+ = S_k^+$*
- *$h \geq 1$, operation set $O = wHw'$ that corresponds to the transition arc belongs to $\mathcal{OP}_{BS-R}^*$ and the following conditions hold simultaneously:*

$$- \quad \{j \in S_k^- : j > k+h\} \subseteq S_{k+h}^- \tag{5.14}$$

$$- \quad \{j \in S_k^- : j \leq k+h\} \cap S_{k+h}^+ = \emptyset \tag{5.15}$$

$$- \quad \{j \in S_{k+h}^+ : j \leq k\} \subseteq S_k^+ \tag{5.16}$$

Thereby $H = \{v_l \ : \ l \in I\}$, $I = (\{k+1, ..., k+h\} \cup S_k^+ \cup S_{k+h}^-) \setminus (S_{k+h}^+ \cup S_k^-)$.

Proof. see Supplement 5.8.1.3. □

The proof of Proposition 17 in Supplement 5.8.1.3 sketches the construction procedure for $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid metastates at any stage k based on the rules of Proposition 13: we basically have to enumerate all possible combinations of (S_k^-, S_k^+, w), which obey conditions (5.11)-(5.13). For example, for $p = 2$, S_k^- and S_k^+ may contain at most one element each (condition (5.11)), thereby $S_k^- \subseteq \{k+1\}$ and $S_k^+ \subseteq \{k\}$ (condition (5.12)); which results in two valid pairs of (S_k^-, S_k^+): $(\emptyset, \emptyset)$ and $(\{k+1\}, \{k\})$; so that we have $2 \cdot |V_r|$ metastates at each *typical* stage (i.e., neither the first nor the last stage). For fixed stage k, all these metastates can be constructed from a well-specified list of valid combinations (S_k^-, S_k^+). The first and second columns of the lookup table (see Table 5.2) provide an example of such list for $p = 4$. Metastates can then be computed by replacing the parameter value k of each valid (S_k^-, S_k^+) from this lookup table by the stage number and combining it with each $w \in V^r$.

Similarly, we can quickly identify all valid transitions in the meta graph based on pre-computed lists, which we construct from Conditions (5.14)-(5.16) of Proposition 14 (see the remaining columns of Table 5.2 for an example). A similar lookup table has been proposed in Balas and Simonetti (2001) for the simpler case of the TSP, with only one column for the transitions limited to neighboring stages.

Example 5. *Let $x = (v_1, v_2, v_3, v_4, v_5)$ and $p = 4$. Then there are 8 valid pairs of (S_3^-, S_3^+) at stage $k = 3$, which results in $8 \cdot n_r$ $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid metastates (see Table*

Table 5.2: Lookup table for $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid transitions for $p = 4$ between stage k and stage $k+h$, $h \leq 3$

No.	Valid pairs of (S_k^-, S_k^+)	Valid transitions stage $k+1$	stage $k+2$	stage $k+3$
1	$(\emptyset,\emptyset)$	1,5,6,7	1,3,4,5,6,7,8	1,2,3,4,5,6,7,8
2	$(\{k+1\},\{k-2\})$	1	1,5,6,7	1,3,4,5,6,7,8,
3	$(\{k+1\},\{k-1\})$	1,2	1,5,6,7	1,3,4,5,6,7,8
4	$(\{k+2\},\{k-1\})$	2,5	1,3,4	1,2,5,6,7
5	$(\{k+1\},\{k\})$	1,3,4	1,2,5,6,7	1,3,4,5,6,7,8
6	$(\{k+2\},\{k\})$	3,5,8	1,2,3,4	1,2,5,6,7
7	$(\{k+3\},\{k\})$	4,6,8	2,3,5,8	1,2,3,4
8	$(\{k+1,k+2\},\{k,k-1\})$	2,3	1,2	1,5,6,7

Note. The values of columns 2 and 1 list valid pairs of (S_k^-, S_k^+) and provide their IDs (numbering), respectively. For each valid pair (S_k^-, S_k^+), columns 3-5 list the IDs (cf. column 1) of valid pairs (S_{k+h}^-, S_{k+h}^+) which can be part of the head states of outgoing valid transition arcs in the meta graph.

5.2). For example, (S_k^-, S_k^+)-pattern no. 8 describes metastates $(S_3^- = \{4,5\}, S_3^+ = \{2,3\}, w), w \in V_r$, i.e. metastates in which destinations $\{v_1, v_4, v_5\}$ have been already visited and the drone subsequently meets the MRS in RL w. The third column of Table 5.2 states valid transitions to stage $k+1 = 4$. These are transitions to states $(S_4^- = \{5\}, S_4^+ = \{2\}, w'), w' \in V_r$ [(S_k^-, S_k^+)-pattern no. 2] and $(S_4^- = \{5\}, S_4^+ = \{3\}, w'), w' \in V_r$ [(S_k^-, S_k^+)-pattern no. 3].

We conclude by stating the computational complexity of the meta graph in Proposition 15.

Proposition 15. *Consider BS-R neighborhood $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}(x,p)$. Given the collection of feasible, $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid operations $\mathcal{OP}^*_{BS-R}$, the best feasible drone tour $\pi_d^* \in \mathcal{N}_{\mathcal{BS}-\mathcal{R}}(x,p)$ can be determined with the worst case computational complexity of $O(n_d^2 n_r^2 \cdot 4^p)$.*

Proof. The best tour $\pi_d^* \in \mathcal{N}_{\mathcal{BS}-\mathcal{R}}(x,p)$ is determined by solving the shortest path problem in meta graph $\mathcal{G}$ with the Bellman's equations (5.7). The complexity of this procedure is linear in the number of transition arcs, which is the highest for $e_{max} \to \infty$. In this case, all outgoing arcs for each state are exactly those listed in the lookup table (cf. Table 5.2). By Proposition 17 from Supplement 5.8.1.3, there are $\mathcal{O}(n_d n_r \cdot 2^p)$ $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid metastates. For increasing h, the list of valid transitions for each metastate at stage k towards stage $k+h$, converges to the complete set of valid metastates at stage $k+h$ (see Table 5.2). From this follows the approximation of the time complexity. □

5.3.4 Time complexity of the developed VLNS approach

Finally, we can combine the insights from the previous sections to receive the overall time complexity of VLNS:

Theorem 9. *For any given instance of DRP-E, sequence x of destinations and given p, a best feasible drone tour $\pi_d^* \in \mathcal{N}_{\mathcal{BS}-\mathcal{R}}(x,p)$ can be found in the worst-case runtime of $O(p^2 n_r^2 n_d^2 \cdot 4^p)$.*

Proof. The time complexity of VLNS sums up from the complexities to construct the ops graph at the first stage and the meta graph at the second stage, which are $O\left(4^p\left(n_r^2 n_d^2 p + n_r n_d^2 p^2\right)\right)$ (see Proposition 12) and $O(n_d^2 n_r^2 \cdot 4^p)$ (see Proposition 15), respectively. This results in $O(p^2 n_r^2 n_d^2 \cdot 4^p)$ for VLNS. □

Computational experiments of Section 5.5.3 show that the bound in Theorem 9 is pessimistic in case of a realistic battery capacity of the drone: With each increase of p by 1, the observed runtime of VLNS about doubles.

5.4 VLNS-based algorithms and extensions

5.4.1 Relevant DRP-E variants and additional problem features

The proposed VLNS approach can be applied to a range of DRP-E variants. In this section, we discuss how to incorporate selected additional or alternate problem features to VLNS, that are widespread in the drone routing literature and practice. The following extensions can be applied straightforwardly and do not affect the overall runtime complexity of VLNS stated in Theorem 9:

Drone hovering while waiting. In many drone applications, the drone must wait for the MRS in a hovering state, to prevent vandalism or theft, thereby consuming energy at a specific rate r_{hov}. In VLNS, this is incorporated by further reducing the collection $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid operation sets $\mathcal{OP}^*_{\mathcal{BS}-\mathcal{R}}$, see Section 5.3.3 with the following feasibility check of cost $O(1)$ for each operation set $O = wSw'$,

$$\max\{c_p(w,w') - C_d(O), 0\} \cdot r_{hov} + C_d(O) \leq e_{max} \tag{5.17}$$

Non-zero service times. Service times $s(v) > 0$ at destinations $v \in V_d$, e.g. for drone delivery- or surveillance, as well as replenishment times $c_{swap} > 0$, e.g. for charging or battery swaps, can be straightforwardly embedded as in the transition weights of the meta graph. If the service times are performed in flight, these must be added as summands to the feasibility checks of Section 5.3.2.3. Furthermore, the MRS may perform a replenishment in motion during a traveling leg (w, w'), whose makespan then augments to $\max\{c_{swap}, c_r(w, w')\}$.

Stationary replenishment stations. We can also apply our method to the case of stationary replenishment stations (instead of MRS) by setting the travel time parameters c_r to zero and prohibiting the so-called traveling legs.

Some practice applications face *complex energy- or resource consumption* of the drone. For instance, changes in altitude in the drone tour might lead to an energy consumption function that is non-linear in flight time. In package delivery applications (cf. Section 5.5.4.2), the energy consumption rate may depend on the current payload of the drone. Furthermore, a limited payload capacity of the drone might act as a further constraining resource. Also in these cases, the overall logic of VLNS remains. In the ops graph, the Bellman equations (5.5) are replaced by a label setting algorithm (cf. Pugliese & Guerriero, 2013) that conserves all Pareto-optimal time-, energy-, and possibly other resource values in every state, which invalidates the time complexity of the ops graph from Proposition 12. The construction of the meta-graph, which is still, in most practice implementations, the more time-consuming stage of the VLNS approach, remains unaffected by this extension and pseudo-polynomial in complexity. Note that the feasibility checks of Section 5.3.2.3 remain in place, for complex energy consumption and other resources, as long as these resources are additive along the arcs of an operation in a *forward* motion. This is not the case in a delivery application where packages to be delivered to the destinations are loaded at every RL, as the payload then in *decreases* along the arcs. This can be fixed by solving with VLNS the *inverted* problem, making it a package pick-up problem, and reversing the received solution.

Most of these extensions are analyzed in Section 5.5. In many realistic applications, there is just one single depot where the drone starts and ends its tour, i.e. $w_0 = w_t$.

Therefore, Online Supplement 5.8.3 comments on how to extend the BS-R neighboorhood by some additional promising solutions in this case in a way that conserves the pseudo-polynomial computational complexity of the neighborhood search.

5.4.2 A VLNS-based local search

The presented neighborhood search, VLNS, should not be perceived as a stand-alone solution algorithm. Instead, it is an *improvement procedure* which can be embedded flexibly in *any* algorithmic scheme, meta- or math-heuristic for DRP-E and its variants. The application of VLNS is especially effective as an *intensification procedure* around promising incumbent solutions.

In our computational experiments (see Section 5.5), we test the performance of VLNS by simply embedding it into a standard *best improvement local search* procedure (see e.g. Taillard, 2023). We initiate the local search with the best visiting sequence of destinations, x_{best}, received after performing an adaptive large neighborhood search (ALNS) with a given time limit. We refer the reader to Voigt (2024) for details on the outline of general ALNS approaches.

The initializing ALNS proposed in this paper is problem-specific and will be dubbed ALNS* in the following. ALNS* operates on the space of destination sequences rather than the solution space of DRP-E itself. For the exploration of different destination sequences, we propose a novel type of operators, which populate the current candidate sequence with RLs and remove them shortly afterward. ALNS* itself is initialized with the shortest Hamiltonian path through all destinations and the depot, x_{TSP}. To evaluate the quality of the current destination sequence in each iteration, replenishments are inserted optimally, e.g. by applying VLNS with $p = 1$, and the objective value of the resulting feasible DRP-E solution is compared to the current best objective value z_{best}. The pseudocode of Algorithm 3, provides a detailed description of ALNS*. We refer to the VLNS-based local search, initialized with ALNS*, as VLNS-LS^{I*} in the following.

Algorithm 3 Initialization procedure ALNS*

```
1: **Input:** $x_{\text{TSP}}$
2: Set $x_{\text{best}} \leftarrow x_{\text{TSP}}$;
3: Set $z_{\text{best}}, \pi_{\text{best}} \leftarrow \text{RTS}(x_{\text{best}})$; # apply RTS to $x_{\text{best}}$, $z_{\text{best}}$ and $\pi_{\text{best}}$ are the resulting objective value and drone
   tour, resp.
4: Set $s' \leftarrow$ multiset of RL's that are visited in $\pi_{\text{best}}$; $s_{\text{best}} \leftarrow s'$;
5: **while** time < time limit **do**
6:     $\pi' \leftarrow \text{TSP}(V_d \cup s')$; # Compute shortest Hamiltonian path through destinations and set of RL's $s'$
7:     $x' \leftarrow \pi' \setminus V_r$; # Remove RL's from $\pi'$ to get new sequence of destinations
8:     $z', \pi' \leftarrow \text{RTS}(x')$;
9:     **if** $z' < z_{\text{best}}$ **then**
10:        $z_{\text{best}} \leftarrow z'$; $x_{\text{best}} \leftarrow x'$; $\pi_{\text{best}} \leftarrow \pi'$;
11:        Set $s' \leftarrow$ multiset of RL's that are visited in $\pi_{\text{best}}$; $s_{\text{best}} \leftarrow s'$;
12:    **else**
13:        $s' \leftarrow$ perturbation($s_{\text{best}}$);
14:    **end if**
15: **end while**
16: **Return:** $x_{\text{best}}, z_{\text{best}}, \pi_{\text{best}}$
```

Note: perturbation(s_{best}) is in this case a random removal of RL's from s_{best}, followed by a random insertion of RL's from V_r

5.4.3 An exact solution approach for DRP-E

One important aspect of VLNS is its scalability with the parameter p: The bigger p, the larger the neighborhood, but also the larger the computational time needed to examine it. With $p \to n_d$ VLNS converges to an exact approach for DRP-E.

A more time-efficient exact solution approach for DRP-E, which we dub *DP-ILP*, can be derived directly from the two-stage structure of VLNS (see Section 5.3) by dropping the restriction to $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid states in the ops graph, and replacing the meta graph with an *integer linear program (ILP)*.

The resulting *exact dynamic programming (DP)* formulation in the ops graph has the worst-case runtime of $O(n_r n_d(n_d + n_r) \cdot 2^{n_d})$: Consider $O(n_r n_d \cdot 2^{n_d})$ non-terminal states of the form $(w, S, v), w \in V_r, S \subseteq V_d, v \in V_d$, each having up to $O(n_r)$ arcs to terminal states for the closure of the current operation at an RL, and, $O(n_d)$ arcs to non-terminal states, for the extension of the operation towards a destination. The feasibility checks of Section 5.3.2.3 are conserved, making the exact DP in the ops graph very efficient in most practical applications, since realistic energy capacities significantly reduce the length of feasible operations and thus, the actual complexity of the ops graph. The resulting set of feasible operations $\mathcal{OP}^*$ (see Section 5.3.3) is then provided to the second stage of the procedure.

In this stage, we could indeed proceed similarly, by dropping the restriction to $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid states in the meta graph of Section 5.3.3. However, the resulting DP-formulation in the meta graph is of large complexity (of exponential order 3^{n^d}) and becomes untractable for large instance sizes. We found it more efficient to construct the optimal drone tour with an ILP that is inspired by Agatz et al. (2018). Therefore, the feasible operation sets in $\mathcal{OP}^*$ and the set of traveling legs $\mathcal{TL}$ are combined to a new set $\mathcal{L}$ of feasible transitions between RLs, representing the main decision variables of ILP. The resulting program resembles the shortest path problem with set covering constraints: Find the shortest path between the initial and the target depots using arcs from $\mathcal{L}$ such that each destination is covered exactly once. Table 5.3 summarizes the additional notation and equations (5.18)-(5.26) the ILP formulation for the most frequent variant of DRP-E, with only one single depot w_0.

