EMERGENCY RESPONSE NETWORKS: DYNAMIC DISPATCHING AND RELOCATION OF RESOURCES

Dmitrii Usanov

The research presented in this dissertation was sponsored by NWO,
under the grant number 438-15-506.

Cover design: Brenda Marcela Perrusquía Tejeida
Email: brenda.perrusquia@hotmail.com | *Instagram*: brend_perrusquia

Printed by: Ridderprint BV, The Netherlands

VRIJE UNIVERSITEIT

EMERGENCY RESPONSE NETWORKS: DYNAMIC DISPATCHING AND RELOCATION OF RESOURCES

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad Doctor
aan de Vrije Universiteit Amsterdam,
op gezag van de rector magnificus
prof.dr. V. Subramaniam,
in het openbaar te verdedigen
ten overstaan van de promotiecommissie
van de Faculteit der Bètawetenschappen
op maandag 11 mei 2020 om 13.45 uur
in de aula van de universiteit,
De Boelelaan 1105

door

Dmitrii Usanov

geboren te Rjazan, Rusland

promotor: prof.dr. R.D. van der Mei
copromotor: dr.ir. P.M. van de Ven

Acknowledgements

My PhD journey has come to an end. This dissertation is an anthology of the research work I have done during that period. It wouldn't be possible without all the wonderful people who surrounded and supported me along the way. I want to take a moment and express my sincere gratitude to them.

First and foremost, I would like to express my deep appreciation to my supervisors Rob van der Mei and Peter van de Ven. Thank you for granting me the opportunity to work with you, for sharing your knowledge and experience, and for all the incredible support throughout these years. Rob, you are genuinely passionate about your work and have a unique ability to bring people and ideas together. Your positive attitude, optimism and everlasting energy are truly motivating. Thank you for giving me the chance in the first place, for the guidance and expertise you provided me with. I believe the phrase "thinking out loud" is doomed to be forever associated with you in my mind. Peter, your calm and wisdom always inspired confidence in me. I admire how you combine razor sharp aptitude and reasoning with modesty. Thank you for all the discussions we had, for your invaluable advice, your fine sense of humor, and, most importantly, for your patience.

I would also like to thank the members of my thesis committee: Ger Koole, Geert-Jan van Houtum, Rudesindo Núñez Queija, Jan van Doremalen, and Pieter van den Berg for the time and effort put into reading this thesis, and for all the valuable comments. Special thanks to Geert-Jan van Houtum and Jan van Doremalen for playing an important role in facilitating the DynaMerge project, and for the insightful feedback on my research during the project meetings.

I want to acknowledge Anya Pechina for the work that led to Chapters 4 and 5 of this thesis, everyone involved in the DynaMerge project for their influence on my research, NWO (Nederlandse Organisatie voor Wetenschappelijk Onderzoek) for funding

the project, and my research institute CWI (Centrum Wiskunde & Informatica). CWI has been an excellent workplace, with great opportunities for personal and professional development, full of inspiring people, friendly and helpful personnel.

I feel happy and incredibly privileged to have met some truly nice people, colleagues and friends, along the way. Guido, the superman data scientist from the Fire Department of Amsterdam-Amstelland, I feel honored to get to know you and your sweet family. I love your open, bright and sincere personality. It is a mystery to me how a person can be as dedicated and enthusiastic all the time. Thank you for your ideas and assistance in the work that led to Chapter 2 of this thesis, and thank you for always making me feel important and contributing (by constantly overstating my abilities :)).

Mihail, Chang-Hun, Ewan, Bohan, Paro, Tommaso, Alessandro, Marco, Anton, Pieter, Bart and many other colleagues from CWi, thank you for making me feel comfortable from day one, for the friendly and enjoyable atmosphere at work and outside work. My office mates Sara and Dirk, thanks for keeping it casual and fun in the office.

Mihail, I feel extremely proud I was the one to teach you and make you excel in so many things, such as playing backgammon, ping pong, chess, squash, etc. "Always two there are, no more, no less. A master and an apprentice". Sorry, I couldn't resist to put it out there as an eternal punch line. Jokes aside, I am truly grateful to have you as a friend. Thank you for all the good talking (for life, math and nonsense), for being there (in fun and sad times).

Chang-Hun, you are a complete paradox to me, I am still wondering if you are actually a human. I love that crazy yet wonderful mix of qualities you possess, and I am very happy to get to know you. Thank you for the times we spent together, I hope there will be more. I am also grateful to you, together with Tommaso and Alessandro, for being so persistent in bringing me to the bouldering gym. Special thanks to Paro for taking it to the next level and introducing me to sport climbing, particularly outdoors. Four years later, it is still a great passion of mine, thanks in large part to you.

Ewan, thank you for sharing your love for classical music with me, and also for your inconceivable longing for and resilience to sarcasm. Bohan, thank you for being there with me to exploit Ewan's resilience. Again, jokes aside, I am happy to get to know both of you. Thank you for sharing your thoughts with me, for the interesting conversations we had, and for all the time we spent together.

To all my climbing friends and buddies, Paro, Pesho, Shadi, Eduard, Laurie, Francois, Lazlo, Federico, Martin, Max, Gorjan, Vanessa, Angelica, and many others, thank you for all the good times, for indulging my passion for climbing, and for keeping me sane and fresh minded during these years.

Victor, thank you for being my friend throughout many years. Despite the distance and the long periods of time without seeing each other, I will always know there is a loving friend out there. I learned a lot from you and am still learning through the memories (you know, I am lagging about ten years behind your level of mental development). Knowing

you and having you as a friend always helps me to overcome whatever difficulties I am facing.

Grace, you brought so much new and beautiful to my life. Thank you for showing me a different side of things, for teaching me harmony and appreciation, for inspiring calm and confidence in me, and for sharing your love and support. You helped me to grow as a person, and thanks to you, I might even cut the lag from Victor by a couple of years.

Of course, I would have never made it to this point without the love and care of my family. I will forever be grateful to my mother Irina and to my grandparents Nina and Peter. Thank you for everything you did, for every sacrifice you made for me. It is only due to your unconditional and selfless support and dedication I could get all the chances I had in my life. Thank you to my brother Yuri and my sister Helena for showing me how to appreciate those chances. Yura, special thanks to you for all the support, and to a certain extent, for playing a role of a father for me, alongside with our grandfather.

Dmitrii Usanov
Amsterdam, March 2020

Contents

1 Introduction

The occurrence of unexpected emergencies may have a major impact on our daily lives, and in many cases short response times are crucial. In severe, life-threatening emergency situations where every second counts, the timely presence of medical aid, firefighters or policemen may make the difference between survival and death. In other areas, emergency incidents may cause considerable disruptions, e.g., in case of failure of production facilities, medical equipment, communication services, financial services and public transport.

At a high level, these examples share the following important characteristics. In an emergency situation, the time needed for the *resources* (e.g., firefighters, doctors, police, repairmen, equipment) to arrive at the *emergency location* should be below some *response time* target value. To ensure short response times, the resources are located at a number of *base locations* (fire or police stations, warehouses) spread across the region of service. Throughout this thesis, networks in which such *emergency incidents* occur will be referred to as *Emergency Response Networks* (ERN's).

In this thesis, we address operational questions ultimately aimed at reducing response times in ERN's. More specifically, we focus of the following two powerful means to improve the responsiveness of ERN's:

1. *dispatching* of resources: "Which resources to be sent to which emergency incident?"
2. *proactive relocation* of resources: "Where to locate/relocate the resources in order to properly anticipate future incidents?"

1.1 Maintenance and Emergency Services

In practice, there are many types of emergency incidents, and there is a variety of services available to deal with them. Two important service categories are *maintenance services* and *emergency services*. Maintenance services are related to keeping capital goods such as machines, trains and aircraft operational, and emergency services refer to application areas such as ambulance and firefighter services, police surveillance and road-side assistance.

1.1.1 Maintenance Services

In the high-tech industry, expensive and complex expensive devices are built, called *capital goods*. These are durable goods used for production or services, such as aircraft, lithography machines and MRI-scanners. The availability of capital goods is crucial to keep the primary processes of their owners and/or users up and running. For instance, the inconvenience of aircraft not being operational when needed is not only inconvenient to travelers, but also causes a significant loss of revenue for airlines. Moreover, it leads to operational costs related to rescheduling corresponding flights. Lithography machines form a bottleneck production step of semi-conductor manufacturing: when a lithography machine is down, it may cause the standstill of an entire factory. The unavailability of MRI-scanners may lead to high costs as well, in addition to inconveniencing patients in the best case, and medical emergencies in the worst case.

Keeping capital goods operational is often costly. In [81] and [116] the authors estimate that the costs of maintenance and unavailability of a capital asset over its lifetime (typically at least a decade) is three to four times the acquisition costs. Spare parts for capital goods are among the main sources of costs, and any delay in delivering the spare parts significantly contributes to the unavailability. In fact, in 2003 spare part sales and services (mostly maintenance) accounted for 8% of the gross domestic product in the United States [119]. More recently, US Bancorp estimated that the yearly expenditure in the US on spare parts amounts to 700 billion US dollars, which is 8% of the US gross domestic product [51]. Service accounts for 25% of total revenue for 120 large manufacturing companies in America, Asia Pacific, and Europe, and after-sales services account for 40% of the profit for these 120 companies [61].

We distinguish between *preventive* and *corrective* maintenance. There is a large body of research literature devoted to preventive maintenance (see, for example, [107] and [120] for overviews), where the goal is to develop policies that are able to detect and prevent potential breakdowns before they occur. It is often implemented in practice as a periodic review policy, where a repairman regularly checks the equipment condition, and decides upon a review whether to repair it or not. Corrective maintenance takes

place when, apart from the "planned" maintenance, a defect suddenly occurs. In this case, a maintenance provider must be able to deliver and repair the spare part very quickly, often within a few hours. The maintenance part of this thesis is focused primarily on optimization and development of heuristics for corrective maintenance operations. Corrective maintenance is an inevitable part of maintenance services and its optimization can be done independently from preventive maintenance.

Because the need for spare parts is uncertain (as it is unknown when machines break down), and the spare parts should be delivered in a timely manner, after-sales departments store resources (spare parts, tools and service engineers) in warehouses near the places where such need can arise. These spare parts are used to service for instance MRI-scanners at hospitals that contracted after-sales services. The stock in these warehouses is resupplied with a certain lead time whenever demand occurs. Sometimes the spare part needed to repair an MRI-scanner is not available at the nearest stock point, and a replacement part is shipped from a more distant warehouse that does have it in stock. Such shipments are called *lateral transshipments*, and are an important distinguishing feature of networks of spare part stocks. This feature enables not only efficient pooling of spare part stocks, but also complicates the analysis and control of such networks because stock points cannot be analyzed in isolation.

Most of the spare parts management research literature, spanning several decades, is devoted to the tactical decisions making. The main focus is on optimization of local inventory management policies [18, 39]. One of the first models of this type was introduced as early as 1968 [95], followed by a number of extensions (see, e.g., [75, 33]). For a detailed overview of the spare parts inventory management models we refer to [117]. The strategic decision of facility location for spare parts networks has received relatively little attention, although some research looked into jointly solving the two problems of network design and inventory stocking [14, 52, 90, 122]. Interestingly enough, some of these are based on the previous work on facility location for Emergency Medical Services (EMS). Limited research was devoted to operational level decision making for spare parts. In recent years, dynamic rules for dispatching and relocation of spare parts seems to attract more attention in the research literature [84, 101, 28, 74].

1.1.2 Emergency Services

The type of operational challenges observed in corrective maintenance appears in many other logistics-related applications, for instance with ambulance, firefighter, police-surveillance and road-side assistance services. In life-threatening emergencies, the ability of ambulance and firefighter service providers to arrive at the emergency scene within a few

minutes is crucial. In practice, a commonly used service-level target is that the response time for high-priority calls should be below some threshold, which depends on the type of emergency. To enable such extremely short response times (i.e., the time between a call and the moment when service is offered at the scene of the emergency) at affordable cost, efficient planning of emergency services is crucial. To this end, emergency service providers face a variety of planning problems at the strategic, tactical, and operational level. For example, consider fire fighting services. In the Netherlands, there is a requirement by law that a fire has to be responded to within 5-10 minutes, depending on the type of call. In order to meet this requirement, there are several fire stations spread around each region. Each station has a number of trucks waiting and ready. When an emergency call is placed, one or more fire trucks are dispatched based on proximity to the emergency. Sufficient equipment must be available to be able to deal with major incidents adequately, as those might require multiple fire trucks simultaneously. Fire trucks must be properly distributed in order to be able to respond to a possible second incident elsewhere.

Independently and in parallel with the developments in spare part management, researchers have been working on models and algorithms for EMS management. Much of this work was devoted to determining the optimal location of base stations, where ambulances await new emergencies. The pioneering models were introduced in the early 1970s [102, 17] and were followed by numerous extensions and case studies. This research literature on strategic decision making gave rise to the development of models for real-time dispatching and relocation of emergency vehicles, with one of the first dynamic relocation models introduced in [30]. Dynamic ambulance management is an active research area, with some recent works including [47, 110, 112, 99, 79]. For overviews of location, relocation and dispatching literature for EMS we recommend [12, 7].

Combining areas of expertise: Despite many similarities in how maintenance and emergency services operate, so far the research in these areas has been done relatively independent from each other, in particular at the operational level. We believe that merging such independent streams of research with apparent commonalities can contribute to development of better methods in each area, as well as propagate and evolve the methodology into other application areas with similar features. This thesis is the first attempt in doing so. We aim to get inspiration from the research done in one application area to address a problem arising in another one.

1.2 Focus and Research Questions

In this thesis we focus on how to exploit the potential of *dispatching* and *relocation* of resources in ERN's by developing new models and optimization methods.

Dispatching: In practice, *dispatching* of resources is often done according to some static rule, where for each potential location of an incident there is a predefined and fixed dispatching order. One example of such policy is the *closest-first* policy, where the closest available resource is dispatched to the incident location. Such a policy, however, does not take into account important spatial properties of ERN's, such as traveling times, differences in incident arrival rates across the region, current location and availability of resources. This can result in coverage gaps in some areas, and, therefore, may lead to large response times for later incidents. Dynamic state-dependent dispatching policies can be useful to maintain good coverage over time and boost the performance of ERN's.

Relocation: Another way to maintain good coverage in real time is by proactive *relocation* of resources between different storage locations. Relocation actions can be used either to compensate for the imbalances caused by static dispatching, or it can complement a dynamic dispatching policy. Often, dispatching options are limited by the service targets such as maximum response time. It can be that such restrictions still result in an unfavorable allocation of resources. This problem can be mitigated by relocating resources from well covered areas to those with a shortage. Both dynamic *dispatching* and *relocation* policies introduce a great deal of flexibility in ERN operations management. There is an enormous potential in improving the overall ERN performance given this flexibility, and we are looking into the question of how to best exploit it.

Research questions: At a high level, the research questions that we address in this thesis are:

1. *How to proactively relocate resources during major and long-lasting incidents?*
2. *How to deal with uncertainty in travel times?*
3. *How to optimally combine dispatching and proactive relocation of resources?*
4. *How to strike a good balance between prevention of, and response to, emergency incidents?*

To address these questions, we will focus on two different examples of such application areas, namely, maintenance services for capital goods and firefighting emergency services. In doing so, we look at both application areas mentioned above, acquiring data and insights from the Fire Department of Amsterdam-Amstelland (FDAA) and Philips Healthcare. We address a number of practical operational problems from firefighter services and spare parts management, combining methodology and ideas from both areas of expertise.

1.3 Outline of the Thesis

The research presented in this thesis is organized in five chapters. Chapters 2, 3, 4 and 6 can be read separately, while reading Chapter 5 should follow Chapter 4. Below we outline our contributions.

Optimal resource relocation during major emergency incidents

Fire departments have to deal with the uncertainty of not just the location and time of an incident, but also its severity. Occasionally, *large fires* occur that might require multiple fire trucks for an extended period of time. The current practice of the fire department is to dispatch the closest vehicles to the incident location, but this leads to significant gaps in coverage. The question then arises, how to protect against the risk of a simultaneous incident elsewhere. One way to ensure reasonable response time to such simultaneous incidents is by *relocating the remaining idle fire trucks* between the fire stations.