Table 5.3: Notations used in the ILP formulation

Sets and locations	
$\mathcal{L}$	Collection of all feasible operation sets and recharging legs
$\mathcal{M}(l)$	Makespan of $l \in \mathcal{L}$
$\mathcal{L}(v)$	For $v \in V_d$, $\mathcal{L}(v)$ is the collection of operation sets that cover v, i.e. $O = wSw' \in \mathcal{L}(v)$ if and only if $v \in S$
$\mathcal{L}^+(w)$	For $w \in V_r$, $\mathcal{L}^+(w)$ is the set of transitions $l \in \mathcal{L}$ that start in w.
$\mathcal{L}^-(w)$	For $w \in V_r$, $\mathcal{L}^-(w)$ is the set of transitions $l \in \mathcal{L}$ that end in w.
$\mathcal{L}^+(T)$	For $T \subset V_r$, $\mathcal{L}^+(T)$ is the set of transitions $l \in \mathcal{L}$ that start in some $w \in T$ and end in some $w' \in V_r \setminus T$
Decision variables	
x_l	For each $l \in \mathcal{L}$, $x_l = 1$ if operation set or recharging leg l is part of the drone tour, and $x_l = 0$ else
y_w	For each $w \in V_r$, $y_w = 1$ if w is part of the MRS's tour, and $y_w = 0$ else

$$\text{Minimize} \sum_{l \in \mathcal{L}} \mathcal{M}(l) \cdot x_l \tag{5.18}$$

s.t.

$$\sum_{l \in \mathcal{L}(v)} x_l \geq 1 \qquad \forall v \in V_d \tag{5.19}$$

$$\sum_{l \in \mathcal{L}^+(w)} x_l = \sum_{l \in \mathcal{L}^-(w)} x_l \qquad \forall w \in V_r \tag{5.20}$$

$$\sum_{l \in \mathcal{L}^+(w_o)} x_l \geq 1 \tag{5.21}$$

$$y_{w_0} \geq 1 \tag{5.22}$$

$$\sum_{l \in \mathcal{L}^+(w)} x_l \leq 2 \cdot n_d \cdot y_w \qquad \forall w \in V_r \tag{5.23}$$

$$\sum_{l \in \mathcal{L}^+(T)} x_l \geq y_w \qquad \forall T \subseteq V_r \setminus \{w_0\}, \forall w \in T \tag{5.24}$$

$$x_l \in \{0,1\} \qquad \forall l \in \mathcal{L} \tag{5.25}$$

$$y_w \in \{0,1\} \qquad \forall w \in V_r \tag{5.26}$$

The objective function (5.18) minimizes the makespan of the drone tour, constraints (5.19) assure that the drone covers every destination $v \in V_d$. Constraints (5.20) are flow constraints for the MRS. Constraints (5.21) and (5.22) include the depot w_0 in the MRS's tour. Constraint (5.23) establishes the relation between x - and y- variables, thereby, note that $2 \cdot n_d$ is an upper bound for the amount of operation sets and recharging legs starting in $w \in V_r$. Equations (5.24) are *connectivity constraints* of the MRS's tour. One important factor of the ILP's success is to add these constraints gradually during the optimization process in the form of *lazy constraints*. Constraints (5.25) and (5.26) define the binary decision variables.

5.5 Computational experiments

The aim of the presented computational study is three-fold. On a structured, well-balanced dataset for the basic DRP-E, presented in Section 5.5.1, we first validate in Section 5.5.2 the performance of the developed VLNS-LS^{I*} algorithm and the exact DP-ILP approach. On the same dataset, we analyze the performance of algorithmic elements of VLNS-LS^{I*}, specifically, the dependence on the initial sequence of destinations and the impact of the scaling parameter p, see Section 5.5.3. Thirdly, in Section 5.5.4, we demonstrate the generalizability of VLNS to different DRP-E variants from the literature and show that VLNS-based approaches can outperform the state-of-the-art heuristics for these problems. In the same section, we also see that DP-ILP finds new optimal solutions for larger problem sizes, compared to the original literature.

Unless stated otherwise, we set $p = 4$ in VLNS. For the experiments, we used an Intel Xeon(R) Gold 6248R CPU 3.00GHz $\times$ 96 processor. The algorithms are coded in Python 3.9 without parallelization. The integer programs are solved with Gurobi 10.0.1 using 8 logical CPU cores.

5.5.1 Tailored data generation procedure

We propose a data set structured around a wide range characteristics that can occur in DRP-E. In the generated instances, a given number of destinations n_d is randomly and

Figure 5.5: Basis setting

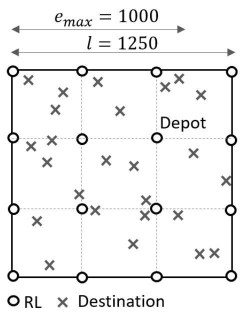


Table 5.4: The nine settings in both data sets Medium and Large

Replenish-ment	Setting	MRS speed δ	Energy capacity e_{max}	Location ratio $\frac{n_d}{n_r}$	Density d $(\times 10^{-6})$
	Basis	$^1/_2$	1000	2	16
Expensive	MRSSpeedLow	**$^1/_3$**	1000	2	16
	EnergyLow	$^1/_2$	**750**	2	16
	RLDensityLow	$^1/_2$	1000	**3**	16
	DensityLow	$^1/_2$	1000	2	**8**
Low-cost	MRSSpeedHigh	**1**	1000	2	16
	EnergyHigh	$^1/_2$	**1250**	2	16
	RLDensityHigh	$^1/_2$	1000	**1**	16
	DensityHigh	$^1/_2$	1000	2	**45**

uniformly scattered in a square $l \times l, l \in \mathbb{N}$. The placement of RLs should ensure the feasibility of instances. Unfeasible instances occur when at least one destination has no close-by RL, such that the battery life of the drone is insufficient even for a visit in a direct return flight. Possible workarounds in the literature include dismissal of infeasible instances (cf. Poikonen & Golden, 2020) or calculation of e_{max} as a function of the realized positions of $v \in V_d$ and $w \in V_r$ (P. Gonzalez-R et al., 2020). For the sake of data generation transparency, and similar to Karak and Abdelghany (2019), we decided to place RLs in the nodes of a uniform grid instead (see Figure 5.5), having as a consequence that the number of RLs should possibly be *quadratic* (e.g. 9=3x3, 16=4x4, 100=10x10). A randomly selected RL is set to be both the initial and target depot of the drone's and MRS's routes. We use the Euclidean metric to measure the distances flown by the drone. Because the MRS is usually restricted to moving on a street network, we use the Manhattan metric for the MRS. The chosen metrics are common in the drone routing literature (cf. Betti Sorbelli et al., 2023; Dell'Amico et al., 2020; Ha et al., 2018; Murray & Chu, 2015).

We generate two data sets:

- Medium-sized data set (*Medium*) contains instances with $n_d := 25$ destinations and $n_r \in \{9, 16, 25\}$ RLs, thus up to 50 locations in total.
- Large-sized data set (*Large*) contains instances with $n_d := 100$ destinations and $n_r \in \{36, 49, 100\}$ RLs, thus up to 200 locations in total.

Each data set contains 180 instances grouped in 9 settings with 10 instances each.

Since the performance of solution algorithms depends more on the ratio of the vehicle speeds, and the relative drone ratio compared to the application area, rather than their absolute values, we normalize the drone speed to 1 and condense further instance characteristics to a few essential details. Such normalization makes our results more informative

Table 5.5: Performance of VLNS-LS^{I*} and DP-ILP on Medium dataset

		Gap to optimality of VLNS-LS$^{I^*}$			Avg. runtime (s)	
Replenish-ment	Setting	Avg. gap (%)	Times gap $\leq 1\%$	Times gap $\leq 5\%$	VLNS-LS$^{I^*}$	DP-ILP
	Basis	1.6	7/10	8/10	27	33
Expensive	MRSSpeedLow	2.9	4/10	8/10	26	42
Expensive	EnergyLow	1.0	7/10	9/10	23	3
Expensive	RLDensityLow	4.2	4/10	8/10	23	1
Expensive	DensityLow	2.1	6/10	9/10	23	2
Low-cost	MRSSpeedHigh	0.5	9/10	10/10	25	24
Low-cost	EnergyHigh	1.4	7/10	9/10	31	165
Low-cost	RLDensiyHigh	0.1	9/10	10/10	32	657
Low-cost	DensityHigh	1.1	6/10	10/10	37	4472
	Total	1.7	59/90	81/90	27	600

Note. DP-ILP could solve all 90 instances to proven optimality. VLNS-LS^{I*} was initialized with 20s of ALNS*.

in view of the ever-changing characteristics of young drone technology and different application scenarios. We construct our datasets around a basic setting, called *Basis* (cf. Table 5.4), for which, when setting the *time unit* $TU := 2$ seconds and the *length unit* $LU := 20$ m, we receive the same vehicle speeds of 10m/sec and 5m/sec, as well identical location density as considered by Poikonen and Golden (2020), combined with a realistic maximal drone flight time of $e_{max} = 33$ minutes (cf. Stolaroff et al., 2018).

We control the following factors in our settings and use a one-factor-at-a-time design around the basic settings (marked in bold):

- *MRS* speed $\delta \in \{\mathbf{1}, \frac{1}{2}, \frac{1}{3}\}$ *TU/LU*. The lower is δ, the larger the impact of the waiting times for the MRS on the routing decisions of the drone.
- *Energy capacity* $e_{max} \in \{750, \mathbf{1000}, 1250\}$ *TU*.
- *The ratio of the number of destinations and RLs* $\frac{n_d}{n_r} \approx 1, 2$ or 3. Given $n^d = 25$ for Medium and $n_d = 100$ for Large, we set $n_r \in \{9, \mathbf{16}, 25\}$ for Medium and $n_r \in \{36, \mathbf{49}, 100\}$ for Large to achieve these values.
- *The number of destinations per square unit (density)* $d \in \{8, 16, 45\} \times 10^{-6}\ LU^{-2}$. To achieve these values, we set $l \in \{750, \mathbf{1250}, 1750\}$ LU for Medium and $l \in \{1500, \mathbf{2500}, 3500\}$ LU for Large.

The eight additional settings to the *Basis* setting are labeled *MRSSpeedLow*, *EnergyLow*, *RLDensityLow*, *DensityLow*, *MRSSpeedHigh*, *EnergyHigh*, *RLDensityHigh* and *Density-High*. The former four settings describe *expensive replenishment*, because each replenishment stop of the drone is more costly than in *Basis*, as it potentially has to wait for a slow MRS, can visit less destinations between subsequent replenishments, or has to fly further to reach an RL. Similarly, the four latter settings describe *low-cost replenishment*.

The data sets are included in the Online Companion of this paper.

5.5.2 Performance analysis of the VLNS-based local search and DP-ILP

Tables 5.5 and 5.6 show the performance of the proposed solution approaches.

First, observe that DP-ILP could solve *all* Medium instances, across *all* settings, with an impressive number of 31-50 locations in total, to *proven optimality*. For expensive replenishment settings EnergyLow, RLDensityLow and DensityLow, all optima could even be found within a maximum of 4s. Strikingly, DP-ILP's efficiency pertains also for Large

Table 5.6: Performance of VLNS-LS^{I*} and DP-ILP on Large dataset

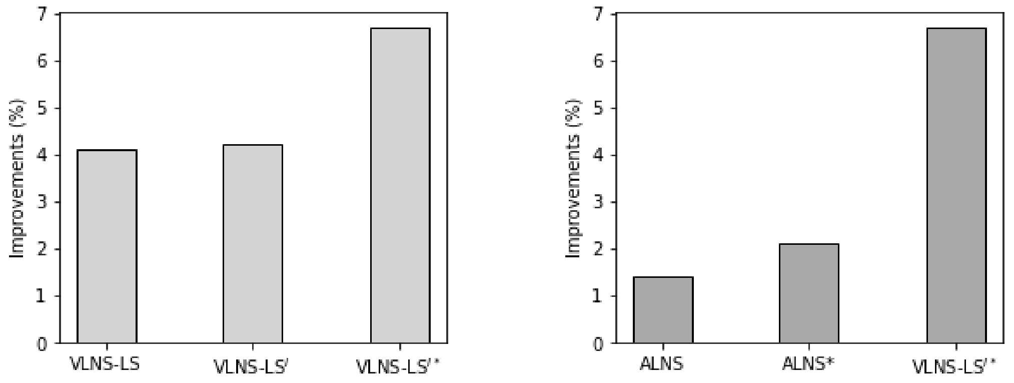
		Performance of DP-ILP			Performance of VLNS-LS$^{I^*}$			Avg. runtime (s)	
Replenish -ment	Setting	Times $\sim$ optimal	Avg. gap(%)	Times gap$\leq 15\%$	Avg. gap(%)	Times gap$\leq 5\%$	Times gap$\leq 15\%$	VLNS-LS$^{I^*}$	DP-ILP
	Basis	1/10	31.2	6/10	1.5	9/10	10/10	1189	1800
Expensive	MRSSpeedLow	0/10	31.3	7/10	2.3	7/10	10/10	1233	1800
	EnergyLow	9/10	0.0	10/10	7.1	2/10	10/10	1142	1471
	RLDensityLow	9/10	0.5	10/10	11.4	1/10	8/10	1132	1381
	DensityLow	10/10	0.0	10/10	9.4	0/10	10/10	1090	796
Low-cost	MRSSpeedHigh	10/10	0.2	10/10	0.8	10/10	10/10	1263	1682
	EnergyHigh	0/10	>100	0/10	0.0	10/10	10/10	1566	1800
	RLDensiyHigh	0/10	>100	0/10	0.0	10/10	10/10	2034	1800
	DensityHigh	0/10	> 100	0/10	0.0	10/10	10/10	1770	1800
	Total	39/90	> 100	53/10	3.6	59 /90	88 /90	1380	1576

Note. DP-ILP is $\sim$ optimal when the optimality gap of the solver $\leq 5\%$; gap refers to the gap to the best known solution, i.e. the minimum of the VLNS-LS^{I*} and DP-ILP objective values. A time limit of 1800s was set to DP-ILP. VLNS-LS^{I*} was initialized with 900s of ALNS*.

instances. In the same three settings, it handles successfully the astonishing total of 149 locations within the runtime limit of 1800s, closing the optimality gap of the optimization solver down to less than 5% in 28 out of the 30 associated instances. This clearly eliminates the need for heuristic approaches in this group of settings. The efficacy can be attributed to the relatively small number of feasible operations found by the exact DP approach in the ops graph, which provide the variables to the ILP solver; e.g., 1562 on average for Medium in the setting DensityLow, compared to 7737 in Basis. Note that, the same does not hold for MRSSpeedLow, since the MRS speed does not affect the feasibility of operations, given that in basic DRP-E, the drone waits without consuming energy.

Table 5.5 furthermore demonstrates the excellent performance of the proposed VLNS-LS^{I*} on Medium instances, showing an overall average gap to optimality of 1.7% and close-to-optimal solutions, with optimality gaps of no more than 5%, in 90% of all tested instances. On Large, in the setting MRSSpeedHigh, all soft-optima (solver interrupted with optimality gap of $\leq 5\%$) are known. In this setting, the VLNS-LS^{I*} solutions showed an average gap of 0.8% to the DP-ILP solutions, encouraging optimism regarding close-to-optimal solutions of VLNS-LS^{I*} also in other low-cost replenishment settings and the Basis. In these settings, where DP-ILP was not effective on Large, VLNS-LS^{I*} rendered the best-known solution in 97.5% of the cases. In the three expensive replenishment settings EnergyHigh, RLDensityHigh and DensityHigh, the gaps of VLNS-LS^{I*} on Large increase to moderately good values between 5% and 15% in most cases. This reminds us of the fact that VLNS is an *improvement procedure* which strongly depends on the initial visiting sequence of destinations. The initialization procedure ALNS* presented in this paper (see Section 5.4.2) builds on shortest Hamiltonian paths for the drone only. Despite also considering the RLs, it cannot fully replicate the tight clustering of drone sorties around these locations or fully capture the critical synchronization with the MRS. In order to find better gaps with VLNS-based approaches, further research on good initial sequences with strong replenishment focus may be needed. However, we remind the reader that the alternate *exact* approach proposed in this paper, DP-ILP, is already sufficiently fast and reliable in these settings, questioning if these further enhancements of VLNS-LS^{I*} are truly warranted. Finally, note that VLNS-LS^{I*} shows a stable runtime over all instances, of around 30s and 1400s for Medium and Large, respectively.

Figure 5.6: Avg. improvements compared to route-first-split-second heuristic in Basis of Large



(a) Importance of initial destination sequence **(b)** Comparision to ALNS-benchmarks for equal runtime

5.5.3 Analysis of algorithmic elements in the VLNS-based approach

In this section, we analyze the performance factors of the proposed VLNS-LS^{I*}. First, we discuss the importance of the initial destination sequence, in the example of Basis on Large, see Figure 5.6a. We compare VLNS-LS^{I*}, initialized with 900s of the newly developed ALNS* (see Section 5.4.2), with the local search VLNS-LS^{I}, that was initialized for 900s with a simpler adaptive large neighborhood search, simply dubbed ALNS. The latter starts with the shortest Hamiltonian sequence of all destinations x_{TSP}, and uses classical destroy- and repair operators on to perturb this sequence. Specifically, ALNS selects by chance either the random removal of 1-3 destinations or the random removal of a sequence of 1-3 destinations from the current candidate sequence (see Voigt, 2024), then re-inserts them randomly. Each new destination sequence is evaluated by inserting RL's optimally, and the candidate solution is updated when the resulting makespan is better than the current upper bound. Figure 5.6a additionally depicts the simple local search VLNS-LS, which takes x_{TSP} as an input and skips the initialization phase. The three algorithms are compared based on their improvement of the simple route-first-split-second heuristic as proposed, e.g., by Poikonen and Golden (2020). The results confirm that the selection of a good initial sequence for the VLNS-based local search can significantly boost the quality of the approach: by using the more effective initialization procedure ALNS*, VLNS-LS^{I*} could increase the improvement by 2.5 % and 2.6%, respectively, compared no initialization (VLNS-LS) and simple initialization (VLNS-LS^{I}).

Taking the same basis for comparison, Figure 5.6b addresses the question, of whether simpler metaheuristics without the VLNS could achieve the same level of improvement than VLNS-LS^{I*} when applied for the same runtime. Specifically, we performed ALNS* and ALNS (introduced in the previous paragraph) to Basis on Large for the VLNS-LS^{I*} runtime. The results suggest, that the use of VLNS significantly increases the improvement potential. Specifically, VLNS-LS^{I*} reached a 4.6 % and 5.3% higher improvement compared to ALNS* and ALNS, respectively. Furthermore, the VLNS-based approach achieved a better improvement compared to both ALNS-based approaches in *all* 10 instances. Also, VLNS-LS^{I*} improved the route-first-split-second solution in all the instances, while ALNS* and ALNS fail to achieve *any* improvement in 4 and 2 out of 10 instances, respectively. The reason is that ALNS* and ALNS examined only 44 and 2021 destination sequences, respectively, on average within their runtime of more than 1300s. On the other side, as discussed in Section 5.3, VLNS-based approaches nontrivially explore exponentially many promising solutions in an efficient manner.

In the following, we quantify the number of feasible solutions examined by VLNS, as well as achieved solution qualities and runtimes at different values of p. For transparency, we report the results of VLNS in *one* BS-R neighborhood of the initial destination sequence x_{TSP}, without proceeding computations until a local optimum is found. Table 5.7 exactly enumerates sequences of destinations contained in one BS-R neighborhood for small values of p. Already for $n_d = 11$ and $p = 4$, VLNS examines more than 22,000 sequences of destination visits. Observe that for each such sequence, we insert RLs optimally. We get that VLNS spends much less time than splitting procedures to analyze one destination sequence, and explores significantly more solutions than ALNS and ALNS*, explaining its superiority.

Table 5.7: The number of destinations' sequences examined by VLNS

	# of destinations			
Parameter p	$n^d = 5$	$n^d = 7$	$n^d = 9$	$n^d = 11$
2	10	23	57	146
3	31	130	594	2,807
4	62	411	3144	22,728

Finally, Table 5.8 reports the trade-off between the computational time and the solution quality of VLNS with different values of p, in the Basis setting of Medium. With each increase of p by 1, the runtime about doubles on average. The results of the remaining settings are similar.

Table 5.8: Performance dynamics of VLNS, Basis setting of Medium

	Parameter $p =$									
	1	2	3	4	5	6	7	8	9	10
Avg gap (%)	6.3	5.7	4.3	4.0	3.2	3.2	2.7	2.1	1.8	1.7
Runtime (s)	0.5	0.8	1.5	2.8	5.4	10.7	21.8	46.2	102.7	239.0

5.5.4 Generalizability of VLNS to DRP-E variants

As discussed in Section 5.1.1, the closest problem formulations to the basic DRP-E are those of Poikonen and Golden (2020) and Zeng et al. (2022). Therefore, we selected the algorithms of these articles as a reference point for VLNS. We perform our tests on the *original* problem formulations from the respective publications for one truck and one drone, to demonstrate the generalizability of VLNS, and present our findings in Sections 5.5.4.1 and 5.5.4.2. Both formulations have a number of problem-specific extensions compared to the basic DRP-E, which we summarize in Table 5.9. VLNS can straightforwardly incorporate all of these extensions, as we showed in Section 5.4.1. Since the actual problem instances are not published, in both cases, we reproduce the data generation procedures of the articles and for transparency, publish all generated instances (720 and 160 instances, respectively) as well as the received solutions in the Online Companion.

5.5.4.1 Drone-lead surveillance DRP-E of Zeng et al. (2022)

Zeng et al. (2022) denote the studied drone-lead surveillance as the *nested VRP*. It has a number of simplifications and extensions compared to the basic DRP-E. On the simplifying side, the set of RLs coincides with the set of destinations ($V_r = V_d \cup \{w_0\}$), and neither the drone nor the truck can visit a location twice. The extensions refer to the following aspects: The drone should spend uninterrupted surveillance time at each destination; the

Table 5.9: Overview of DRP-E variants studied in computational experiments

	Problem features				
Variant	Application	Service times at $v \in V_d$	Drone hovers while waiting	Service time for replenishment	Energy consumption
Nested VRP by Zeng et al. (2022) (Section 5.5.4.1)	Surveillance	✓ surveillance times $s(v) > 0$	✓	✓ at RL or traveling leg	time-linear at rate r
MVDRP by Poikonen and Golden (2020) (Section 5.5.4.2)	Package delivery	✗	✓	✓ at RL	time-linear at rate $r(w_{\sigma(i)})$, $w_{\sigma(i)}$ is current load
Basic DRP-E (Section 5.5.2)	General	✗	✗	✗	time-linear at rate r

Table 5.10: Comparative performance of VLNS-LS and NS of Zeng et al. (2022) within same runtime

	Comparison of VLNS-LS relative to NS solutions				Comparison to the optimum			
Instance size n_d+1	Average improv.(%)	Best improv.(%)	Times improved	Times deteriorated	Average gap(%) VLNS-LS	NS	Times gap $\leq$ 3% VLNS-LS	NS
7-10	1.4	15.4	205/360	0/360	0.2	1.7	352/360	296/360
20	1.0	5.0	77/90	0/90	0.4	1.4	88/90	77/90
50	0.6	2.5	90/90	0/90	0.4	1.0	28/28	27/28
75	0.7	2.7	90/90	0/90	–	–	–	–
100	0.6	1.9	90/90	0/90	–	–	–	–
Total	1.08	15.38	552/720	0/720	0.3	1.6	468/478	400/478

Note. A time-limit of 3600s was set to DP-ILP. VLNS-LS was initialized with destination sequence x_{TSP}.

drone can wait for the truck only in the hovering state, which consumes energy; battery-swapping times c_{swap} are non-zero and can be performed either in a stationary state at RL's or in motion during a traveling leg.

Zeng et al. (2022) propose an iterated local search, dubbed *NS*. At each iteration, an operation with much unused energy before the battery swap is destroyed together with a neighboring operation or traveling leg. The resulting set of disconnected destinations with fixed starting and ending points is then given as input to an exact mixed-integer program (MIP) which reconstructs the subroute locally optimally (local search procedure). The new solution is accepted if it improves the current best solution, else it is accepted with probability $\frac{1}{2}$. Since in the pretests, our adapted DP-ILP was faster by a significant factor than the original MIP of Zeng et al. (2022) even on small instances, we used DP-ILP in the local search procedure to provide a fair advantage to the method.

Table 5.10 compares VLNS-LS (see Section 5.5.3), and NS on the data sets with $(n_d + 1) \in \{7, 8, 9, 10, 20, 50, 75, 100\}$ proposed by Zeng et al. (2022). Thereby NS was given the same run time as VLNS-LS and VLNS-LS used $p = 4$ on small ($7 \leq n_d \leq 20$) and $p = 3$ on large ($n_d \geq 50$) instances. VLNS-LS outperformed NS in every instance, with the best improvement of 15.4%. The runtime of VLNS-LS (and NS) equaled less than a second on small ($7 \leq n_d \leq 20$) instances and around 30 min on large ($n_d \geq 50$) instances, on average. Note that DP-ILP could solve all instances with $(n_d + 1) \leq 20$ to optimality within a runtime of 1200s, and additionally, 28 out of 90 instances with $(n_d + 1) = 50$ within a time limit of 3600s. This is significantly more than the exact solution method proposed by Zeng et al. (2022), which could close the optimality gap for instance sizes of at most $(n_d + 1) \leq 9$. For small instances ($n^d + 1 \leq 20$), VLNS-LS found close to optimal solutions (with a gap of $\leq 3\%$) in 98% of the cases, for the large instances ($n^d + 1 = 50$) with available optimum, it was even in 100% of the cases.

Table 5.11: Performance of VLNS-LS^{I*} and RTS of Poikonen and Golden (2020)

	Comparison of VLNS-LS^{I*} relative to RTS solutions				Comparison to the optimum			
Setting	Average improv.(%)	Best improv.(%)	Times improved	Times deteriorated	Average gap (%) VLNS-LS^{I*}	RTS	Times gap $\leq 3\%$ VLNS-LS^{I*}	RTS
Quadcopter								
Low energy	8.0	21.8	40/40	0/40	2.1	12.5	21/28	2/28
High energy	7.6	16.5	40/40	0/40	2.1	11.4	16/21	1/21
Octocopter								
Low energy	4.4	20.0	39/40	0/40	0.7	5.1	12/12	5/12
High energy	1.4	6.3	26/40	0/40	–	–	–	–
Total	5.3	21.8	145/160	0/160	1.8	10.7	49/61	8/61

Note. VLNS-LS^{I*} was initialized with 900s of ALNS*. A time-limit of 3600s was set to DP-ILP.

5.5.4.2 Delivery DRP-E of Poikonen and Golden (2020) with weight-dependent energy consumption

In the last-mile delivery DRP-E with a single drone of Poikonen and Golden (2020), each destination $v \in V_d$ requires a drone delivery of a package with weight $w(v)$. Therefore, the energy consumption of the drone is not linear in time, as in the basic DRP-E, but also depends on the currently carried weight. Furthermore, the drone can only wait for the truck while hovering, which consumes energy at a constant rate. Poikonen and Golden (2020) propose RTS to solve this delivery DRP-E, which optimally inserts RLs into a fixed sequence of destinations, namely the shortest Hamiltonian tour x_{TSP} over all destinations and the depot.

The left-hand side of Table 5.11 compares our VLNS-LS^{I*} with RTS on the data sets proposed by Poikonen and Golden (2020) with $|V_r \cup V_d| \in \{50, 75, 100\}$. The data sets feature a quadcopter and an octocopter. The latter has twice as many rotors, a much larger battery, and usually requires less battery swaps in operation. Each drone is considered with a battery of *low energy* density (540.000 J/kg) and *high energy* density (900.000 J/kg). Observe that RTS cannot return better solutions than VLNS-LS^{I*}, since it can be interpreted as a special case of VLNS-LS with $p = 1$. Remarkably, VLNS-LS^{I*} improved RTS in 91% of instances, in some cases by over 21%. The utility of VLNS-LS^{I*} compared to the simple approach RTS is higher in the case of less powerful drones with lower battery capacity, because the drone tour involves more frequent detours for battery swapping and thus the fixed delivery sequence x_{TSP} considered in RTS might be sub-optimal.

The right-hand side of Table 5.11 compares VLNS-LS^{I*} to optimal solutions generated with DP-ILP. Notice that in the original work of Poikonen and Golden (2020), no optimal solutions have been available. For instances with $|V_r \cup V_d| = 50$, DP-ILP found the optimal solution for 19 out of 20 quadcopter instances, all 10 instances for octocopter with low energy density, and further 32 optima for large-sized instances with $|V_r \cup V_d| \geq 75$. No optimal solutions have been found for the octocopter with a high energy density. Take note of the excellent performance achieved by VLNS-LS^{I*} on these instances: with an overall average gap of 1.8%, VLNS-LS^{I*} achieved close-to-optimal solutions (with a gap $\leq 3\%$) in 80% of the cases; for the powerful octocopter, it was even in 100% of the cases.

5.6 Conclusions

In this paper, we propose VLNS – a flexible improvement procedure for the drone routing problem with energy replenishment (DRP-E). In DRP-E, the drone, which is supported by a mobile replenishment station (MRS), has to visit a set of destinations. This problem setting is relevant for many drone applications and is a generalization of a number of classical routing problems. The proposed VLNS is a non-trivial very large-scale neighborhood search, which synergetically leverages two large polynomially solvable DRP-E subproblems (SP1 and SP2). Roughly speaking, the number of examined feasible solutions by VLNS is a multiple of those in SP1 and SP2, and, thus, grow exponentially in the size of the DRP-instance, whereas the computational complexity remains pseudo-polynomial. In VLNS, the size of the associated neighborhood, and thus, the computational time complexity, can be flexibly controlled via parameter p. If $p = 1$, the solutions of VLNS are equivalent to the state-of-the-art algorithm RTS (Poikonen & Golden, 2020), and with large enough values of p, VLNS turns into an exact approach.

It must be stressed, that VLNS *is not an algorithm* developed for a *specific* application, but it is a *general improvement tool* suitable for a wide family of DRP-E-like problems. In computational experiments (Section 5.5), we provided an insight on the generalizability of VLNS by applying it to several distinct DRP-E variants, in the context of a drone surveillance operation, and a last-mile delivery problem with payload restrictions and non-trivial energy consumption of the drone. We demonstrated that even a simple implementation of VLNS as a local search procedure, possibly equipped with a novel, problem-specific initialization procedure (VLNS-LS^{I*}), outperformed the state-of-the-art heuristics for DRP-E variants by a significant margin.

As a valuable by-product of our research, we introduced an exact approach, DP-ILP, which was able to find new, optimal solutions for large-sized problem instances of the studied DRP-E variants from the literature, and solved specific DRP-E settings with up to 149 locations to optimality.

Future research should refine the concept of VLNS by reasonably embedding it in further meta- and hyper-heuristic frameworks. Although the computational time of VLNS is low-degree polynomial in the number of nodes, it takes quite a significant time to search the neighborhood for very large problem instances. Promising opportunities for a speed-up may include a heuristic search of the developed neighborhood or the customization of the neighborhood by turning parameter p to be node-specific, see the related discussions in Balas and Simonetti (2001). Future research should also address how to adapt the proposed VLNS methodology to a number of non-trivial extensions of DRP-E, such as multiple drones or MRSs, or stochastic flight times.

5.7 Acknowledgements

This project was supported by the Deutsche Forschungsgemeinschaft (DFG - German Research Foundation) as part of the SuPerPlan project GZ: OT 500/4-1. Research of Daniele Vigo was partially supported by U.S. Air Force Office of Scientific Research (Grant FA8655-21-1-7046).

5.8 Supplemental material

5.8.1 Supplement: Technical details and proofs

5.8.1.1 Construction of the ops graph

In order to achieve polynomial complexity of the ops graph, we represent only operations in the graph which are $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid for the given neighborhood $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}(x,p)$. Therefore, we introduced the concept of $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid states in Section 5.3.2.1 and limited the ops graph to $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid states only. The goal of this section is to prove Lemma 23 from Section 5.3.2.1 which provides instruction how to construct the ops graph in a forward induction.

The example from Figure 5.2 provides an intuition for identifying $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid states. Lemma 24 formalizes this intuition by introducing a property which limits the possible sets of visited destinations inside of an operation which is relevant for $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}(x,p)$.

Lemma 24. *For the BS-R neighborhood $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}(x,p)$, consider a $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid operation $o = wsw'$ with $\{s\} = S$ and let define $m := \min\{j : v_j \in S\}$ (the lowest index of destinations in S) and $M := \max\{j : v_j \in S\}$ (the largest index of destinations in S). Then*

$$\forall i \in [m+p, M-p] \text{ we have } v_i \in S \tag{5.27}$$

Proof. Consider a $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid operation $o \in wsw'$ with $\{s\} = S$ and m and M as defined above. We prove Condition (5.27) by the law of contraposition. Let $v_l \in (V_d \setminus S)$. Then, for any drone tour $\pi_d \in \mathcal{N}_{\mathcal{BS}-\mathcal{R}}(x,p)$ with $o \in \pi_d$, two cases are possible. Either v_l precedes the operation o in π_d, then $l < m+p$ by the precedence constraints (5.1) of the Balas-Simonetti neighborhood, or v_l follows the operation o in π_d, then $l > M-p$ by the precedence constraints (5.1) of the Balas-Simonetti neighborhood. We conclude that $v_i \in S \ \forall i \in [m+p, M-p]$. □

Proposition 16 uses this lemma to give a full characterization $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid states in the ops graph. Note that all initial states are $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid by definition.