In Chapter 2 we address this question, and conduct a case study using FDAA data. We introduce a fast heuristic that allows us to strike a balance between the gain in coverage and the number of relocations made. The heuristic is based on solving a mathematical program that makes use of spatial distribution of incidents, as well as the location and availability of fire trucks. Using ten years of historical data provided by FDAA, we test the algorithm in a simulation against three benchmark policies including the current FDAA practice. The proposed algorithm outperforms all benchmarks, including the heuristic currently used by FDAA. The obtained results demonstrate that significant improvements can be made by making relocations as opposed to not relocating at all, and provide insight into how the system performance can be improved with the willingness to make more relocations.

Dealing with uncertainty in travel times

Another source of uncertainty for firefighting services is the driving time. Traffic jams or weather conditions may negatively affect response times to an incident. In the city center of Amsterdam, for example, where the streets are narrow and crowded, even a single garbage truck or a lifted bridge may cause a delay. *Stochastic driving times* may affect the optimal dispatching policy, especially when multiple vehicles have to be dispatched to the same incident location. The problem gets more complicated when correlation in driving time is present, where two vehicles driving through the same congested area can both get delayed. This brings us to the next question. How to *dispatch fire trucks* to incidents under stochastic driving times and driving time correlation? When dispatching fire trucks to incidents, a common practice of the fire departments is to always dispatch

the closest vehicles. This policy proved to be sub-optimal in some of the previous studies for EMS, even when a single emergency vehicle is dispatched to each incident and when driving times are assumed to be deterministic. The fire departments, however, often have to dispatch multiple fire trucks to the same incident. In the city center of Amsterdam, for example, FDAA always dispatches two trucks to ensure quick response times in case one truck gets delayed.

Chapter 3 models this type of system and provides an algorithm for producing dispatching policies. We model the system as a Markov Decision Process (MDP) that takes into account stochastic driving times and driving time correlation. The optimal policy shows significant improvement over the closest-first dispatching. As the optimal policy cannot be computed for real life applications, we develop a heuristic based on a queuing approximation for the future costs in the MDP formulation. The heuristic is scalable, while able to produce high quality dispatching policies.

Combining dispatching and relocation of resources

In *maintenance networks* for capital goods, when a breakdown occurs, a service engineer has to be dispatched. Dispatching decisions have to be made in real time, and can be affected by the spatial distribution of the machines, as well as by the current location and status of the service engineers. Service engineers, like fire trucks, are spread across the region to ensure quick response times to breakdowns. The question is then how to dispatch these service engineers for the best performance, and whether the performance can be improved by relocating the service engineers between their base stations.

In Chapters 4 and 5 we look into the problem of *dynamic dispatching and relocation of service engineers*. Inspired by the extensive research done for EMS, we propose a number of scalable heuristic policies both for dispatching and relocation of service engineers. The proposed heuristics are compared against each other in a simulation for various types of service logistics networks. The best combination of dispatching and relocation heuristics shows close to optimal performance on a small toy instance, where the optimal policy is obtained using policy iteration algorithm for the MDP formulation of the problem.

Balancing prevention and fast response to emergencies

Spare parts are typically stored in multiple warehouses, and can also be relocated between them. One of the biggest trends in maintenance research is predictive or *condition-based maintenance* that aims at reducing the maintenance costs by repairing machines before they break down. Doing preventive maintenance introduces one more decision layer to an already complicated problem. For instance, assume each breakdown requires a spare part, then how should the spare parts be dispatched to breakdowns, and how they should be relocated between warehouses? With the introduction of condition-based

maintenance we answer yet another question of when and which machines should be preventively repaired to reach the maximum performance of the system?

Chapter 6 introduces a new model that explicitly incorporates the degradation process of the machines in a maintenance service network and enables proactive maintenance along with corrective dispatching and relocation of spare parts. We demonstrate that this significantly improves the optimal policy independent of the cost structure.

Cross-fertilization of ideas from different application areas: In the last paragraph of Section 1.1, we stated that in this thesis we aim to draw inspiration from research done in the area of emergency services to address problems in the area of maintenance services, and vice versa. Indeed, for the results on dispatching fire trucks in Chapter 3, we draw inspiration from the paper by Tiemessen et al. [101], who propose an approximation algorithm for optimal dispatching in the context of spare parts. In Chapter 3, we adapt the approximation in [101] to our problem of dispatching multiple fire trucks. Conversely, we also use insights from the EMS domain for addressing planning problems in the context of maintenance services. More specifically, for the results presented in Chapter 4 we develop a number of heuristics for dispatching service engineers motivated by results in the EMS literature, including compliance tables [108] and the DMEXCLP heuristic proposed by Jagtenberg et al. [47]. Moreover, the results on the ADP-based approach presented in Chapter 5 are based on ideas borrowed from the EMS-domain [71, 79]. This way, the cross-fertilization of ideas from the different application areas form the basis for the results presented in this thesis.

Combining the ideas from the two application areas is possible due to the many similarities in how maintenance and emergency services operate. It is worth mentioning, however, that the two domains have some specific features as well. For example, in firefighter services, we have to deal with the fact that a single incident may require more than one fire truck of the same type. In contrast, in spare parts management for capital goods, it is unlikely that a breakdown of a machine requires multiple spare parts of the same type or more than one service engineer. In turn, in spare parts management, it might be possible and beneficial to postpone a dispatching decision, and wait, for example, for a busy service engineer to become available. It is also possible to perform preventive dispatching/maintenance before an actual breakdown occurs. However, this is not the case in emergency services, where dispatching decisions are always reactive and have to be made as soon as an emergency occurs. Such features need to be taken into account when modelling, and especially when adapting an approach from a different domain.

1.4 Publications of the Author

The results presented in this thesis are based on the following publications:

1. D. Usanov, G.A.G. Legemaate, P.M. van de Ven, and R.D. van der Mei. Fire truck relocation during major incidents. *Naval Research Logistics*, 66(2): 105 — 122, 2019.

2. D. Usanov, P.M. van de Ven, and R.D. van der Mei. Dispatching fire trucks under stochastic driving times. *Computers & Operations Research*, 114, 2020.

3. A. Pechina, D. Usanov, P.M. van de Ven, and R.D. van der Mei. Real-time Dispatching and Relocation of Emergency Service Engineers. *European Journal of Operational Research*, 2019. In revision.

4. D. Usanov, A. Pechina, P.M. van de Ven, and R.D. van der Mei. Approximate Dynamic Programming for Real-time Dispatching and Relocation of Emergency Service Engineers (2019). Manuscript submitted for publication.

5. D. Usanov, P.M. van de Ven, and R.D. van der Mei. Integrating Condition-Based Maintenance into Dynamic Spare Parts Management (2019). Manuscript submitted for publication.

2

Fire Truck Relocation during Major Incidents

The effectiveness of a fire department is largely determined by its ability to respond to incidents in a timely manner. To this end, fire departments typically have fire stations spread evenly across the region, and dispatch the closest truck(s) whenever a new incident occurs. However, large gaps in coverage may arise in the case of a major incident that requires many nearby fire trucks over a long period of time, substantially increasing response times for emergencies that occur subsequently. Motivated by this, we propose a heuristic for relocating idle trucks in order to retain good coverage during a major incident. This is done by solving a mathematical program that takes into account both the location of the available fire trucks and the historic spatial distribution of incidents. This heuristic allows the user to strike a balance between the coverage and the number of truck movements. Using extensive simulations, we test the heuristic for the operations of the Fire Department of Amsterdam-Amstelland (FDAA), and compare it against three other benchmark strategies in a simulation fitted using ten years of historical data. We demonstrate substantial improvement over the current relocation policy, and show that not relocating during major incidents may lead to a significant decrease in performance.

The work in this chapter is based on [103]: D. Usanov, G.A.G. Legemaate, P.M. van de Ven, and R.D. van der Mei. Fire truck relocation during major incidents. *Naval Research Logistics* 66(2): 105 — 122, 2019.

2.1 Introduction

Fire fighting services are designed and operated to minimize the response time to fires and other incidents that require fire department presence. To this end, fire stations are positioned throughout the coverage area of a fire department to allow for a fast response to any incident, irrespective of its location. This coverage may be disrupted by major incidents, such as large fires, which can occupy many nearby trucks over an extended period of time. Consequently, emergencies that arise during a major incident may experience a slower response. To address this issue, it is standard practice of many fire departments to reduce the gap in coverage by temporarily relocating idle fire trucks [34].