Proposition 16. *For the BS-R neighborhood $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}(x,p)$, consider state (w,S,v) with $w \in V_r$, $S \subseteq V_d, |S| \geq 1$, and $v \in S \cup V_r$. Define indices m and M as in Lemma 24. Then (w,S,v) is $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid iff one of the following is true:*

- *if $v \in V_r$ then*

$$- \qquad \forall l \in [m+p, M-p] : \ v_l \in S \tag{5.28}$$

- *if $v = v_i \in S$ then*

$$- \qquad \forall l \in [m+p, M-p] : \ v_l \in S \tag{5.29}$$

$$- \qquad \forall v_l \in S : \ i > l-p \tag{5.30}$$

Proof. The necessity of Conditions (5.28) and (5.29) follows directly from Lemma 24 and the necessity of Conditions (5.30) as a consequence of the Balas-Simonetti precedence constraints (5.1). Let's consider the sufficiency of Conditions (5.29) and (5.30) for the case of $v \in S$, since the proof can be directly extended to the remaining case. Define set T_1 as the subset of $V_d \setminus S$ such that $\forall v_j \in T_1 : j < m+p$ and T_2 as the subset of $V_d \setminus S$ such that $\forall v_j \in T_2 : j > M-p$. Because of Condition (5.29), we have $T_1 \cup S \cup T_2 = V_d$. Let's define $\langle T \rangle$ as a sequence of destinations of some set $T \subseteq V_d$ ordered in increasing order of their indices. Consider the sequence of destinations $\rho = (\langle T_1 \rangle, \langle S \setminus \{v\} \rangle, v, \langle T_2 \rangle)$. Observe that:

- $\forall v_j \in T_1, v_l \in S$: we have $j < l + p$ and Balas-Simonetti precedence constraints (5.1) hold.
- $\forall v_l \in S \setminus \{v = v_i\}$, v_i: we have by condition (5.30) $i > l - p$ and Balas-Simonetti precedence constraints (5.1) hold.
- $\forall v_j \in S, v_l \in T_2$: we have $l > j - p$ and Balas-Simonetti precedence constraints (5.1).

Since constraints 5.1 applies straightforwardly to the remaining cases by construction, we conclude that $\rho \in \mathcal{N}_{\mathcal{BS}}(x, p)$. We can construct a drone tour $\pi_d \in \mathcal{N}_{\mathcal{BS}-\mathcal{R}}(x, p)$ with operation $o = (w, \langle S \setminus \{v\} \rangle, v, w') \in \pi_d$ for some arbitrary $w' \in V_r$ and $\pi_d(V_d) = \rho$ accordingly. Thus state (w, S, v) is $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid. □

Now we are able to proof Lemma 23.

Proof of Lemma 23. The second statement follows straightforwardly. For the first statement, define $M' := \max\{M, i\}$ and $S' = S \cup \{v'\}$. The necessity of Condition (5.4) for state $(w, S', v') \in \mathcal{O}$ follows from the following analysis of the values of i in $v' = v_i$:

- $i \leq M - p$: impossible given Condition (5.30) of Proposition 16.
- $i = M$: impossible since $v_i \in V_d \setminus S$.
- $i \in [M + p + 1, n_d]$: impossible since in this case there is $v_l \in V_d \setminus S'$ with $l \in [M + 1, M' - p]$, in contradiction with Condition (5.29) of Proposition 16.
- $i \in [\max\{M + 1, m + 2p\}, M + p]$: In this case, $M' = i$ and $[m + p, M' - p] \neq \emptyset$. Therefore, Proposition 16 applies to (w, S', v') only if $(v_j \in S \ \forall j \in [m + p, i - p])$.

We proof the sufficiency of Condition (5.4), by distinguishing the following cases:

- $i \in [M - p + 1, M - 1]$: Since $M' = M$ and $(w, S, v) \in \mathcal{O}$, Proposition 16 applies to (w, S', v').
- $i \in [M + 1, m + 2p - 1]$: Since $M' = i$ and $[m + p, M' - p] = \emptyset$, Proposition 16 applies to (w, S', v').
- $i \in [\max\{M + 1, m + 2p\}, M + p]$: Since $M' = i$, Proposition 16 applies to (w, S', v'). □

5.8.1.2 Complexity of the ops graph

The goal of this section is to prove the upper bound for the polynomial time complexity of the ops graph $\mathcal{O}$ from Proposition 12 in Section 5.3.2.2. We outlined that the complexity is linear to the amount of transition arcs in $\mathcal{O}$, and that the number of outgoing arcs of non-terminal states is bounded by $n_r + 2p - 1$, while terminal states have no outgoing arcs. What remains to complete the proof of Proposition 12 is to give approximate the total amount of non-terminal states in $\mathcal{O}$.

Lemma 25. *Let U be the number of subsets $S \subseteq V_d$ with $|S| = k > 1$, $n_d \geq 4p - 2$ (large enough number of destinations), $m := \min\{j : v_j \in S\}$ and $M := \max\{j : v_j \in S\}$ such that Condition (5.29) from Proposition 16 holds, i.e:*

$$\forall l \in [m + p, M - p] : v_l \in S \tag{5.31}$$

Then:

- *if $k \in [2p, n_d - 2p + 2]$, then $U = (n_d - k - p + 2)4^{p-1}$* (5.32)
- *if $k < 2p$, then $U \leq (n_d - k - p + 2)4^{p-1}$* (5.33)
- *if $k > n_d - 2p + 2$, then $U \leq p \cdot 4^{p-1}$* (5.34)

Proof. Let construct a subset S as described above. For any choice of the minimal index m and maximal index M, let denote by $T_{m,M}$ the subset of all destinations with indices between m and M: $T_{m,M} = \{v_m, v_{m+1}, ...v_{M-1}, v_M\}$, and let denote $l_{m,M}$ its length: $l_{m,M} = M - m + 1$. In the following, we will count possible ways to select $T_{m,M}$ from V_d and S from $T_{m,M}$ for each possible $k = |S|$. Since S needs to accommodate k destinations, and $S \subseteq T_{m,M}$, we must have: $l_{m,M} \geq k$. On the other hand, $T_{m,M} \subseteq V_d$, thus: $l_{m,M} \leq n_d$.

i) The desired subset S is completely defined by the choice of m, M and $T_{m,M} \setminus S$. According to Condition (5.31), $\forall v_j \in T_{m,M} \setminus S$, we must have:

$$j \in \ I_{m,M} :=]m, \min\{m + p, M\}[\ \cup\ ]\max\{M - p, m\}, M[,$$

with $2p - 2$ being the maximal length of $I_{m,M}$.

ii) All together, m and M must be chosen such that:

$$k \leq l_{m,M} \leq \min\{|S| + |I_{m,M}|, n_d\} \leq \min\{k + 2p - 2, n_d\}$$

iii) For fixed length $l_{m,M}$, there are exactly $n_d - l_{m,M} + 1$ subsets $T_{m,M}$ of that length in V_d.

Case 1: $2p \leq k \leq n_d - 2p + 2$.

Since $l_{m,M} \geq k \geq 2p$, then $M - m + 1 \geq 2p$ as well as the interval $I_{m,M}$ is exactly $[m + 1, m + p - 1] \cup [M - p + 1, M - 1]$ and has $2p - 2$ elements. So, for given m, M with associated length $l_{m,M} = k + i$, and $i \in [0, \min\{2p - 2, n_d - k\}] = [0, 2p - 2]$, we have $\binom{2p-2}{i}$ possibilities to select the indices of $T_{m,M} \setminus S$ out of $I_{m,M}$. Putting i), ii) and iii) together, the amount U of choices for S with $|S| = k$, is given by:

$$\begin{aligned}
U = &\sum_{i=0}^{2p-2} (n_d - (k + i) + 1)\binom{2p-2}{i} \\
=&(n_d - k + 1) \sum_{i=0}^{2p-2} \binom{2p-2}{i} - \sum_{i=0}^{2p-2} i\binom{2p-2}{i} \\
=&(n_d - k + 1)2^{2p-2} - (2p - 2)2^{2p-2-1} \\
=&(n_d - k - p + 2)4^{p-1}
\end{aligned} \tag{5.35}$$

Case 2: $k < 2p$.

In this case, it is possible to select m and M such that $l_{m,M} < 2p$, which implies that the left sub-interval and right sub-interval of $I_{m,M}$ overlap. The size of $I_{m,M}$ reduces then from $2p-2$ to $l_{m,M}-2$.

Thus, we need to split up the number of choices U in two sums, according to the distance of the selected m and M. Note that $l_{m,M}$ can still take all values $k+i$ for $i \in [0, 2p-2]$.

$$U = \sum_{i=0}^{2p-1-k} (n_d - (k+i) + 1) \binom{k+i-2}{i} + \sum_{i=2p-k}^{2p-2} (n_d - (k+i) + 1) \binom{2p-2}{i}$$

The amount U is obviously bounded by the expression on the right-hand side from Equation (5.35).

Case 3: $k = n_d - 2p + 2 + j$ for $j \in [1, 2p-2]$.

Observe that $k + 2p - 2 - j = n_d$. In this case, m and M must be chosen such that $l_{m,M} \leq n_d = k + 2p - 2 - j$. The remaining construction of U is identical to case 1.

$$\begin{aligned}
U &= \sum_{i=0}^{2p-2-j} (n_d - (k+i) + 1) \binom{2p-2}{i} \\
&\leq \sum_{i=0}^{2p-2-j} (n_d - (k+i) + 1 + j) \binom{2p-2}{i} \\
&\leq \sum_{i=0}^{2p-2} (n_d - (k-j+i) + 1) \binom{2p-2}{i} \\
&= p \cdot 4^{p-1} \text{ by Formula (5.35)} \qquad \square
\end{aligned}$$

Using the technical upper bound from Lemma (25), we can now provide an approximation for the number of non-terminal states in $\mathcal{O}$:

Lemma 26. *Consider the BS-R neighborhood $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}(x,p)$. The associated ops graph $\mathcal{O}$ has $O(n_d^2 n_r p \cdot 4^p)$ non-terminal states.*

Proof. By construction, there are $n_d \cdot n_r$ non-terminal states at stage 1, each having transitions to at most $2p-1$ different non-terminal states at stage 2 by Lemma 23. Thus there are $O(n_d \cdot n_r(2p-1))$ non-terminal states at stage 2. For any state $(w, S, v), v \in S$ at stage $k = |S| \geq 3$:

- w can take n_r possible values
- set $S \setminus \{v\}$, which has $k-1$ items, can take the following number of values (insert $|S \setminus \{v\}| = k-1$ in equations (5.32)-(5.34) of Lemma 25):
  - not more than $(n_d - k - p + 3)4^{p-1}$ values if $k \leq n_d - 2p + 3$
  - not more than $p \cdot 4^{p-1}$ values otherwise
- v, provided fixed $S \setminus \{v\}$, can take not more than $2p-1$ values according to Lemma 23

Observe that there are $(2p-3)$ stages with $k > n_d - 2p + 3$. In total, for all stages $k \geq 3$, the number of non-terminal states is not larger than:

$$\begin{aligned}
&(2p-3)n_r(2p-1)p \cdot 4^{p-1} + \sum_{k=3}^{n_d-2p+3} n_r(2p-1)(n_d-k-p+3)4^{p-1} \\
&= n_r(2p-1)\cdot 4^{p-1} \cdot [\ (2p-3)p + \sum_{k=0}^{n_d-2p}(n_d-k-p)] \\
&= n_r(2p-1)\cdot 4^{p-1} \cdot [\ (2p-3)p + (n_d-2p+1)\frac{1}{2}n_d] \\
&= O\left(p \cdot n_r \cdot 4^p \cdot [(n_d)^2 + p^2 - p \cdot n_d]\right) \\
&= O\left(p \cdot n_d^2 n_r \cdot 4^p\right) \qquad \square
\end{aligned}$$

Multiplying the amount of non-terminal states in $\mathcal{O}$ with the upper bound of outgoing transition arcs for each of those states, provides us with the complexity of the ops graph as stated in Proposition 12.

5.8.1.3 Construction and complexity of the meta graph

The meta graph implicitly enumerates all drone tours in a given BS-R neighborhood $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}(x,p)$. In order to achieve polynomial complexity of the meta graph, we reduce it to so-called $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid metastates and transitions which are relevant for $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}(x,p)$, while preserving the general architecture of the graph described in Section 5.3.3.1. In this section, we prove Proposition 13 and Proposition 14, which characterize the special structure of valid metastates and transitions, respectively, given the new encoding scheme introduced in Section 5.3.3.2.

Proof of Proposition 13. The necessity of the left hand side of Condition (5.11) follows straightforwardly from the definition of sets S_k^- and S_k^+. We refer to Balas, 1999 for the proof of necessity for the remaining conditions.

Now suppose that for two sets S_k^- and S_k^+ as defined by Equations (5.10), Conditions (5.11)-(5.13) are satisfied. We can define a corresponding permutation σ of $[n_d]$, such that the BS precedence constraints for $x' = (x_{\sigma^{-1}(1)}, x_{\sigma^{-1}(2)}, ..., x_{\sigma^{-1}(n_d)})$ hold.

Set $\sigma(i) = i$ for $i \in [n_d] \setminus (S_k^- \cup S_k^+)$ and consider any bijective mapping from S_k^- to S_k^+. Let check precedences for all possible configurations of a pair of indices $i, j \in [n_d]$ with $i + p \leq j$:

- $i, j \notin S_k^- \cup S_k^+$: by definition $\sigma(i) = i < j = \sigma(j)$
- $i \notin S_k^- \cup S_k^+$ and $j \in S_k^-$: then by definition and Condition (5.13), $\sigma(j) \geq \min\{h : h \in S_k^+\} > \max\{l : l \in S_k^-\} - p \geq j - p \geq i = \sigma(i)$
- $i \notin S_k^- \cup S_k^+$ and $j \in S_k^+$: then $i \leq k$ and $\sigma(j) > k \geq i = \sigma(i)$ by definition
- $i \in S_k^+$: then we must have $j \notin S_k^- \cup S_k^+$ because of Conditions (5.12) and (5.13). So, $\sigma(i) \leq \max\{l : l \in S_k^-\} < \min\{h : h \in S_k^+\} + p \leq i + p \leq j = \sigma(j)$
- $i \in S_k^-$: then $j > k$ and $j \notin S_k^- \cup S_k^+$ because of Conditions (5.12) and (5.13). So, $\sigma(i) \leq k < j = \sigma(j)$

In Section 5.2 we assumed that the maximal flight time e_{max} suffices to visit each destination in a return flight from its closest-by RL. Since state (S_k^-, S_k^+, w) is relevant for this π_d, it is $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid. $\square$

Proof of Proposition 14. The case of $h = 0$ is straightforward.

Consider $h \geq 1$. Let $S = [k] \setminus S_k^+ \cup S_k^-$ and $T = [k+h] \setminus S_{k+h}^+ \cup S_{k+h}^-$ be the index sets of the k first visited and $k+h$ first visited destinations respectively. By the definition of transition arcs in Section 5.3.3.1, the following should be true: $S \subset T$.

Let show that $S \subset T$ is equivalent to Conditions (5.14)-(5.16).

Given $j \in S$, let show that $j \in T$ by analysing the following cases:

- $j \leq k$: Then $j \notin S_k^+$ (by definition of S), $j \notin S_{k+h}^+$ because of Condition (5.16). Thus, $j \in T$.
- $k < j \leq k+h$: Since $j \in S$, we have $j \in S_k^-$. Then $j \in T$ because of Condition (5.15).
- $j > k+h$: Since $j \in S$, we have $j \in S_k^-$. Then $j \in T$ because of Condition (5.14).

The necessity of conditions Conditions (5.14)-(5.16) if $S \subset T$ follows straightforwardly.

Now, we need a formula to deduce the operation set $O = wHw'$ corresponding to the transition arc between metastate $(S_k^-, S_k^+, w) \in \mathcal{V}$ at stage k and metastate $(S_{k+h}^-, S_{k+h}^+, w') \in \mathcal{V}$ at stage $k+h, h \geq 1$, with the new encoding. Let $I_k = [k] \setminus S_k^+ \cup S_k^-$ and $I_{k+h} = [k+h] \setminus S_{k+h}^+ \cup S_{k+h}^-$ be the index sets of the k first visited and $k+h$ first visited destinations, respectively.

We can compute the set of destinations H as $H = \{v_l \ : \ l \in I\}$, with

$$I = I_{k+h} \setminus I_k = (\{k+1, ..., k+h\} \cup S_k^+ \cup S_{k+h}^-) \setminus (S_{k+h}^+ \cup S_k^-) \tag{5.36}$$

Note that we got the second equality of (5.36) by applying the well-known set properties

(1) $A \setminus (B \cup C) = A \setminus B \setminus C = A \setminus C \setminus B$

(2) $(A \cup B) \setminus C = (A \setminus C) \cup (B \setminus C)$

(3) $A \setminus (B \setminus C) = (A \setminus B) \cup (A \cap C)$

to I_{k+h} and I_k from the left-hand side. □

Given the characteristics of $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}(x,p)$-valid metastates from Proposition 13 , Proposition 17 counts the amount of these states in the metagraph. The proof also sketches the construction for these states at fixed stage k which we use to fill in the lookup table (cf. Table 5.2).

Proposition 17. *For a given sequence of destinations x and a given parameter p, meta graph $\mathcal{G}$ for neighbourhood $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}(x,p)$ has $O(n_d n_r 2^p)$ states.*

Proof. Consider state (S_k^-, S_k^+, w) with $i := |S_k^-| = |S_k^+| \geq \frac{p}{2}$ at some *typical* stage k (i.e., neither the first $(p-1)$ nor last $(p-1)$ stages). If $i = 0$, there is exactly one possible pair of sets $S_k^- = S_k^+ = \emptyset$. For $1 \leq i \leq \lfloor \frac{p}{2} \rfloor$, the rules of Proposition 13 allow us to construct all $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid meta states in the following manner:

Let denote $m := \max\{l \ : \ l \in S_k^-\}$. S_k^- and S_k^+ can be chosen as any i-sized subsets of $\{k+1, k+2, ..., m\}$ that include m, and $\{m-p+1, ..., k-1, k\}$, respectively. To ensure that S_k^-, S_k^+ fit i elements, the range of possible values for m is given by: $k+i \leq m \leq k+p-i$.

Putting all together, the amount of $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid metastates at stage k equals exactly:

$$n_r \left(1 + \sum_{i=1}^{\lfloor \frac{p}{2} \rfloor} \sum_{m=i}^{p-i} \binom{m-1}{i-1} \binom{p-m}{i} \right) = n_r 2^{p-1}, \tag{5.37}$$

where Equality (5.37) can be proven in a similar fashion as Lemma 4.6 in (Balas, 1999).

For *non-typical stages*, we can adapt a similar construction procedure of states, and further limit S_k^- and S_k^+ to subsets of $[n_d]$. The statement of Proposition 17 follows straightforwardly. $\square$

5.8.2 Supplement: Pseudocodes for main functions of VLNS

The implementation of the neighborhood search VLNS is based on three main functions only: The function *extend_operations*(w) is called for each RL $w \in V_r$ and, stage by stage, extends states of the operations graph described in Section 3.2.1 towards non-terminal $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid states, by prolonging a drone flight by one not yet visited destination. The function *close_operations* gets as input the state collections $\mathcal{VO}^w$ for $w \in V_r$ created by *extend_operations*(w) and closes feasible $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid operations. The output of the latter is the collection of all feasible, $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid operation sets $\mathcal{OP}^*{}_{BS-R}$, where each operation set is associated with its lowest-possible makespan and the visiting sequence to attain this makespan is re-traceable with the information stored in $\mathcal{VO}^w$. The function *close_operations* calls two auxiliary routines, *closing_distance* and *closing_location*, which take as input some $v \in V_d$. The output of *closing_distance*(v) is a vector with n_r entries, which orders the distances of v toward each RL $w \in V_r$, $c_d(v,w)$, in an increasing fashion. For the same $v \in V_d$, *closing_location*(v) returns a list of RL's, ordered in the way of increasing distance toward v. The pseudocodes for *extend_operations* and *closing_distance* are presented by Algorithms 4 and 5.

Algorithm 4 *extend_operations*

```
1: Input $w \in V_r$, $n_d := |V_d|$ , $x = (v_1,...,v_{n_d})$ visiting sequence of $V_d$, $p \in \mathbb{N}$ parameter
2: $\mathcal{VO}^w \leftarrow \emptyset$ ;
3: # Extensions from initial state $(w,\emptyset,w)$ of operations graph
4: for $i \in [1,n_d]$ do
5:     if $c_d(w,v_i) + \min\{c_d(v_i,u) | u \in V_r\} \le e_{max}]$ then
6:         add state $s = (\{i\},i)$ to $\mathcal{VO}^w$ with $s.value = c^d(w,v_i), s.last = w, s.m = i, s.M = i$;
7:     end if
8: end for
9: # Extend other states in the order of increasing stage $k$
10: for $k \in [1,n_d - 1]$ do
11:     for all states $s = (S,j) \in \mathcal{VO}^w$ with $|S| = k$ do
12:         $treshold = \max\{s.M, s.m + 2p\}$;
13:         for $i \in ([1,n_d] \setminus S) \cap [s.M - p + 1, \min\{n, \text{treshold} - 1\}]$ do
14:             $z \leftarrow s.value + c_d(v_j,v_i)$;
15:             if state $s' = (S \cup i, i) \notin \mathcal{VO}^w$ and $z + \min\{c_d(v_i,w) | w \in V_r\} \le e_{max}$ then
16:                 add $s'$ to $\mathcal{VO}^w$ with $s'.value = z, s'.last = j, s'.m = \min\{i, s.m\}, s'.M = \max\{i, s.M\}$;
17:             else if state $s' = (S \cup i, i) \in \mathcal{VO}^w$ and $z < s'.value$ then
18:                 update $s'$ in $\mathcal{VO}^w$ with $s'.value = z, s'.last = j, s'.m = \min\{i, s.m\}, s'.M = \max\{i, s.M\}$
19:             end if
20:         end for
21:         $i \leftarrow$ treshold ;
22:         while $i - p \in S$ and $i \le n_d$ do
23:             if $i \in [1,n_d] \setminus S$ then
24:                 $z \leftarrow s.value + c_d(v_j,v_i)$;
25:                 if state $s' = (S \cup i, i) \notin \mathcal{VO}^w$ and $z + \min\{c_d(x_j,w) | w \in V_r\} \le e_{max}$ then
26:                     add $s'$ to $\mathcal{VO}^w$ with $s'.value = z, s'.last = j, s'.m = \min\{i, s.m\}, s'.M = \max\{i, s.M\}$
27:                 else if state $s' = (S \cup i, i) \in \mathcal{VO}^w$ and $z < s'.value$ then
28:                     update $s'$ in $\mathcal{VO}^w$ with $s'.value = z, s'.last = j, s'.m = \min\{i, s.m\}, s'.M = \max\{i, s.M\}$
29:                 end if
30:             end if
31:             $i \leftarrow i + 1$
32:         end while
33:     end for
34: end for
35: Return $\mathcal{VO}^w$
```

Algorithm 5 *closing_operations*

```
1: Input $\mathcal{VO}^w$ for $w \in V_r$, $x = (v_1, ..., v_{n_d})$ visiting sequence of $V_d$, $p \in \mathbb{N}$ parameter;
2: $\mathcal{OP}^*_{BS-R} \leftarrow \emptyset$;
3: for $w \in V_r$ do
4:     for all states $s = (S, i) \in \mathcal{VO}^w$ do
5:         $j \leftarrow 1$;
6:         $z \leftarrow s.value + closing\_distance(v_i)_1$;
7:         while $z \leq e_{max}$ do
8:             $u \leftarrow closing\_location(v_i)_j$;
9:             if $c_r(w, u) \leq e_{max}$ then
10:                if state $s' = (w, S, u) \notin \mathcal{OP}^*_{BS-R}$ then
11:                    add $s'$ to $\mathcal{OP}^*_{BS-R}$ with $s'.value = \max\{z, c_r(w, u)\}, s'.last = i$;
12:                else if $\max\{z, c_r(w, u)\} < s'.value$ then
13:                    update $s'$ in $\mathcal{OP}^*_{BS-R}$ with $s'.value = \max\{z, c_r(w, u)\}, s'.last = i$
14:                end if
15:            end if
16:            if $j < n_r$ then
17:                $j \leftarrow j + 1$;
18:                $z = s.value + closing\_distance(v_i)_j$;
19:            else
20:                break
21:            end if
22:        end while
23:    end for
24: end for
25: Return $\mathcal{OP}^*_{BS-R}$
```

The function *metagraph* of Algorithm 6 takes as input the set of $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid operations $\mathcal{OP}^*_{BS-R}$ and the lookup table (cf. Table 2) of $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid metastates with respect to the parameter $p \in \mathbb{N}$. It creates the metagraph in a forward movement, by extending states of type $s = (k, ind, w)$, where k is the number of visited destinations and the stage of state s , ind is the index of the $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid meta state s in the lookup table and $w \in V_r$ is the current position of the drone at an RL. Each state s is attributed with the following information $s.value$ is the minimum time to reach this state, $s.leg$ contains the RL $w' \in V^r$, such that the current position w was reached directly through the recharging leg $r = (w', w)$, and $s.last$ saves the predecessor state before going that recharging leg.

5.8.3 Supplement: Special case of DRP-E with a single depot

Consider instances with a single depot $w_0 = w_t$. The first and the last visited destinations of a good drone tour π_d will probably be geographically close to w_0 and, thus, close to one another. This implies that it may be reasonable to visit destinations at terminal and initial positions of sequence $x = \pi_d(V_d)$ within the same operation. But BS-R neighbourhood does not allow this if $n_d \gg (2p + 1)$ and if the maximal flight time e_{max} is limited. For example, as follows from Lemma 1, v_1 and v_{n_d} may belong to the same operation only if destinations $[v_{1+p}, v_{n_d - p}]$ belong to it as well.

The adaptation of neighbourhood $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}(x, p)$ to the case of the single depot is not straightforward. We proceed as follows: After having examined $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}(x, p)$ as described above, we create a small number, $p(p - 1)$, of additional permutations of x by shifting one node in x either from one of the $(p - 1)$ positions in the beginning to one of the $(p - 1)$ positions in the end of sequence x, or *vice versa*. For each such permutation, we compute an optimal positioning of the RLs in $O\left(n_d^2 n_r^2\right)$.

The computational complexity of VLNS remains unchanged.

Algorithm 6 function *metagraph*

```
1: Input set of $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid operations $\mathcal{OP}^*_{BS-R}$, lookup
2: # Initialization
3: $\mathcal{M} \leftarrow \{s_0\}$ where $s_0$ is the initial state $s_0 = (0, 0, w_0)$ with $s_0.value = 0,\ s_0.leg = None,\ s_0.last = None$ ;
4: # Start main loops
5: for $i \in [0, n_d]$ do
6:     # Extend by replenishment legs
7:     for all states $s = (k, ind, w) \in \mathcal{M}$ with $k = i$ do
8:         if $s.leg = None$ or $s.leg = w$ then
9:             for $u \in V_r$ do
10:                $z = s.value + c_r(w, u)$;
11:                if state $s' = (k, ind, u) \notin \mathcal{M}$ then
12:                    add state $s'$ to $\mathcal{M}$ with $s'.value = z, s'.leg = w, s' - last = s.last$
13:                else if $s'.value \geq z$ then
14:                    update state $s'$ in $\mathcal{M}$ with $s'.value = z, s'.leg = w, s' - last = s.last$
15:                end if
16:            end for
17:        end if
18:    end for
19:    # Extend by operations
20:    for all states $s = (k, ind, w) \in \mathcal{M}$ with $k = i$ do
21:        for all $j \in [1, n_d - k], u \in V_r$ do
22:            In row $ind$ (which encodes the current $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid state $s$ at stage $k$) and in column $j+1$ of $lookup$,
               we find the $list$ of state indices at stage $k+j$ which can be reached by an $\mathcal{N}_{\mathcal{BS}-\mathcal{R}}$-valid transition
               from the current state $s$
23:            for each state index $ind' \in list$ do
24:                Let $H$ be the set of destinations corresponding to this transition, given by the formula from
                   Proposition 14;
25:                if $op = (w, H, u) \in \mathcal{OP}^*_{BS-R}$ then
26:                    $z = s.value + op.value$;
27:                    if state $s' = (k + j, ind', u) \notin \mathcal{M}$ then
28:                        add state $s'$ to $\mathcal{M}$ with $s'.value = z, s'.leg = None, s' - last = s$
29:                    else if $s'.value > z$ then
30:                        update state $s'$ to $\mathcal{M}$ with $s'.value = z, s'.leg = None, s' - last = s$
31:                    end if
32:                end if
33:            end for
34:        end for
35:    end for
36: end for
37: # Extend by last replenishment legs connecting to target depot
38: for all states $s = (n, ind, w) \in \mathcal{M}$ do
39:     $z = s.value + c_r(w, w_0)$;
40:     if state $s' = (n, 0, w_0) \notin \mathcal{M}$ then
41:         add state $s'$ to $\mathcal{M}$ with $s'.value = z, s'.leg = w, s' - last = s.last$;
42:     else if $s'.value \geq z$ then
43:         update state $s'$ in $\mathcal{M}$ with $s'.value = z, s'.leg = w, s' - last = s.last$
44:     end if
45: end for
46: Return $\mathcal{M}$
```

Chapter 6

Conclusions and future directions

In this thesis, we considered selected topics on online optimization and its associated subproblems. The main goal was to contribute to the landscape of *contemporary, real-world dynamic optimization problems* with new recommendations for good online policies based on solid *analytical studies and evaluation mechanisms*.

Some background about online algorithms and the difficulty of evaluating them has been provided in Chapter 1, including a glimpse on the importance of robotic applications in dynamic environments. In Chapter 2, we formally introduced the *Online Order Batching, Sequencing, and Picker Routing Problem (OOBSRP)* with manual- and robotic carts, and demonstrated through *competitive and probabilistic analysis*, as well as a *computational validation*, the close-to-optimal performance of the myopic *reoptimization policy* (*Reopt*). Chapter 3 continued the study of OOBSRP from a managerial perspective, by providing actionable advice on how to reduce both the operating costs and the delivery times through simple algorithmic enhancements, by conducting a *pattern analysis of Complete-Information Optimal policy Solutions (CIOSs)*. Chapter 4 turned to disaster relief distribution with dynamically revealing edge blockages, specifically the *Traveling Salesman Problem with a Truck and a Drone under Incomplete Information (TSP-DI)*, and presented a *parametric competitive analysis* for *Reopt*, a *conservative*- delivery policy, and a policy with prior *drone surveillance*. Given the technicality of the study, the battery limitations of the drone have been left aside and considered separately in Chapter 5, which develops the *very large-scale neighborhood search VLNS*, a general algorithmic tool to tackle drone routing with energy replenishment.

Our research highlights the volatility of what characterizes a good online algorithm with the structure of the given problem: While *myopic policies* are *near-optimal* in order picking with dynamically arriving orders, myopic decision-making could be *the downfall* of truck-and-drone deliveries with network uncertainty. The same was observed for anticipatory mechanisms: In OOBSRP, the intuitive meaningfulness of *strategic waiting*, i.e. the deliberate delay of picking for a better fit of future orders, was *deceptive* (see Chapter 3), while in TSP-DI, the seemingly conservative postponement of relief delivery for additional knowledge acquisition demonstrated *extremely useful* (see Chapter 4). Managers and planners should keep this in mind and analyze carefully the nature of their problem at hand, before trusting their instincts on allegedly forward-thinking tweaks, and also before investing in high-stake *advanced anticipatory mechanisms* - in some cases, the latter will lead to substantial increases in profitability and cost efficiency, in other cases, the improvements to simple policies might be only marginal. Last but not least, managers should also give thorough consideration to implementing *automation* and integrating *robots* in their dynamic environment - their mechanical abilities may potentially allow for an important expansion of algorithmic limits (see Chapter 4).