A substantial research effort has been devoted to organizing the fire department and other emergency services on the strategic, tactical and operational level, which has succeeded in reducing response time. However, the problem of relocating fire trucks during major incidents has received relatively little attention, and in practice this is done based mainly on the dispatchers' intuition. In Appendix 2.8 we describe (an abstraction of) the relocation heuristic currently used by the FDAA, which covers Amsterdam and its surrounding areas, obtained from discussions with its dispatchers. This heuristic does a single relocation in case of a major incident, moving an idle truck to the now empty fire station closest to the incident. Discussions with the FDAA revealed that, in order for any relocation algorithm to be acceptable in practice, it should be simple to implement and intuitive to explain. Most importantly, the number of relocations done after a major incident should be small, and determined by the dispatchers. The latter constraint is designed to prevent relocations of limited utility, which may cause unnecessary inconvenience to the firefighters.

To our knowledge, the only study that considers relocations during major incidents is [59], in the context of the Fire Department of the City of New York (FDNY). The approach proposed there was adopted by the FDNY, and was for instance successfully used during the terrorist attacks on September 11, 2001, to maintain good coverage throughout the city [34]. While successful in New York, we are not aware of any other fire departments that have implemented this algorithm, at least not in the Netherlands. We conjecture that this is because this approach lacks some of the desired characteristics outlined above. In particular:

(i) the procedure used to calculate cost coefficients for the objective function is complicated and hard to explain to practitioners,

(ii) some of the assumptions made seem specific to the grid-like street network of New York,

(iii) it does not allow the user to control the number of relocations.

In this chapter we propose a relocation heuristic that has all these desired features, and in addition allows us to find better relocations.

We consider the situation where a new major incident has just started, and fire trucks have been dispatched to the incident. We then solve a coverage-maximization problem that takes into account *both* the location of the remaining idle fire trucks *and* the historic spatial distribution of incidents. Our objective function contains a parameter indicating the willingness to relocate, which can be used to control the number of relocations made during a major incident. Moreover, we impose some measure of fairness across the region by ensuring that each location is covered by a certain minimum number of fire trucks. Once the major incident is resolved, the relocated trucks return to their base stations.

To assess the effectiveness of our approach in practice, we apply it to the case of the FDAA, by fitting our model to ten years of incident and dispatch data. We demonstrate a substantial improvement over the current practice, and comfirm the importance of relocations by showing a significant reduction in the response time compared to not doing relocations at all.

The contribution presented in this chapter is three-fold:

1. We introduce a new relocation heuristic which is easy to implement and to explain to practitioners;
2. This heuristic grants the user significant control in terms of the number of relocations made per major incident, allowing him to strike a balance between coverage gain and inconvenience to firefighters caused by additional relocations;
3. Using real-life data, it is tested against three other relocation methods. Our heuristic shows better performance, especially when there are only a few trucks available.

The remainder of this chapter is structured as follows. In Section 2.2 we provide an overview of the relevant literature. The model outline is described in Section 2.3, followed by Section 2.4 where our relocation algorithm is presented. In Section 2.5 we discuss the performance metrics used to evaluate the relocation methods. The simulation and the data used to conduct computational experiments, together with the results of the experiments, are discussed in Section 2.6. In Section 2.7 we conclude and outline future research directions.

2.2 Literature Review

The topic of this chapter falls into the area of organizing emergency service systems, which is usually divided into three levels: *strategic*, *tactical* and *operational*. At the strategic level, facility location problems are solved to determine where to optimally locate the system facilities (e.g., fire stations). At the tactical level, the problem of allocating vehicles (e.g., fire trucks) to the facilities is addressed. Often, the strategic and tactical level problems are solved jointly. In contrast, the operational level concerns short-term decisions, such as how to dispatch vehicles to incidents or how to relocate vehicles between the facilities in real time.

The majority of the research on organizing emergency service systems have been motivated by ambulance management. Reviews of the emergency facility location and ambulance relocation models can be found in [12, 66]. One of the first emergency facility location models is the Location Set Covering Model (LSCM), introduced in [102]. LSCM finds the smallest number and the locations of facilities required to cover every demand point within a certain universal time threshold. The same concept of coverage was used in the Maximal Covering Location Problem (MCLP), formulated in [17]. However, the objective of MCLP is to maximize population covered by a given number of facilities. These two basic models were followed by extensions that incorporated backup or multiple coverage, and partial coverage. Examples of such extensions are the hierarchical objective set covering model [20], backup coverage models [37], maximum availability location problem [91], double standard model (DSM) [29], and MCLP with partial coverage [54]. In [19], the Maximum Expected Covering Location Problem (MEXCLP) was introduced, a probabilistic extension of MCLP. The MEXCLP model uses the concept of marginal coverage accounting for the probability that facilities may be busy responding to incidents. The MEXCLP model was further followed by extensions incorporating stochastic travel times [46, 114], time-dependent demand [113], and survival probabilities [26, 57].

One of the first models for facility location in a fire department context was introduced in [38], where the authors proposed a greedy heuristic for determining the locations of the fire stations. Since then, various studies have looked at formulating and solving mathematical programs for fire department related coverage problems. Such studies include [87], where the authors used a hierarchical objective function for the set-covering problem in a case study for the Denver Fire Department. In [92], MCLP and a multi-objective formulation were applied to the city of Baltimore. A multi-objective model was also used in [4]. Recent fire department-specific facility location case studies include [16], [21] and [115].

At the operational level, we limit ourselves to discussing literature related to relo-

cations. The locations of the emergency facilities are assumed to be given, of interest is the decision how to relocate vehicles between those facilities in real time. The first problem of such type was addressed in [59] in the early 1970s. The authors introduced the mathematical programming formulation and a heuristic for relocating idle trucks during a major incident. The problem of dynamic ambulance relocation was first discussed in [9], where the authors used dynamic programming to find an optimal solution.

The basic concepts and models developed to solve the strategic and tactical level problems were further used to develop relocation models at the operational level for Emergency Medical Services (EMS). Such models include the dynamic extensions of DSM [30] and MEXCLP [31, 108]. Additionally, recent approaches addressed the problem using heuristics [47, 110], approximate dynamic programming [71, 93], stochastic optimization [78], and Markov chains [1].

It is worth noting that insights and heuristics obtained for EMS cannot directly be applied to the fire department setting. One of the main reasons is that fire departments usually experience much lower incident rates than EMS, and consequently, the fraction of time each truck is busy responding to incidents is relatively small. This allows the use of one-shot decision formulations instead of multiple-step or infinite horizon. Moreover, EMS models are often driven by regulatory requirements that are uniform across the coverage area. Fire departments, however, may impose different time thresholds for different buildings depending on their function and location. Another distinguishing feature of the fire departments' operations is that often multiple trucks are required for one incident.

2.3 Model

We consider a region partitioned into a set of demand locations $\mathcal{L}$, and assume that new incidents start at each demand location $l \in \mathcal{L}$ according to a Poisson process with rate λ_l. Poisson arrivals are common in the research literature on emergency service operations, where the times between successive events are memoryless and independent. The rates at which new incidents occur may differ between demand locations due to, for instance, population density and building types.

The region is served by a set of fire stations $\mathcal{N}$. Denote by $g(i) \in \mathcal{L}$ the demand location that station $i \in \mathcal{N}$ is located in. In practice, the fire department uses a range of vehicles, including pumpers, ladder trucks and trucks specialized in roadside accidents. A particular incident may require one specific truck type, or a mix. To simplify the analysis, we limit ourselves to a single type of fire truck that is dispatched to all incidents. All

		Vehicle Type			
Priority	**Incidents**	**Pumper**	**Ladder**	**Rescue**	**Marine Rescue**
1	88879	99%	28%	3%	3%
2	28432	95%	30%	1%	2%
3	10085	87%	20%	2%	2%
Total	**127396**	**97%**	**28%**	**3%**	**2%**

Table 2.1: Deployment per vehicle type grouped by priority (data: FDAA 2008 - 2018)

results, however, generalize easily to the case with multiple types of vehicles, as discussed in Section 2.4.1. This assumption is motivated by the example of FDAA, where a pumper is dispatched to almost every high-priority incident. The FDAA fleet usage statistics are summarized by vehicle type in Table 2.1. It shows the number of incidents that occurred over a 10-year period, and for each vehicle type and incident priority level (priority 1 being the highest) the percentage of incidents of that priority that required at least one truck of that type. From this table it is clear that pumpers are dispatched to almost every incident. Each fire truck has a base station where it is located when not handling an incident or temporarily relocated to another station. We assume that each fire station is the base station for at least one truck.