A similar recommendation shall be addressed to academics entering the field of online optimization. Before pursuing sophisticated learning-based and data-driven approaches, they should first carefully assess the *accessibility and reliability* of the available data and the stochastic assumptions in the targeted dynamic environment. Additionally, they need to evaluate the importance of *robustness* and *worst-case* performance compared to *average quality and speed* in the associated applications. Following this, conducting competitive or probabilistic analyses, as well as developing and learning from *optimal* online algorithms on targeted subproblems, could be extremely beneficial. This process will enhance their understanding of the problem at hand and provide a solid foundation for further investigations. A wide array of practical dynamic optimization problems may benefit from this class of analytical studies, representing a largely unexplored field that offers numerous rich research opportunities.

This dissertation, which has forged a path in this direction for online order picking and delivery with dynamic edge blockages, has answered numerous research questions, but it has also opened the door to many more. Recall for instance Conjecture 1 of Chapter 2 that left the intriguing question, of whether *Reopt* for OOBSRP under the Poisson process model for order arrivals pertains its asymptotic optimality for *all* rates of incoming orders, or, whether there exist fixed *thresholds* for rates bounding this property. On a more practical side, the recommendation of *simple* policies may have to be revised, when extending the problem definition of OOBSRP to packing and delivery, or, to the collaboration of multiple pickers. The latter for instance, may add an additional source of dynamism (see Section 1.1.1) to the problem through dynamic congestion of the aisles, which could alter the algorithmic properties. Similarly, additional problem features may be incorporated in the study of TSP-DI, like, for instance, the extension of uncertainty to the demand of relief supply, the collaboration of multiple vehicle teams, and - most importantly, the energy limitations of the drone. The developed VLNS in Chapter 5 is inherently designed to seamlessly integrate into algorithmic schemes like reoptimization or the multiple scenario approach (cf. Section 1.1.2), paving the way for exciting algorithmic research in the area of dynamic robot routing problems.

Bibliography

Abouee-Mehrizi, H., & O.Baron. (2016). State-dependent M/G/1 queueing systems. *Queuing Systems*, *82*, 21–148.

Accorsi, L., & Vigo, D. (2020). A hybrid metaheuristic for single truck and trailer routing problems. *Transportation Science*, *54*(5), 1351–1371.

Aerts, B., Cornelissens, T., & Sörensen, K. (2021). The joint order batching and picker routing problem: Modelled and solved as a clustered vehicle routing problem. *Computers & Operations Research*, *129*, 105168.

Agatz, N., Bouman, P., & Schmidt, M. (2018). Optimization approaches for the traveling salesman problem with drone. *Transportation Science*, *52*, 965–981.

Ahmadi, M., Seifi, A., & Tootooni, B. (2015). A humanitarian logistics model for disaster relief operation considering network failure and standard relief time: A case study on San Francisco district. *Transportation Research Part E*, *75*, 145–163.

Ahuja, R. K., Ergun, O., Orlin, J. B., & Punnen, A. P. (2002). A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, *123*(1), 75–102.

Akbari, V., & Shiri, D. (2021). Weighted online minimum latency problem with edge uncertainty. *European Journal of Operational Research*, *295*, 51–65.

Akbari, V., & Shiri, D. (2022). An online optimization approach for post-disaster relief distribution with online blocked edges. *Computers & Operations Research*, *137*, 105533.

Aksakalli, V., Sahin, O. F., & Ari, I. (2016). An AO* based exact algorithm for the Canadian traveler problem. *INFORMS Journal on Computing*, *28*, 96–111.

Albers, S. (1999). Better bounds for online scheduling. *SIAM Journal on Computing*, *29*(2), 459–473.

Albers, S. (2003). Online algorithms: A survey. *Mathematical. Programming, Series. B*, *97*, 3–26.

Alipour, M., Mehrjedrdi, Y. Z., & Mostafaeipour, A. (2020). A rule-based heuristic algorithm for on-line order batching and scheduling in an order picking warehouse with multiple pickers. *RAIRO-Operations Research*, *54*(1), 101–107.

Allulli, L., & Laura, G. A. L. (2005). On the power of lookahead in on-line vehicle routing problems. *Computing and Combinatorics*, 728–736.

American Red Cross. (2015). *Drones for disaster response and relief operations* (Report). American Red Cross and Measure. http://www.issuelab.org/resources/21683/21683.pdf

Annear, L. M., Akhavan-Tabatabaei, R., & Schmid, V. (2023). Dynamic assignment of a multi-skilled workforce in job shops: An approximate dynamic programming approach. *European Journal of Operational Research*, *306*(3), 1109–1125.

Ascheuer, N., Krumke, S. O., & Rambau, J. (2000). Online dial-a-ride problems: Minimizing the completion time. *STACS 2000: 17th Annual Symposium on Theoretical Aspects of Computer Science Lille, France, February 17–19, 2000 Proceedings 17*, 639–650.

Ausiello, G., Feuerstein, E., Leonardi, S., Stougie, L., & Talamo, M. (2001). Algorithms for the on-line travelling salesman. *Algorithmica*, *29*, 560–581.

Azadeh, K., De Koster, R., & Roy, D. (2019). Robotized and automated warehouse systems: Review and recent developments. *Transportation Science*, *53*(4), 917–945.

Balas, E. (1999). New classes of efficiently solvable generalized traveling salesman problems. *Annals of Operations Research*, *86*, 529–558.

Balas, E., & Simonetti, N. (2001). Linear time dynamic-programming algorithms for new classes of restricted TSPs: A computational study. *INFORMS Journal on Computing*, *13*(1), 56–75.

Ball, M. O., & Queyranne, M. (2009). Toward robust revenue management: Competitive analysis of online booking. *Operations Research*, *57*(4), 950–963.

Barrett, É., Reiling, M., Mirhassani, S., Meijering, R., Jager, J., Mimmo, N., Callegati, F., Marconi, L., Carloni, R., & Stramigioli, S. (2018). Autonomous battery exchange of UAVs with a mobile ground base. *Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA)*, 699–705.

Baykasoğlu, A., Madenoğlu, F. S., & Hamzadayı, A. (2020). Greedy randomized adaptive search for dynamic flexible job-shop scheduling. *Journal of Manufacturing Systems*, *56*, 425–451.

Bellman, R. (1952). On the theory of dynamic programming. *Proceedings of the National Academy of Sciences of the United States of America*, *8*(8), 716–719.

Bent, R., & Van Hentenryck, P. (2004). Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research*, *52*, 977–987.

Bertsekas, D. P. (2020). *Dynamic programming and optimal control*. Athena Scientific.

Bertsimas, D., Jaillet, P., & Martin, S. (2019). Online vehicle routing: The edge of optimization in large-scale applications. *Operations Research*, *67*(1), 143–162.

Bertsimas, D. J., & van Ryzin, G. (1993). Stochastic and dynamic vehicle routing in the Euclidean plane with multiple capacitated vehicles. *Operations Research*, *41*, 60–76.

Betti Sorbelli, F., Pinotti, C. M., & Rigoni, G. (2023). On the evaluation of a drone-based delivery system on a mixed Euclidean-Manhattan grid. *IEEE Transactions on Intelligent Transportation Systems*, *24*(1), 1276–1287.

Bock, S., Bomsdorf, S., Boysen, N., & Schneider, M. (2024). A survey on the traveling salesman problem and its variants in a warehousing context. *European Journal of Operational Research*.

Bono, G., Dibangoye, J. S., Simonin, O., Matignon, L., & Pereyron, F. (2021). Solving multi-agent routing problems using deep attention mechanisms. *IEEE Transactions on Intelligent Transportation Systems*, *22*(12), 7804–7813.

Borodin, A., & El-Yaniv, R. (1998). *Online Computation and Competitive Analysis*. Cambridge University Press.

Bouman, P., Agatz, N., & Schmidt, M. (2018). Dynamic programming approaches for the traveling salesman problem with drone. *Networks*, *72*, 528–542.

Boysen, N., De Koster, R., & Weidinger, F. (2019). Warehousing in the e-commerce era: A survey. *European Journal of Operational Research*, *277*(2), 396–411.

Branke, J., Middendorf, M., Noeth, G., & Dessouky, M. (2005). Waiting strategies for dynamic vehicle routing. *Transportation Science*, *39*(3), 298–312.

Brueggemann, T., & Hurink, J. (2011). Matching based very large-scale neighborhoods for parallel machine scheduling. *Journal of Heuristics*, *17*, 637–658.

Brynjolfsson, E., Hu, Y., & Smith, M. D. (2003). Consumer surplus in the digital economy: Estimating the value of increased product variety at online booksellers. *Management Science*, *49*, 1580–1596.

Bukchin, Y., Khmelnitsky, E., & Yakuel, P. (2012). Optimizing a dynamic order-picking process. *European Journal of Operational Research*, *219*(2), 335–346.

Büttner, S., & Krumke, S. O. (2016). The Canadian tour operator problem on paths: Tight bounds and resource augmentation. *Journal of Combinatorial Optimization*, *32*, 842–854.

Cabreira, T. M., Brisolara, L. B., & Paulo, F. (2019). Survey on coverage path planning with unmanned aerial vehicles. *Drones*, *3*(1), 4.

Cals, B., Zhang, Y., Dijkman, R., & van Dorst, C. (2021). Solving the online batching problem using deep reinforcement learning. *Computers & Industrial Engineering*, *156*, 107221.

Cañas, H., Mula, J., Díaz-Madroñero, M., & Campuzano-Bolarín, F. (2021). Implementing industry 4.0 principles. *Computers & Industrial Engineering*, *158*, 107379.

Çatay, B., & Sadati, İ. (2023). An improved matheuristic for solving the electric vehicle routing problem with time windows and synchronized mobile charging/battery swapping. *Computers & Operations Research*, *159*, 106310.

Chan, H., & Wanab, M. I. M. (2024). A machine learning framework for predicting weather impact on retail sales. *Supply Chain Analytics*, *5*, 100058.

Chen, C.-M., Gong, Y., de Koster, R., & van Nunen, J. (2010). A flexible evaluative framework for order picking systems. *Production and Operations Management*, *19*, 70–82.

Chen, T.-L., Cheng, C.-Y., Chen, Y.-Y., & Chan, L.-K. (2015). An efficient hybrid algorithm for integrated order batching, sequencing and routing problem. *International Journal of Production Economics*, *159*, 158–167.

Chen, Y., & Xie, J. (2008). Online consumer review: Word-of-mouth as a new element of marketing communication mix. *Management Science*, *54*, 477–491.

Chowdhury, S., Emelogu, A., Marufuzzaman, M., Nurre, S. G., & Bian, L. (2017). Drones for disaster response and relief operations: A continuous approximation model. *International Journal of Production Economics*, *188*, 167–184.

Chung, S. H., Sah, B., & Lee, J. (2020). Optimization for drone and drone-truck combined operations: A review of the state of the art and future directions. *Computers & Operations Research*, *123*(4), 105004.

Corder, G., & Foreman, D. (2009). Comparing variables of ordinal or dichotomous scales: Spearman rank-order, point-biserial, and biserial correlations. In *Nonparametric statistics for non-statisticians* (pp. 122–154). John Wiley & Sons, Ltd.

Cui, R., Gallino, S., Moreno, A., & Zhang, D. J. (2018). The operational value of social media information. *Production and Operations Management*, *27*, 1749–1769.

Dai, H., Xiao, Q., Yan, N., Xu, X., & Tong, T. (2022). Item-level forecasting for e-commerce demand with high-dimensional data using a two-stage feature selection algorithm. *Journal of Systems Science and Systems Engineering*, *31*, 247–264.

Dauod, H., & Won, D. (2022). Real-time order picking planning framework for warehouses and distribution centres. *International Journal of Production Research*, *60*(18), 5468–5487.

De Koster, M. B. M., & Balk, B. M. (2008). Benchmarking and monitoring international warehouse operations in Europe. *Production and Operations Management*, *17*(2), 175–183.

De Koster, M. B. M., Van der Poort, E., & Wolters, M. (1999). Efficient orderbatching methods in warehouses. *International Journal of Production Research*, *37*(7), 1479–1504.

Dell'Amico, M., Montemanni, R., & Novellani, S. (2020). Matheuristic algorithms for the parallel drone scheduling traveling salesman problem. *Annals of Operations Research*, *289*, 211–226.

Desrochers, M., & Soumis, F. (1988). A generalized permanent labeling algorithm for the shortest path problem with time windows. *Information Systems and Operations Research*, *26*, 191–212.

D'Haen, R., Braekers, K., & Ramaekers, K. (2023). Integrated scheduling of order picking operations under dynamic order arrivals. *International Journal of Production Research*, *61*(10), 3205–3226.

Didden, J. B., Dang, Q.-.-V., & Adan, I. J. (2024). Enhancing stability and robustness in online machine shop scheduling: A multi-agent system and negotiation-based approach for handling machine downtime in industry 4.0. *European Journal of Operational Research*, *316*(2), 569–583.

Drexl, M. (2012). Synchronization in vehicle routing – A survey of VRPs with multiple synchronization constraints. *Transportation Science*, *46*(3), 297–316.

Drummer, R. (2023). Amazon cites warehouse changes in tripling profit, slashing shipping costs. *CoStar News*, *October 27*. https://www.costar.com/article/316365876/amazon-cites-warehouse-changes-in-tripling-profit-slashing-shipping-costs

Dukkanci, O., Koberstein, A., & Kara, B. Y. (2023). Drones for relief logistics under uncertainty after an earthquake. *European Journal of Operational Research*, *310*, 117–132.

Eliyan, A. F., & Kerbache, L. (2024). Vehicle relocation in one-way carsharing: A review. *Sustainability*, *16*(3), 1014.

Evers, L., Barros, A. I., Monsuur, H., & Wagelmans, A. (2014). Online stochastic UAV mission planning with time windows and time-sensitive targets. *European Journal of Operational Research*, *238*(1), 348–362.

FAA. (2022). Drones by the numbers. *Federal Aviation Administration, U.S. Department of Transportation, Accessed June 26 ,2022*. https://www.faa.gov/uas/resources

Faiz, T. I., Vogiatzis, C., Liu, J., & Noor-E-Alam, M. (2024). A robust optimization framework for two-echelon vehicle and UAV routing for post-disaster humanitarian logistics operations. *Networks*, *84*(2), 200–219.

Farahani, R. Z., Lotfi, M., Baghaian, A., Ruiz, R., & Rezapour, S. (2020). Mass casualty management in disaster scene: A systematic review of OR&MS research in humanitarian operations. *European Journal of Operational Research*, *287*, 787–819.

Farzaneh, M. A., Rezapour, S., Baghaian, A., & Amini, M. H. (2023). An integrative framework for coordination of damage assessment, road restoration, and relief distribution in disasters. *Omega*, *115*, 102748.

Feigin, P. D. (1979). On the characterization of point processes with the order statistic property. *Journal of Applied Probability*, *16*(2), 297–304.

Ferreira, K. J., Lee, B. H. A., & Simchi-Levi, D. (2016). Analytics for an online retailer: Demand forecasting and price optimization. *Manufacturing & Service Operations Management*, *18*, 69–88.

Feuerstein, E., & Stougie, L. (2001). On-line single-server dial-a-ride problems. *Theoretical Computer Science*, *268*, 91–105.

Fiat, A., & Woeginger, G. J. (1998). *Online algorithms* (Vol. 1442). Springer: Berlin.

Gademann, N., & van de Velde, S. (2005). Order batching to minimize total travel time in a parallel-aisle warehouse. *IIE transactions*, *37*(1), 63–75.

Galambos, G., & Woeginger, G. J. (1993). An on-line scheduling heuristic with better worst-case ratio than Graham's list scheduling. *SIAM Journal on Computing*, *22*(2), 349–355.

Galceran, E., & Carreras, M. (2013). A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, *61*, 1258–1276.

Gao, J., Zhen, L., & Wang, S. (2023). Multi-trucks-and-drones cooperative pickup and delivery problem. *Transportation Research Part C: Emerging Technologies*, *157*, 104407.

Gendreau, M., Guertin, F., Potvin, J.-Y., & Taillard, É. (1999). Parallel tabu search for real-time vehicle routing and dispatching. *Transportation science*, *33*(4), 381–390.

Giannikas, V., Lu, W., Robertson, B., & McFarlane, D. (2017). An interventionist strategy for warehouse order picking: Evidence from two case studies. *International Journal of Production Economics*, *189*, 63–76.

Gil-Borrás, S., Pardo, E. G., Alonso-Ayuso, A., & Duarte, A. (2020). GRASP with variable neighborhood descent for the online order batching problem. *Journal of Global Optimization*, *78*(2), 295–325.

Gil-Borrás, S., Pardo, E. G., Alonso-Ayuso, A., & Duarte, A. (2021). A heuristic approach for the online order batching problem with multiple pickers. *Computers & Industrial Engineering*, *160*, 107517.

Gil-Borrás, S., Pardo, E. G., Jiménez, E., & Sörensen, K. (2024). The time-window strategy in the online order batching problem. *International Journal of Production Research*, *62*(12), 4446–4469.

Gökalp, E., Gülpınar, N., & Doan, X. (2023). Dynamic surgery management under uncertainty. *European Journal of Operational Research*, *309*(2), 832–844.

Golabi, M., Shavarani, S. M., & Izbirak, G. (2017). An edge-based stochastic facility location problem in UAV-supported humanitarian relief logistics: A case study of Tehran earthquake. *Natural Hazards*, *87*, 1–21.

Gong, Y., & Gong, R. D. K. (2009). Approximate optimal order batch sizes in a parallel aisle warehouse. *Innovations in Distribution Logistics*, 175–194.

Gonzalez-R, P., Canca, D., Andrade, J., Calle, M., & Leon-Blanco, J. (2020). Truck-drone team logistics: A heuristic approach to multi-drop route planning. *Transportation Research Part C: Emerging Technologies*, *114*, 657–680.

Gonzalez-R, P. L., Sanchez-Wells, D., & Andrade-Pineda, J. L. (2024). A bi-criteria approach to the truck-multidrone routing problem. *Expert Systems with Applications*, *243*, 122809.

Goodson, J. C., Thomas, B. W., & Ohlmann, J. W. (2016). Restocking-based rollout policies for the vehicle routing problem with stochastic demand and duration limits. *Transportation Science*, *50*(2), 591–607.

Google Cloud Services. (2023). *Building real-time streaming pipelines for market data. Accessed August 27, 2024*. https://cloud.google.com/blog/topics/financial-services

Graham, R. L. (1969). Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, *17*(2), 416–429.

Grangier, P., Gendreau, M., Lehuédé, F., & Rousseau, L.-M. (2017). A matheuristic based on large neighborhood search for the vehicle routing problem with cross-docking. *Computers & Operations Research*, *84*, 116–126.

Guerrero, J. A., & Bestaoui, Y. (2013). UAV path planning for structure inspection in windy environments. *Journal of Intelligent and Robotic Systems*, *69*, 297–311.

Ha, Q. M., Deville, Y., Pham, Q. D., & Hà, M. H. (2018). On the min-cost traveling salesman problem with drone. *Transportation Research Part C: Emerging Technologies*, *86*, 597–621.

Hall, N. G., Posner, M. E., & Potts, C. N. (1998). Scheduling with finite capacity input buffers. *Operations Research*, *46*(3), 154–159.

Hamdan, I. K. A., Aziguli, W., Zhang, D., & Sumarliah, E. (2023). Machine learning in supply chain: prediction of real-time e-order arrivals using ANFIS. *International Journal of System Assurance Engineering and Management*, *14*, 549–568.

Henderson, J. (2020). Consumers want their deliveries to be free and fast. *Supply Chain Digital*, *May 17*. https://supplychaindigital.com/sustainability/consumers-want-their-deliveries-be-free-and-fast

Henn, S. (2012). Algorithms for on-line order batching in an order picking warehouse. *Computers & Operations Research*, *39*(11), 2549–2563.

Herrington, D. (2024). Amazon delivered to Prime members at the fastest speeds ever in 2023. *Amazon*, *January 30*. https://www.aboutamazon.com/news/operations/doug-herrington-amazon-prime-delivery-speed-2024-updates

Herrmann, J., Perez, F., Trautwein, V., & Weidmann, M. (2019). Getting a handle on warehousing costs. *McKinsey & Company*, *June 26*. https://www.mckinsey.com/capabilities/operations/our-insights/getting-a-handle-on-warehousing-costs#/

Hof, J., & Schneider, M. (2021). Intraroute resource replenishment with mobile depots. *Transportation Science*, *55*(3), 660–686.

Huang, T., & van Mieghem, J. A. (2014). Clickstream data and inventory management: Model and empirical analysis. *Production and Operations Management*, *23*, 333–347.

Hwang, D., & Jaillet, P. (2018). Online scheduling with multi-state machines. *Networks*, *71*(3), 209–251.

Hwang, D., Jaillet, P., & Manshadi, V. (2021). Online resource allocation under partially predictable demand. *Operations Research*, *69*(3), 895–915.

Ichoua, S., Gendreau, M., & Potvin, J.-Y. (2000). Diversion issues in real-time vehicle dispatching. *Transportation Science*, *34*(4), 426–438.

Irnich, S., & Desaulniers, G. (2005). Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, & M. M. Solomon (Eds.), *Column generation* (pp. 33–65). Springer US.

Jaillet, P., & Wagner, M. R. (2008a). Online vehicle routing problems: A survey. In B. Golden, S. Raghavan, & E. Wasil (Eds.), *The vehicle routing problem: Latest advances and new challenges* (pp. 221–237). Springer Science Publishers.

Jaillet, P., & Wagner, M. R. (2008b). Generalized online routing: New competitive ratios, resource augmentation, and asymptotic analyses. *Operations Research*, *56*(3), 745–757.

Jaillet, P., & Wagner, M. R. (2010). Almost sure asymptotic optimality for online routing and machine scheduling problems. *Networks*, *55*(1), 2–12.

Jalota, D., Paccagnan, D., Schiffer, M., & Pavone, M. (2023). Online routing over parallel networks: Deterministic limits and data-driven enhancements. *INFORMS Journal on Computing*, *35*(3), 560–577.

Jawhar, I., Mohamed, N., Al-Jaroodi, J., & Zhang, S. (2014). A framework for using unmanned aerial vehicles for data collection in linear wireless sensor networks. *Journal of Intelligent and Robotic Systems*, *74*, 437–453.

Jiang, J., Dai, Y., Yang, F., & Ma, Z. (2024). A multi-visit flexible-docking vehicle routing problem with drones for simultaneous pickup and delivery services. *European Journal of Operational Research*, *312*(1), 125–137.

Kahneman, D., & Tversky, A. (2000). Choices, Values, and Frames. Cambridge University Press: Cambridge.

Kalyanasundaram, B., & Pruhs, K. (1993). Online weighted matching. *Journal of Algorithms*, *14*(3), 478–488.

Karak, A., & Abdelghany, K. (2019). The hybrid vehicle-drone routing problem for pick-up and delivery services. *Transportation Research Part C: Emerging Technologies*, *102*, 427–449.

Karp, R. M., Vazirani, U. V., & Vazirani, V. (1990). An optimal algorithm for on-line bipartite matching. *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, 352–358.

Karp, R. M. (1992). On-line algorithms versus off-line algorithms: How much is it worth to know the future? *Proceedings of the IFIP 12th World Computer Congress on Algorithms, Software, Architecture - Information Processing '92, Volume 1 - Volume I*, 416–429.

Kleywegt, A. J., & Papastavrou, J. D. (1998). The dynamic and stochastic knapsack problem. *Operations Research*, *46*(1), 17–35.

Koutsoupias, E., & Papadimitriou, C. H. (1995). On the k-server conjecture. *Journal of the ACM*, *42*, 971–983.

Koutsoupias, E., & Papadimitriou, C. H. (2000). Beyond competitive analysis. *SIAM Journal on Computing*, *30*(1), 300–317.

Kundu, T., Sheu, J.-B., & Kuo, H.-T. (2022). Emergency logistics management – Review and propositions for future research. *Transportation Research Part E: Logistics and Transportation Review*, *164*, 102789.

Le Gall, J.-F. (2022). *Measure theory, probability, and stochastic processes* (Vol. 295). Springer Nature.

Le-Duc, T., & De Koster, R. M. (2007). Travel time estimation and order batching in a 2-block warehouse. *European Journal of Operational Research*, *176*(1), 374–388.

Lee, C. C., & Lee, D. T. (1985). A simple on-line bin-packing algorithm. *Journal of the ACM*, *32*(3), 562–572.

Letchford, A. N., Nasiri, S. D., & Theis, D. O. (2013). Compact formulations of the Steiner traveling salesman problem and related problems. *European Journal of Operational Research*, *228*(1), 83–92.

Li, H., Chen, J., Wang, F., & Bai, M. (2021). Ground-vehicle and unmanned-aerial-vehicle routing problems from two-echelon scheme perspective: A review. *European Journal of Operational Research*, *294*(3), 1078–1095.

Li, J.-Q., Mirchandani, P. B., & Borenstein, D. (2009). Real-time vehicle rerouting problems with time windows. *European Journal of Operational Research*, *194*(3), 711–727.

Li, K., Ni, W., Wang, X., Liu, R. P., Kanhere, S. S., & Jha, S. (2016). Energy-efficient cooperative relaying for unmanned aerial vehicles. *IEEE Transactions on Mobile Computing*, *15*(6), 1377–1386.

Li, Y., Zhang, R., & Jiang, D. (2022). Order-picking efficiency in e-commerce warehouses: A literature review. *Journal of Theoretical and Applied Electronic Commerce Research*, *17*(4), 1812–1830.

Liao, C.-S., & Huang, Y. (2014). The covering Canadian traveller problem. *Theoretical Computer Science*, *530*, 80–88.

Liu, Z., & Yu, W. (2000). Scheduling one batch processor subject to job release dates. *Discrete Applied Mathematics*, *105*(1-3), 129–136.

Löffler, M., Boysen, N., & Schneider, M. (2022). Picker routing in AGV-assisted order picking systems. *INFORMS Journal on Computing*, *34*(1), 440–462.

Löffler, M., Boysen, N., & Schneider, M. (2023). Human-robot cooperation: Coordinating autonomous mobile robots and human order pickers. *Transportation Science*, *57*(4), 839–1114.

Lorenz, C., Otto, A., & Gendreau, M. (2024). Exact solution approaches for the order batching, sequencing, and picker routing problem with release times. *Working* *paper series of the chair of Management Science, Operations- and Supply Chain Management of the University of Passau*. https://www.wiwi.uni-passau.de/en/management-science/research/publications

Lu, W., McFarlane, D., Giannikas, V., & Zhang, Q. (2016). An algorithm for dynamic order-picking in warehouse operations. *European Journal of Operational Research*, *248*(1), 107–122.

Macias, J. W., Goldbeck, N., Hsu, P., Angeloudis, P., & Ochieng, W. (2020). Endogenous stochastic optimisation for relief distribution assisted with unmanned aerial vehicles. *OR Spectrum*, *42*, 1089–1125.

Macrina, G., Di Puglia Pugliese, L., Guerriero, F., & Laporte, G. (2020). Drone-aided routing: A literature review. *Transportation Research Part C: Emerging Technologies*, *120*, 102762.

Madani, B., Ndiaye, M., & Salhi, S. (2024). Hybrid truck-drone delivery system with multi-visits and multi-launch and retrieval locations: Mathematical model and adaptive variable neighborhood search with neighborhood categorization. *European Journal of Operational Research*, *316*(1), 100–125.

Mak, V., & Boland, N. (2007). Polyhedral results and exact algorithms for the asymmetric travelling salesman problem with replenishment arcs. *Discrete Applied Mathematics*, *155*(16), 2093–2110.

Manasse, M., McGeoch, L., & Sleator, D. (1988). Competitive algorithms for on-line problems. *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, 322–333.

Marchet, G., Melacini, M., & Perotti, S. (2015). Investigating order picking system adoption: A case-study-based approach. *International Journal of Logistics Research and Applications*, *18*(1), 82–98.

Marrekchi, E., Besbes, W., Dhouib, D., & Demir, E. (2021). A review of recent advances in the operations research literature on the green routing problem and its variants. *Annals of Operations Research*, *304*, 1–46.

Masmoudi, M., Mancini, S., Baldacci, R., & Kuo, Y.-H. (2022). Vehicle routing problems with drones equipped with multi-package payload compartments. *Transportation Research Part E: Logistics and Transportation Review*, *164*, 102757.

Meng, S., Guo, X., Li, D., & Liu, G. (2023). The multi-visit drone routing problem for pickup and delivery services. *Transportation Research Part E: Logistics and Transportation Review*, *169*, 102990.

Meyersohn, N. (2023). Amazon is changing its deliveries behind the scenes to cut shipping times. *CNN*, *May 15*. https://edition.cnn.com/2023/05/15/business/amazon-delivery-packages/index.html

Mills, A. F., Argon, N. T., & Ziya, S. (2018). Dynamic distribution of patients to medical facilities in the aftermath of a disaster. *Operations Research*, *66*(3), 716–732.

Mitrović-Minić, S., & Laporte, G. (2004). Waiting strategies for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, *38*(7), 635–655.

Mohsan, S. A. H., Othman, N. Q. H., Khan, M. A., Amjad, H., & Żywiołek, J. (2022). A comprehensive review of micro UAV charging techniques. *Micromachines*, *13*(6).

Moreno, A., Alem, D., Ferreira, D., & Clark, A. (2018). An effective two-stage stochastic multi-trip location-transportation model with social concerns in relief supply chains. *European Journal of Operational Research*, *269*(3), 1050–1071.

Moreno, A., Alem, D., Gendreau, M., & Munari, P. (2020). The heterogeneous multicrew scheduling and routing problem in road restoration. *Transportation Research Part B: Methodological*, *141*, 24–58.

Moretti Branchini, R., Amaral Armentano, V., & Løkketangen, A. (2009). Adaptive granular local search heuristic for a dynamic vehicle routing problem. *Computers & Operations Research*, *36*(11), 2955–2968.

Moshref-Javadi, M., & Winkenbach, M. (2021). Applications and research avenues for drone-based models in logistics: A classification and review. *Expert Systems with Applications*, *177*, 114854.

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

Muter, İ., & Öncan, T. (2015). An exact solution approach for the order batching problem. *IIE Transactions*, *47*(7), 728–738.

Napolitano, M. (2012). 2012 warehouse/DC operations survey: Mixed signals. *Logistics Management (Highlands Ranch, Colo.: 2002)*, *51*(11).

National Information Standards Organization. (2022). *Credit: Contributor role taxonomy. Accessed: August 11, 2024*. https://credit.niso.org/

Ninikas, G., & Minis, I. (2014). Reoptimization strategies for a dynamic vehicle routing problem with mixed backhauls. *Networks*, *64*(3), 214–231.

Ojeda Rios, B. H., Xavier, E. C., Miyazawa, F. K., Amorim, P., Curcio, E., & Santos, M. J. (2021). Recent dynamic vehicle routing problems: A survey. *Computers & Industrial Engineering*, *160*, 107604.

Oruc, B. E., & Kara, B. Y. (2018). Post-disaster assessment routing problem. *Transportation Research Part B: Methodological*, *116*, 76–102.

Otto, A., Agatz, N., Campbell, J., Golden, B., & Pesch, E. (2018). Optimization approaches for civil applications of unmanned aerial vehicles (UAVs), or drones: A survey. *Networks*, *72*, 411–458.

Otto, A., Poikonen, S., & Golden, B. (2018). Planning deliveries in disaster relief by truck and drone. *Book of Extended Abstracts. Seventh International Workshop on Freight Transportation and Logistics (Odysseus 2018)*.

Ouelhadj, D., & Petrovic, S. (2009). A survey of dynamic scheduling in manufacturing systems. *Journal of Scheduling*, *12*, 417–431.

Özdamar, L., & Ertem, M. A. (2015). Models, solutions and enabling technologies in humanitarian logistics. *European Journal of Operational Research*, *244*, 55–65.

Palmer, A. (2024). Amazon reaches $2 trillion market cap for the first time. *CNBC, June 26*. https://www.cnbc.com/2024/06/26/amazon-reaches-2-trillion-market-cap-for-the-first-time.html

Papadimitriou, C. H., & Yannakakis, M. (1991). Shortest paths without a map. *Theoretical Computer Science*, *84*, 127–150.

Pardo, E. G., Gil-Borrás, S., Alonso-Ayuso, A., & Duarte, A. (2024). Order batching problems: Taxonomy and literature review. *European Journal of Operational Research*, *313*(1), 1–24.