The travel time t_{lm} between each pair of demand locations $l, m \in \mathcal{L}$ is assumed to be deterministic and known. The time it takes for a truck at station i to travel to another fire station j ($j \neq i$) or an incident location l is equal to the travel time between the corresponding demand locations (i.e., $t_{g(i)g(j)}$ and $t_{g(i)l}$, respectively). Let q_i be the (deterministic) dispatch time corresponding to station $i \in \mathcal{N}$, i.e., the time it takes for a truck to leave its base station i after an incident started. We define the response time of a fire truck from station $i \in \mathcal{N}$ to an incident at demand location $l \in \mathcal{L}$ as $r_{g(i)l} := q_i + t_{g(i)l}$. Because both the travel times and dispatch times are assumed to be deterministic, so are the response times.

We denote by $i^{(k)}(l) \in \mathcal{N}$ the kth closest fire station to demand location l measured in terms of response time, $k = 1, \ldots, |\mathcal{N}|$. We define the service area SA_i of a fire station $i \in \mathcal{N}$ as the set of demand locations to which this fire station is closest in terms of response time, i.e., $SA_i = \{l \in \mathcal{L} |\ i = i^{(1)}(l)\}$. We assume that for every demand location $l \in \mathcal{L}$ there are no two stations i and j ($i \neq j$) such that $r_{g(i)l} = r_{g(j)l}$, that is, the corresponding response times are different for all stations. Let $d_i = \sum_{l \in SA_i} \lambda_l$ be the total demand corresponding to the service area of station i.

The number of trucks required for a new incident is a random variable S, and assumed

to be independent of other incidents and identically distributed within the same service area. When a new incident arises, all required trucks are dispatched simultaneously. In case there are insufficient idle trucks, the remainder will be provided by neighbouring fire departments. Whenever a new incident arises, those idle fire trucks with the smallest response time for the corresponding demand location are dispatched. After the incident is resolved, all trucks return to their base station. Since we only consider incidents of the highest priority, this is in accordance with the current dispatching policy of FDAA (and fire departments elsewhere).

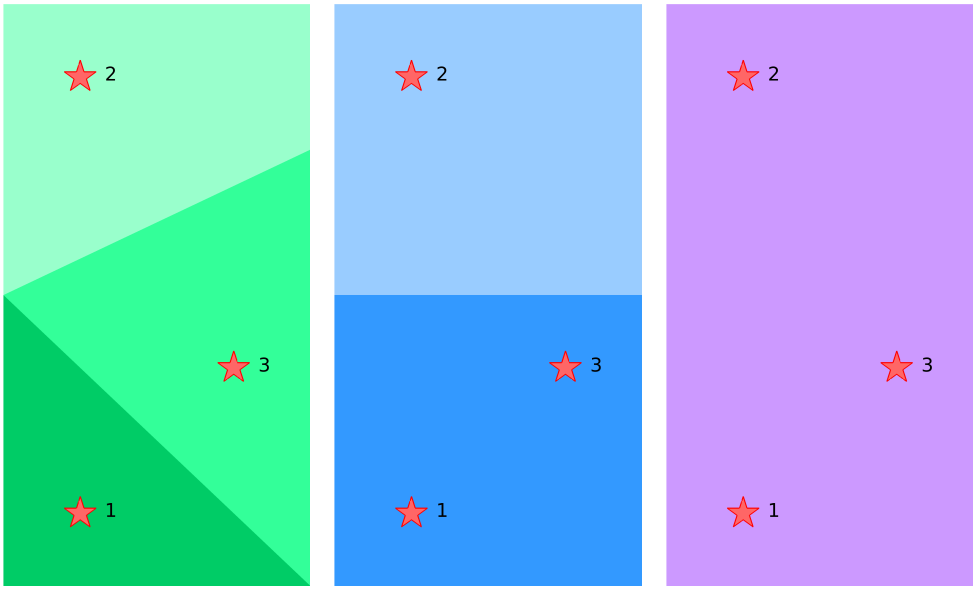
2.3.1 Response Neighborhoods

The relocation heuristic that we present in Section 2.4 will strive to relocate trucks to improve coverage, i.e., position the idle trucks to maximize the probability that the next incident is responded to in time. However, in the fire fighting domain fairness is an important secondary criterion, as we want to avoid neglecting certain areas. For instance, *not* covering rural areas because this is not optimal from a coverage perspective may not be acceptable for a fire department, as all fires should be responded to within certain time limits. Hence, assuming that the fire department considers its original allocation of trucks to be fair, we try to maintain that relative distribution of trucks across the region when relocating.

In order to measure fairness, we use the concept of a response neighborhood (RN) of a set of fire stations $N \subseteq \mathcal{N}$, defined as the set of all demand locations for which the fire stations in N are the $|N|$ closest, i.e., $\text{RN}(N) = \{l \in \mathcal{L} \mid N = \{i^{(1)}(l), \ldots, i^{(|N|)}(l)\}\}$. If N contains a single station (i.e., $|N| = 1$), its response neighborhood corresponds to the service area of that station (i.e., $\text{RN}(\{i\}) = SA_i$). If N contains all stations (i.e., $N = \mathcal{N}$), its response neighborhood is simply the collection of all demand locations (i.e., $\text{RN}(\mathcal{N}) = \mathcal{L}$). Note that the response neighborhood of a set of fire stations may be empty, for instance if those stations are located on opposite sides of the service region.

We are particularly interested in the collection of response neighborhoods corresponding to all sets of fire stations of equal size $n \in \{1, ..., |\mathcal{N}|\}$. We denote this collection by $\mathcal{K}_n = \{\text{RN}(N) \mid |N| = n\}$, and observe that for each n, $\mathcal{K}_n$ forms a partition of the set of all demand locations $\mathcal{L}$. We say that a *fire station i serves response neighborhood $k \in \mathcal{K}_n$* if it is one of the n closest stations for that response neighborhood.

The partitioning of demand locations is illustrated in Figure 2.1, which visualizes $\mathcal{K}_n$ in a toy example with three fire stations. Every point of the rectangular region in Figure 2.1 is considered as a separate demand location, and Euclidean distance is used to



(a) $n = 1$ (b) $n = 2$ (c) $n = 3$

Figure 2.1: Representation of the response neighborhoods $\mathcal{K}_n$ for $n = 1,\ 2,\ 3$

determine the response time. Points belonging to the same response neighborhood have the same color. For $n = 2$ (Figure 2.1b), for example, the region is partitioned into two response neighborhoods: the light blue is served by fire stations 2 and 3, and the dark blue is served by stations 1 and 3. In this case there are no points with 1 and 2 as their closest stations, so $\text{RN}(\{1, 2\}) = \emptyset$.

To store the relation between fire stations $\mathcal{N}$ and response neighborhoods $\mathcal{K}_n$, we use an $|\mathcal{N}|$-by-$|\mathcal{K}_n|$ incidence matrix A_n, with an element $a^n_{ik} = 1$ if the fire station $i \in \mathcal{N}$ serves the response neighborhood $k \in \mathcal{K}_n$, and $a^n_{ik} = 0$ otherwise. One fire station can serve several response neighborhoods of size n, and one response neighborhood of size n is served by exactly n fire stations. We say that a response neighborhood k is *covered* if at least one of the fire station serving this response neighborhood has a truck ready to respond to an incident.

The notion of response neighborhoods was originally introduced in [59] for $n = 3$, and in this chapter we extend it to general n, to allow us to address the feasibility issues discussed in Section 2.4.1 below.

2.4 Relocation Algorithm

We consider the moment when a new incident occurs and the required trucks are dispatched, and are interested in how to relocate the remaining idle fire trucks between stations to compensate for the temporary loss of coverage. For ease of presentation and implementation, we decompose this problem into two parts, with no loss in performance. In Section 2.4.1 we introduce an integer program that identifies (1) a set of trucks to be relocated and (2) a set of empty stations to be filled with those trucks. In Section 2.4.2 we apply the Linear Bottleneck Assignment Problem (LBAP) [13] to determine which of those trucks should be relocated to which stations. The relocation algorithm uses these two formulations, and is summarized in pseudocode in Appendix 2.8.

To provide an even coverage of the region, we require that each response neighborhood $k \in \mathcal{K}_n$, for some fixed value of n, is covered by at least one truck. In other words, for every demand location, at least one of the n closest fire stations should have a fire truck available. In practice, the appropriate value of n is decided upon by the fire department. If an incident involving at least n trucks happens, some response neighborhoods in $\mathcal{K}_n$ may become uncovered, and some trucks have to be relocated to satisfy the requirement.

The choice of n influences how frequently relocations will be made, and how uniformly trucks are redistributed over the region when making relocations. For lower n, we will have to make relocations more frequently, since the response neighborhoods of smaller size lose coverage more often. However, the distribution of trucks over the region will be more even when smaller response neighborhoods are covered.

Not every incident should necessarily lead to making relocations, as the coverage may still remain sufficient, or the coverage loss may be for a short period of time. The condition that triggers the relocation algorithm can be anything, such as uncovering a response neighborhood, or the number of idle trucks falling below some threshold. In the numerical evaluation in Section 2.6, we run the relocation algorithm whenever three or more trucks are dispatched in a single major incident.

2.4.1 Maximum Coverage Relocation Problem

We now introduce some additional notation, in order to formulate the decision which trucks to relocate as a mathematical program. Let f_i be the number of trucks available at a station i right after a major incident occurred and the required trucks are dispatched to it. We also introduce three sets of fire stations: the set of empty stations $\mathcal{E} = \{i \in \mathcal{N} : f_i = 0\}$, stations with exactly one available truck $\mathcal{S} = \{i \in \mathcal{N} : f_i = 1\}$, and stations with more than one available truck $\mathcal{M} = \{i \in \mathcal{N} : f_i \geq 2\}$. Finally, we use the

following three sets of variables. The variable x_{ij} is equal to 1 if we decide to relocate a truck from station i to station j, and 0 otherwise. The binary variable z_i is equal to 1 if station i has no trucks available after all the relocations are made, and 0 otherwise. The variable y_i is equal to the number of trucks at station i after all relocations are completed.

The objective that we want to optimize is a combination of (i) the gain in coverage obtained from relocation, and (ii) some penalty for making too many relocations. The former consists of multiple terms, depending on whether the relocated trucks came from stations with multiple trucks or not. If not, the net gain in coverage can be written as

$$\sum_{i\in\mathcal{S}}\sum_{j\in\mathcal{E}} x_{ij}(d_j - d_i),$$

and the gain for the cases with multiple trucks present is represented as

$$\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{E}} x_{ij}d_j - \sum_{i\in\mathcal{M}} z_i d_i.$$

The penalty for relocation is simply given by the total number of relocations made, $\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{E}} x_{ij}$. Combining these using some weight parameter $W \in [0,1]$, we obtain the objective function (2.1) below.

The parameter W serves two purposes. First, when chosen correctly it ensures that both components of the objective function have the same order of magnitude. Second, it indicates the willingness to relocate. If $W = 0$, the smaller number of relocations is made to satisfy the constraints. If $W = 1$, the gain in demand covered is maximized independently of the number of relocations made. The value of W can be set by the user of the relocation heuristic. The relevant range of parameter W depends on the data, as the order of magnitude of the gain in coverage (the first term of the objective (2.1), see below) depends on the fire department's policy. Specifically, it is affected by the locations of fire stations, the allocation of trucks, and on the frequency and spatial distribution of the incidents. If W is too large (close to 1), too many relocations are made. Conversely, if W is close to 0 then coverage is ignored completely, and an arbitrary feasible solution (satisfying (2.2), see below) is chosen. For instance, in our case, $W = 0.01$ was sufficient to ensure that the number of relocations made does not exceed the minimum required by the constraints (2.2) while resulting in substantial coverage gains. However, the choice of W also depends on the user's willingness to relocate fire trucks. In Section 2.6.3 below we show how the system performance can be improved by increasing W and allowing users to make additional relocations.

As mentioned above, in addition to maximizing coverage, we also aim for fairness, by ensuring that all response neighborhoods are covered after relocations are finished. To do

this, we impose constraint (2.2) below. Combining the objective function and this fairness constraint, we are in position to provide the Maximum Coverage Relocation Problem (MCRP) formulation:

$$\max\ W\Big(\sum_{i\in\mathcal{S}}\sum_{j\in\mathcal{E}} x_{ij}(d_j-d_i)+\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{E}} x_{ij}d_j-\sum_{i\in\mathcal{M}} z_i d_i\Big) - \big(1-W\big)\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{E}} x_{ij} \tag{2.1}$$

$$\text{s.t.}\ \sum_{i\in\mathcal{N}} a^n_{ik} y_i \geq 1, \qquad k\in\mathcal{K}_n \tag{2.2}$$

$$\sum_{j\in\mathcal{N}} x_{ij} \leq f_i, \qquad i\in\mathcal{N} \tag{2.3}$$

$$\sum_{j\in\mathcal{N}} x_{ji} \leq 1, \qquad i\in\mathcal{E} \tag{2.4}$$

$$1-z_i \leq y_i, \qquad i\in\mathcal{M} \tag{2.5}$$

$$y_i = f_i+\sum_{j\in\mathcal{N}} x_{ji}-\sum_{j\in\mathcal{N}} x_{ij}, \qquad i\in\mathcal{N} \tag{2.6}$$

$$x_{ij}=0, \qquad i\in\mathcal{N},\ j\in\mathcal{S}\cup\mathcal{M} \tag{2.7}$$

$$x_{ij}, z_i \in\{0,1\}, \qquad i,j\in\mathcal{N} \tag{2.8}$$

$$y_i\in\{0,1,...\}, \qquad i\in\mathcal{N} \tag{2.9}$$

Here, constraints (2.3) do not allow to relocate more trucks than available at a station. At most one truck is relocated to the same empty station due to (2.4). Constraints (2.5) force the decision variable z_i to take value 1 if station i becomes uncovered in a given solution. Constraints (2.6) ensure that the variables y_i have the correct values. Finally, (2.7) makes sure that relocations are made only to empty stations.

Fire departments typically have very strict rules about what vehicles are dispatched to what types of incidents (in particular for high priority incidents). Specifically, FDAA uses a dispatching policy where for each type of incident it is predefined how many trucks of each type are needed, and the vehicles of different types are typically not mutually substitutable. Hence, the model can be easily applied to multiple types of trucks by decomposing the problem into different vehicle types. In this case response neighborhoods, coverage requirements and the objective coefficients are defined for each vehicle type separately. The same formulation can then be used with different input data to find

optimal relocations for each type of trucks independently of other types.

Note that different fire departments may have policies or rules that impose additional constraints which can be easily included in our model. In the case of FDAA, for example, fire stations are of two types: *professional* and *volunteer*. Trucks from volunteer fire stations are not allowed to relocate. We can take this into account by adding the following constraint to the MCRP formulation:

$$y_i \geq v_i \quad \forall i \in \mathcal{N},$$

where v_i is the number of volunteer trucks at station i before making relocations. In our numerical evaluation in Section 2.6 we will include this constraint as well.

Remark 1 (MCRP Feasibility) It may be infeasible to satisfy the MCRP constraints (2.2) for a given value of n if the number of available idle trucks is too small to cover all response neighborhoods in $\mathcal{K}_n$. A similar set of constraints to ours was used in [59] with the definition of response neighborhood implying a fixed size of it. The authors of [59] admit that there may be no feasible solution to their problem, and that the fire department in this case uses some emergency allocation procedures. To handle this problem we introduce the starting response neighborhoods' size $n_0 \in \mathbb{N}$, and initially try to solve MCRP with $n = n_0$. If the problem is infeasible, we set $n = n_0 + 1$, and solve MCRP again. We continue incrementing n by 1 until the problem is feasible. As the size n of response neighborhoods increases, fewer trucks are needed to satisfy constraint (2.2). Assuming that there is at least one idle truck available, the problem is always feasible with $n = |\mathcal{N}|$, as there is only one response neighborhood in $\mathcal{K}_{|\mathcal{N}|}$.