Pavone, M., Saberi, A., Schiffer, M., & Tsao, M. W. (2022). Technical note – online hypergraph matching with delays. *Operations Research*, *70*(4), 2194–2212.

Pavone, M., Smith, S. L., Frazzoli, E., & Rus, D. (2012). Robotic load balancing for mobility-on-demand systems. *The International Journal of Robotics Research*, *31*(7), 839–854.

Pérez-Rodríguez, R., Hernández-Aguirre, A., & Jöns, S. (2015). A continuous estimation of distribution algorithm for the online order-batching problem. *The International Journal of Advanced Manufacturing Technology*, *79*, 569–588.

Phuyal, S., Bista, D., & Bista, R. (2020). Challenges, opportunities and future directions of smart manufacturing: A state of art review. *Sustainable Futures*, *2*, 100023.

Pillac, V., Gendreau, M., Guéret, C., & Medaglia, A. L. (2013). A review of dynamic vehicle routing problems. *European Journal of Operational Research*, *225*(1), 1–11.

Poikonen, S., & Golden, B. (2020). Multi-visit drone routing problem. *Computers & Operations Research*, *113*, 104802.

Powell, W. B. (2011). *Approximate dynamic programming: Solving the curses of dimensionality*. John Wiley & Sons, Inc.

Psaraftis, H. N. (Ed.). (2016). *Green transportation logistics: The quest for win-win solutions*. Springer International Publishing.

Pugliese, L. D. P., & Guerriero, F. (2013). A survey of resource constrained shortest path problems: Exact solution approaches. *Networks*, *62*(3), 183–200.

Raeesi, R., & Zografos, K. G. (2020). The electric vehicle routing problem with time windows and synchronised mobile battery swapping. *Transportation Research Part B: Methodological*, *140*, 101–129.

Raeesi, R., & Zografos, K. G. (2022). Coordinated routing of electric commercial vehicles with intra-route recharging and en-route battery swapping. *European Journal of Operational Research*, *301*(1), 82–109.

Rath, S., Gendreau, M., & Gutjahr, W. J. (2016). Bi-objective stochastic programming models for determining depot locations in disaster relief operations. *International Transactions in Operational Research*, *23*, 997–1023.

Ratliff, H. D., & Rosenthal, A. S. (1983). Order-picking in a rectangular warehouse: A solvable case of the traveling salesman problem. *Operations research*, *31*(3), 507–521.

Ren, X.-X., Fan, H.-M., Bao, M.-X., & Fan, H. (2023). The time-dependent electric vehicle routing problem with drone and synchronized mobile battery swapping. *Advanced Engineering Informatics*, *57*, 102071.

Reyes-Rubiano, L., Voegl, J., & Hirsch, P. (2022). An online algorithm for routing an unmanned aerial vehicle for road network exploration operations after disasters under different refueling strategies. *Algorithms*, *15*(6), 217.

Reyes-Rubiano, L., Voegl, J., Rest, K.-D., Faulin, J., & Hirsch, P. (2021). Exploration of a disrupted road network after a disaster with an online routing algorithm. *OR Spectrum*, *43*, 1–38.

Roodbergen, K. J., & De Koster, R. (2001). Routing methods for warehouses with multiple cross aisles. *International Journal of Production Research*, *39*(9), 1865–1883.

Rudin, J. (2001). *Improved bounds for the on-line scheduling problem, ph.d. thesis*. The University of Texas at Dallas.

S. O'Neill. (2024). How Thomas Hoe helps Amazon understand european customers. *Amazon*, *March 21*. https://www.amazon.science/working-at-amazon/how-thomas-hoe-helps-amazon-understand-european-customers

Sadana, U., Chenreddy, A., Delage, E., Forel, A., Frejinger, E., & Vidal, T. (2024). A survey of contextual optimization methods for decision-making under uncertainty. *European Journal of Operational Research*, 271–289.

Satton, R. S., & Barto, A. G. (2018). *Reinforcement learning*. MIT Press.

Schiffer, M., Boysen, N., Klein, P. S., Laporte, G., & Pavone, M. (2022). Optimal picking policies in e-commerce warehouses. *Management Science*, *68*(10), 7497–7517.

Schiffer, M., Schneider, M., Walther, G., & Laporte, G. (2019). Vehicle routing and location routing with intermediate stops: A review. *Transportation Science*, *53*(2), 319–343.

Schilde, M., Doerner, K., & Hartl, R. (2014). Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem. *European Journal of Operational Research*, *238*(1), 18–30.

Schleyer, M., & Gue, K. (2012). Throughput time distribution analysis for a one-block warehouse. *Transportation Research Part E: Logistics and Transportation Review*, *48*(3), 652–666.

Schneider, V. E., Au, C. N., Budak, A., & Oeffner, J. (2022). The development of a battery hot swap prototype for use on the autonomous surface vehicle SeaML: SeaLion. *Proceedings of the 14th Symposium on High-Performance Marine Vehicles, HIPER'22, Cortona, 29-31 Aug 2022.*

Scholz, A., Schubert, D., & Wäscher, G. (2017). Order picking with multiple pickers and due dates–simultaneous solution of order batching, batch assignment and sequencing, and picker routing problems. *European Journal of Operational Research*, *263*(2), 461–478.

Shelke, O., Baniwal, V., & Khadilkar, H. (2021). Anticipatory decisions in retail e-commerce warehouses using reinforcement learning. *Proceedings of the 3rd ACM India Joint International Conference on Data Science & Management of Data (8th ACM IKDD CODS & 26th COMAD)*, 272–280.

Shin, Y., Kim, S., & Moon, I. (2019). Integrated optimal scheduling of repair crew and relief vehicle after disaster. *Computers & Operations Research*, *105*, 237–247.

Sleator, D. D., & Tarjan, R. E. (1985). Amortized efficiency of list update and paging rules. *Communications of the ACM*, *28*(2), 202–208.

Sluijk, N., Florio, A. M., Kinable, J., Dellaert, N., & Van Woensel, T. (2023). Two-echelon vehicle routing problems: A literature review. *European Journal of Operational Research*, *304*(3), 865–886.

Smith, O., Boland, N., & Waterer, H. (2012). Solving shortest path problems with a weight constraint and replenishment arcs. *Computers & Operations Research*, *39*(5), 964–984.

Soares, R., Marques, A., Amorim, P., & Parragh, S. N. (2023). Synchronisation in vehicle routing: Classification schema, modelling framework and literature review [In Press]. *European Journal of Operational Research*.

Soeffker, N., Ulmer, M. W., & Mattfeld, D. C. (2022). Stochastic dynamic vehicle routing in the light of prescriptive analytics: A review. *European Journal of Operational Research*, *298*(3), 801–820.

Solomon, M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, *35*, 254–265.

Srinivas, S., Ramachandiran, S., & Rajendran, S. (2022). Autonomous robot-driven deliveries: A review of recent developments and future directions. *Transportation Research Part E: Logistics and Transportation Review*, *165*, 102834.

Steinker, S., Hoberg, K., & Thonemann, U. W. (2017). The value of weather information for e-commerce operations. *Production and Operations Management*, *26*, 1854–1874.

Stolaroff, J. K., Samaras, C., O'Neill, E. R., Lubers, A., Mitchell, A. S., & Ceperley, D. (2018). Energy use and life cycle greenhouse gas emissions of drones for commercial package delivery. *Nature Communications.*, *9*, 409.

Taillard, É. D. (2023). Local search. In *Design of heuristic algorithms for hard optimization: With python codes for the travelling salesman problem* (pp. 103–129). Springer International Publishing.

Tarantilis, C. D., Zachariadis, E. E., & Kiranoudis, C. T. (2008). A hybrid guided local search for the vehicle-routing problem with intermediate replenishment facilities. *INFORMS Journal on Computing*, *20*(1), 154–168.

Tian, J., Cheng, T., Ng, C., & Yuan, J. (2012). An improved on-line algorithm for single parallel-batch machine scheduling with delivery times. *Discrete Applied Mathematics*, *160*(7), 1191–1210.

Ulmer, M. W., Mattfeld, D. C., & Köster, F. (2018). Budgeting time for dynamic vehicle routing with stochastic customer requests. *Transportation Science*, *52*(1), 20–37.

Ulmer, M. W., & Thomas, B. W. (2018). Same-day delivery with heterogeneous fleets of drones and vehicles. *Networks*, *72*(4), 475–505.

Ulmer, M. W., Thomas, B. W., Campbell, A. M., & Woyak, N. (2021). The restaurant meal delivery problem: Dynamic pickup and delivery with deadlines and random ready times. *Transportation Science*, *55*(1), 75–100.

Ulmer, M. W. (2017). *Approximate Dynamic Programming for Dynamic Vehicle Routing* (Vol. 61). Springer.

Valle, C. A., Beasley, J. E., & Da Cunha, A. S. (2016). Modelling and solving the joint order batching and picker routing problem in inventories. *Combinatorial Optimization*, 81–97.

Valle, C. A., Beasley, J. E., & Da Cunha, A. S. (2017). Optimally solving the joint order batching and picker routing problem. *European Journal of Operational Research*, *262*(3), 817–834.

Van Gils, T., Caris, A., Ramaekers, K., & Braekers, K. (2019). Formulating and solving the integrated batching, routing, and picker scheduling problem in a real-life spare parts warehouse. *European Journal of Operational Research*, *277*(3), 814–830.

Van Nieuwenhuyse, I., & De Koster, R. B. (2009). Evaluating order throughput time in 2-block warehouses with time window batching. *International Journal of Production Economics*, *121*(2), 654–664.

Van Steenbergen, R., Mes, M., & van Heeswijk, W. (2023). Reinforcement learning for humanitarian relief distribution with trucks and UAVs under travel time uncertainty. *Transportation Research Part C: Emerging Technologies*, *157*.

Vanheusden, S., van Gils, T., Ramaekers, K., Cornelissens, T., & Caris, A. (2023). Practical factors in order picking planning: State-of-the-art classification and review. *International Journal of Production Research*, *61*(6), 2032–2056.

Vansteenwegen, P., Souffriau, W., & Oudheusden, D. V. (2011). The orienteering problem: A survey. *European Journal of Operational Research*, *209*(1), 1–10.

Vidal, T., Laporte, G., & Matl, P. (2020). A concise guide to existing and emerging vehicle routing problem variants. *European Journal of Operational Research*, *286*(2), 401–416.

Vodopivec, N., & Miller-Hooks, E. (2017). An optimal stopping approach to managing travel-time uncertainty for time-sensitive customer pickup. *Transportation Research Part B: Methodological*, *102*, 22–37.

Voigt, S. (2024). A review and ranking of operators in adaptive large neighborhood search for vehicle routing problems. *European Journal of Operational Research*.

Wahlen, J., & Gschwind, T. (2023). Branch-price-and-cut-based solution of order batching problems. *Transportation Science*, *57*(3), 756–777.

Wang, X., Crainic, T. G., & Wallace, S. W. (2019). Stochastic network design for planning scheduled transportation services: The value of deterministic solutions. *INFORMS Journal of Computing*, *31*, 1–192.

Weibo Ren, Y. H., Yan Yan, & Guan, Y. (2022). Joint optimisation for dynamic flexible job-shop scheduling problem with transportation time and resource constraints. *International Journal of Production Research*, *60*(18), 5675–5696.

Weidinger, F., & Boysen, N. (2018). Scattered storage: How to distribute stock keeping units all around a mixed-shelves warehouse. *Transportation Science*, *52*, 1412–1427.

Williamson, D. P., & Shmoys, D. B. (2011). *The design of approximation algorithms*. Cambridge University Press.

WMO. (2021). *Weather-related disasters increase over past 50 years, causing more damage but fewer deaths* (tech. rep.) (https://public.wmo.int). World Meterological Organization.

Wolpert, D., & Macready, W. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, *1*(1), 67–82.

Xiao, J., Liu, X., Liu, T., Li, N., & Martinez-Sykora, A. (2024). The electric vehicle routing problem with synchronized mobile partial recharging and non-strict waiting strategy. *Annals of Operations Research*, 1–45.

Xie, L., Li, H., & Luttmann, L. (2023). Formulating and solving integrated order batching and routing in multi-depot AGV-assisted mixed-shelves warehouses. *European Journal of Operational Research*, *307*(2), 713–730.

Xu, Y., Hu, M., Su, B., Zhu, B., & Zhu, Z. (2009). The Canadian traveller problem and its competitive analysis. *Journal of Combinatorial Optimization*, *18*, 195–205.

Yang, J., Jaillet, P., & Mahmassani, H. (2004). Real-time multivehicle truckload pickup and delivery problems. *Transportation Science*, *38*(2), 135–148.

Yang, M., Kumar, S., Wang, X., & Fry, M. J. (2024). Scenario-robust pre-disaster planning for multiple relief items. *Annals of Operations Research*, *335*(3), 1241–1266.

Yin, Y., Li, D., Wang, D., Ignatius, J., Cheng, T., & Wang, S. (2023). A branch-and-price-and-cut algorithm for the truck-based drone delivery routing problem with time windows. *European Journal of Operational Research*, *309*(3), 1125–1144.

Yu, M., & De Koster, R. (2008). Performance approximation and design of pick-and-pass order picking systems. *IIE Transactions*, *40*(11), 1054–1069.

Yu, M., & De Koster, R. B. (2009). The impact of order batching and picking area zoning on order picking system performance. *European Journal of Operational Research*, *198*(2), 480–490.

Yucesoy, E., Balcik, B., & Coban, E. (2024). The role of drones in disaster response: A literature review of operations research applications [Article in Press]. *International Transactions in Operational Research*.

Zeng, F., Chen, Z., Clarke, J.-P., & Goldsman, D. (2022). Nested vehicle routing problem: Optimizing drone-truck surveillance operations. *Transportation Research Part C: Emerging Technologies*, *139*, 103645.

Zhang, G., Jia, N., Zhu, N., Adulyasak, Y., & Ma, S. (2023). Robust drone selective routing in humanitarian transportation network assessment. *European Journal of Operational Research*, *305*, 400–428.

Zhang, G., Cai, X., & Wong, C. (2001). On-line algorithms for minimizing makespan on batch processing machines. *Naval Research Logistics*, *48*(3), 241–258.

Zhang, G., Zhu, N., Ma, S., & Xia, J. (2021). Humanitarian relief network assessment using collaborative truck-and-drone system. *Transportation Research Part E: Logistics and Transportation Review*, *152*(102417).

Zhang, H., Tong, W., Xu, Y., & Lin, G. (2015). The Steiner traveling salesman problem with online edge blockages. *European Journal of Operational Research*, *243*, 30–40.

Zhang, H., Tong, W., Lin, G., & Xu, Y. (2019). Online minimum latency problem with edge uncertainty. *European Journal of Operational Research*, *273*, 418–429.

Zhang, H., Tong, W., Xu, Y., & Lin, G. (2016). The Steiner traveling salesman problem with online advanced edge blockages. *Computers & Operations Research*, *70*, 26–38.

Zhang, J., & Woensel, T. V. (2023). Dynamic vehicle routing with random requests: A literature review. *International Journal of Production Economics*, *256*, 108751.

Zhang, J., Jia, L., Niu, S., Zhang, F., Tong, L., & Zhou, X. (2015). A space-time network-based modeling framework for dynamic unmanned aerial vehicle routing in traffic incident monitoring applications. *Sensors*, *15*(6), 13874–13898.

Zhang, J., Wang, X., Chan, F. T., & Ruan, J. (2017). On-line order batching and sequencing problem with multiple pickers: A hybrid rule-based algorithm. *Applied Mathematical Modelling*, *45*, 271–284.

Zhang, J., Wang, X., & Huang, K. (2016). Integrated on-line scheduling of order batching and delivery under B2C e-commerce. *Computers & Industrial Engineering*, *94*, 280–289.

Zhang, Y., Zhang, Z., Liu, Z., & Chen, Q. (2022). An asymptotically tight online algorithm for m-Steiner Traveling Salesman Problem. *Information Processing Letters*, *174*(106177).

Zhong, S., Cheng, R., Jiang, Y., Wang, Z., Larsen, A., & Nielsen, O. A. (2020). Risk-averse optimization of disaster relief facility location and vehicle routing under stochastic demand. *Transportation Research Part E: Logistics and Transportation Review*, *141*, 102015.

Žulj, I., Salewski, H., Goeke, D., & Schneider, M. (2022). Order batching and batch sequencing in an AMR-assisted picker-to-parts system. *European Journal of Operational Research*, *298*(1), 182–201.