Remark 2 (MCRP Generalization) In the formulation (2.1)-(2.9) we partition the region into response neighborhoods of the same size n to ensure that each demand location has at least one idle truck at one of the n closest fire stations. This approach appeals to FDAA as it provides fairness across the region, independent of the arrival rates of new incidents. If needed, by increasing the W parameter additional relocations can be made so that the busier response neighborhoods are covered by more trucks if the number of idle trucks exceeds the minimum required to satisfy constraint (2.2). Although this definition of fairness was requested by FDAA, other fire departments may have different constraints. For example, one could require for one set of demand locations to have at least one idle truck at one of the two closest stations, and for another set to have at least two trucks at one of the five closest stations. To allow for this, in Appendix 2.8 we provide a generalized formulation of MCRP that can incorporate more complicated response neighborhood structures and their coverage requirements.

2.4.2 Linear Bottleneck Assignment Problem

In general, there may be multiple optimal solutions to MCRP that would relocate the same set of trucks to the same set of empty stations. For instance, assume that the MCRP model proposes to relocate one truck from station 1 to station 2, and another truck from station 3 to station 4. For the MCRP model this solution is equivalent to the one where we relocate a truck from station 1 to station 4, and another truck from station 2 to station 3. However, in practice, because of differences in travelling time between the stations, these two solutions can differ in terms of the time it takes to realize them.

To maintain good coverage levels in real-time, we want to move to a new configuration of trucks as fast as possible. A similar task for ambulance relocation was addressed in [108] using the Linear Bottleneck Assignment Problem (LBAP), that can be solved in polynomial time [13]. We formulate LBAP in the context of fire truck relocation. Let x_{ij} for $i, j \in \mathcal{N}$ be the solution of MCRP. Next, we construct the set of origin fire stations O and the set of destination fire stations D as follows. For every pair (i, j) such that $x_{ij} = 1$, we add station i into the set of origins O, and we add station j into the set of destinations D. There can be more than one truck relocated from the same station i elsewhere. In this case, we add station i to the set O as a separate element for each truck relocation from this station. Hence, multiple origins $o \in O$ may correspond to the same fire station. Due to constraints (2.4), it is never optimal in MCRP to relocate more than one truck to the same station j, so each of the destination stations appears in the set D only once. The obtained sets O and D are of the same size, containing origins and destinations for all the trucks that have to be relocated. Let the binary decision variable $\hat{x}_{od}$ be equal to 1 if a truck should be relocated from station $o \in O$ to station $d \in D$, and 0 otherwise. The problem of minimizing the maximum traveling time over all relocations can then be formulated as follows:

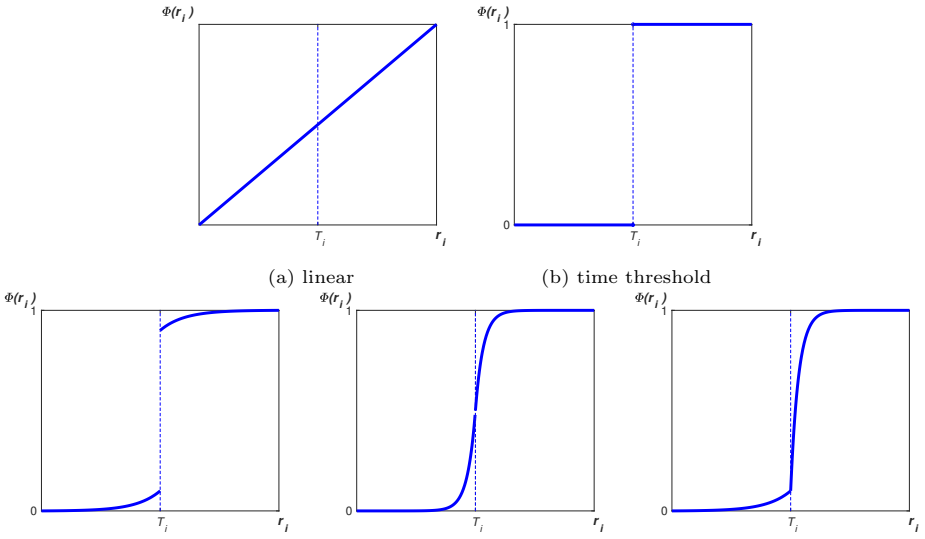
$$\min \max_{o \in O,\ d \in D} t_{g(o)g(d)} \hat{x}_{od} \tag{2.10}$$

$$\text{s.t.} \sum_{d \in D} \hat{x}_{od} = 1, \qquad o \in O \tag{2.11}$$

$$\sum_{o \in O} \hat{x}_{od} = 1, \qquad d \in D \tag{2.12}$$

$$\hat{x}_{od} \in \{0, 1\}, \qquad o \in O, d \in D \tag{2.13}$$

Here, we use the function g introduced in Section 2.3 to indicate the demand locations corresponding to the elements of O and D. Constraints (2.11) and (2.12) ensure that exactly one truck is relocated from each origin $o \in O$, and exactly one truck is assigned



(a) linear

(b) time threshold

(c) $a = 0.1$, $b = 0.1$, $\alpha = 5$, $\beta = 5$

(d) $a = 0.5$, $b = 0.5$, $\alpha = 15$, $\beta = 15$

(e) $a = 0.1$, $b = 0.9$, $\alpha = 5$, $\beta = 15$

Figure 2.2: Examples of penalty functions $\Phi(r_i)$

to each destination $d \in D$, respectively.

2.5 Performance Metrics

There are many ways of measuring the performance of an emergency service system. In this section we present some of the main performance metrics used by practitioners and researchers. Assume that we have a set of incidents $\mathcal{I}$. Let r_i denote the response time for incident $i \in \mathcal{I}$. The performance metrics we consider are of the form

$$\sum_{i=1}^{|\mathcal{I}|} \Phi(r_i)/|\mathcal{I}|,$$

where $\Phi(.)$ is a non-decreasing one-dimensional penalty function. So we consider the average penalty over incidents in $\mathcal{I}$.

One of the most commonly used penalty functions, shown in Figure 2.2a, is a linear penalty function:

$$\Phi(r_i) = r_i, \ i \in \mathcal{I}, \tag{2.14}$$

which represents the response time. The disadvantage of this performance measure is that even if the overall average response time is low, there can be a lot of variability in response time for particular incidents.

Alternatively, the fire department can use time thresholds, indicating how soon incidents should be responded to since the moment of an alarm. It can be a single time threshold for the whole region, or different time thresholds for different demand locations. Assume T_i is the time threshold corresponding to the ith incident's location. The penalty function displayed in Figure 2.2b corresponds to the 'fraction of late arrivals' performance metric, and is defined as follows:

$$\Phi(r_i) = \begin{cases} 0, & \text{if } r_i \leq T_i, \\ 1, & \text{if } r_i > T_i. \end{cases} \tag{2.15}$$

However, the disadvantage is that this function gives the same penalty no matter how much the response time exceeds the time threshold. So, once the response time threshold has been exceeded, further delays will not be penalized.

The final penalty function we consider is a combination of the first two:

$$\Phi(r_i) = \begin{cases} a\frac{e^{\alpha r_i/T_i}-1}{e^{\alpha}-1}, & \text{if } r_i \leq T_i, \\ 1 - b\frac{e^{\beta(2T_i-r_i)/T_i}-1}{e^{\beta}-1}, & \text{if } r_i > T_i. \end{cases} \tag{2.16}$$

Here, parameters a, b, α and β allow us to adjust the shape of the function, providing flexibility in the system performance evaluation. The parameters a and b define the points where the two functions comprising $\Phi(.)$ intersect the time threshold T_i, and the parameters α and β define the steepness of those functions. Examples of the resulting penalty function for different parameter values are presented on Figures 2.2c, 2.2d and 2.2e.

2.6 Numerical Experiments

In this section we evaluate the performance of our relocation algorithm by applying it to incident data from FDAA. In our computational experiments we compare it to the following three benchmarks:

1. **The Kolesar-Walker benchmark.** This benchmark uses our algorithm (Appendix 2.8) with the MCRP formulation substituted with the adapted version of the mathematical program proposed by Kolesar and Walker [59]. We refer to this relocation strategy as **KW**. The adapted formulation can be found in Appendix

2.8, where we discuss in detail how to implement this formulation, and highlight several implementation issues that may arise.

2. **The Current Practice benchmark.** This benchmark is the relocation algorithm used in current practice by FDAA. This algorithm was originally developed between 1994 and 1996, and different dispatchers use their own interpretation of it during deployment, based on their experiences and intuition. A detailed description of this algorithm can be found in Appendix 2.8. We refer to this relocation strategy as **CP**.

3. **The No Relocations benchmark.** This benchmark makes no relocations, and referred to as **NR**.

In [59], the authors note that the integer programming formulation used in the KW heuristic cannot be solved exactly in a reasonable amount of time. They, therefore, decompose the problem in two stages and solve it heuristically. However, today computational time is no longer an issue for solving both MCRP and KW integer programs exactly, for realistic problem sizes. In our computational experiments, we used Gurobi MIP solver [35] that was able to find exact solutions in a matter of seconds.

2.6.1 Simulation

We simulate the FDAA operations to assess the performance of the four strategies. In this section, we briefly describe how the simulation works.

We generate the sequence of incidents over a given time horizon. Each incident i has four attributes: *time* t_i, *location* l_i, *size* S_i in terms of the number of trucks involved, and *duration* D_i. The duration of an incident is defined as the time between the arrival of the first truck to the location of the incident, and the end of the incident. We then process the sequence of incidents using one of the four relocation strategies.

In each demand location l, new incidents arrive with rate λ_l. Given the demand location of a new incident l_i, we also know the corresponding service area. The service area is further used to sample the random size of an incident S_i. The size of an incident is independent of other incidents and identically distributed for the same service area. It is drawn from an empirical distribution based on data for the corresponding service area. For the duration of an incident D_i we use a Weibull distribution, where the parameters are fitted to the data corresponding to the service area and the size of an incident. As there are less data available for large incidents, we group these and use the same parameters for all major incidents in the same service area. In order to arrive at realistic values for the duration, this distribution is truncated between 0.1 and 24 hours. We choose the

Weibull distribution, because it has positive support, can represent both heavy and light tailed behavior, and allows us to accurately fit the data. Weibul distribution was used, for example, for hospital treatment time in EMS literature [71, 110, 111].

When we process the sequence of incidents, the trucks are dispatched to incidents according to their mean response time for a given demand location. The dispatch and travelling times are assumed to be deterministic and known. Each truck can be in one of the two states. It is either 'busy' with an incident or 'available' to be dispatched. When a truck is dispatched to an incident, its state changes to 'busy'. The state of a truck is switched to 'available' again immediately after the incident is finished, and the truck starts travelling to the fire station it was assigned to. We do not track the exact location of fire trucks. We only track their 'destination' fire stations. So, when dispatching a truck that is relocating or returning from another incident, we assume it to be dispatched from its 'destination' fire station.

Whenever a major incident occurs, we consider relocating trucks using one of the four relocation strategies mentioned earlier. If the truck is relocated, its 'destination' changes to the fire station it is relocated to. The state of such truck remains 'available'. The relocated truck goes back (changes its 'destination') to its base station whenever another 'available' truck is assigned to the station the first truck was relocated to.

2.6.2 Data

To estimate the input parameters of the simulation, we use the real-world data from the FDAA. This fire department currently operates 22 pumpers located at 19 fire stations, and covers 6 municipalities with total population of approximately one million inhabitants. In our simulation, we omit one volunteer station with one pumper that does not have its own service area. There are several professional fire stations close to it, so the truck from this station is never the fastest to any incident because of the relatively large dispatch time associated with volunteer firefighters.

We use the partitioning of the region into 2663 demand locations defined and used by the FDAA. Those demand locations are the polygons comprising the region in Figure 2.3. The FDAA also provided us with the average travelling times between each pair of demand locations. In addition, we received information on all the incidents that occurred in the FDAA coverage area over the 10-year period 2006-2015. This information includes for each incident its location, starting and end times, and the trucks used to handle the incident. For each truck, we know the time it took to dispatch to an incident location from the moment of an alarm, the travelling time between the fire station and the incident location, the time spent at the scene, and the time it took to return to the fire station.

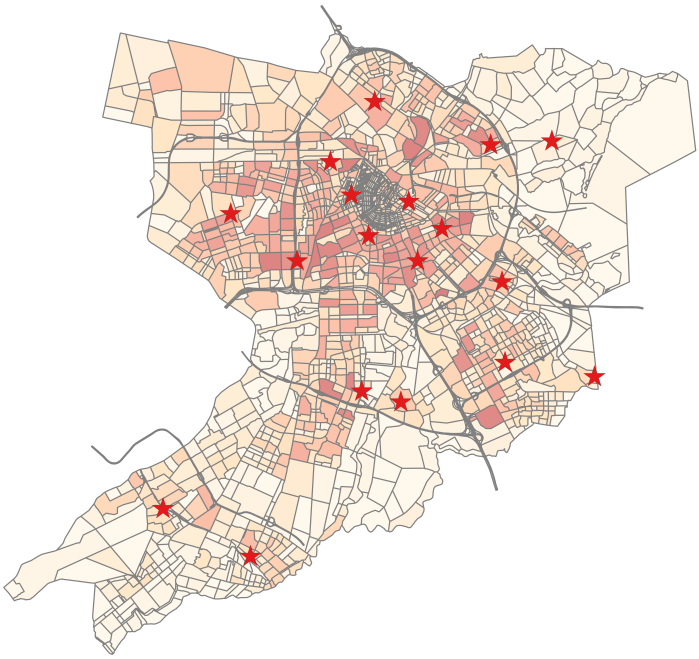


Figure 2.3: Spatial distribution of incidents in Amsterdam-Amstelland

The incidents are distinguished into *three priority levels*. Priority 1 incidents are the most important and constitute the majority of all incidents. The trucks busy with either a priority 2 or a priority 3 incident can be dispatched to a priority 1 incident upon request. To evaluate the arrival rates, we use only the data on priority 1 incidents to which at least one fire truck was dispatched. Figure 2.3 represents the spatial distribution of incidents, with darker demand locations corresponding to higher arrival rates. The overall arrival rate is 21.28 incidents per day. The average duration of an incident is 1.16 hours, and the average number of available trucks upon an incident arrival is 19.6 out of 21. So on average the trucks are idle most of the time. In fact, an average fire truck is busy responding to priority 1 incidents only about 3.5% of the time.

The FDAA uses four different time thresholds T, depending on the type of the building where an incident happened: 5, 6, 8 or 10 minutes. For every demand location $l \in \mathcal{L}$ we know the number n_{lT} of buildings with the corresponding time threshold equal to T. To

get a single time threshold T_l for every demand location $l \in \mathcal{L}$ we compute a weighted average as follows: $T_l = (5n_{l5} + 6n_{l6} + 8n_{l8} + 10n_{l10})/(n_{l5} + n_{l6} + n_{l8} + n_{l10})$. These time thresholds are used to calculate performance measures below.

2.6.3 Computational Results

In this section, we present the results of the experiments conducted using the FDAA data and the simulation. Both MCRP and KW formulations were solved using Gurobi Optimizer [35].

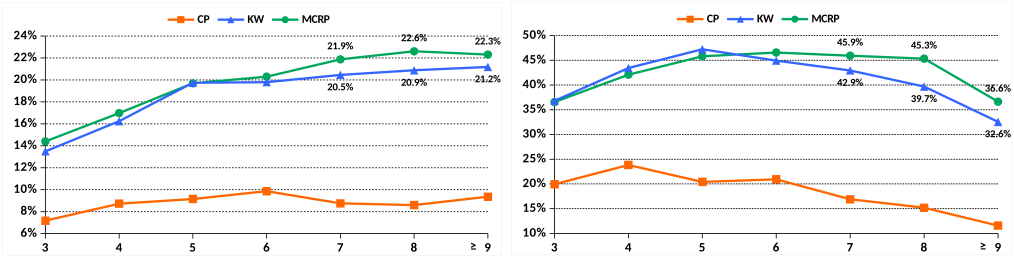
Aggregate performance

First, we run the simulation of FDAA over a time horizon of 200 years with the starting RN size n_0 equal to 3 and parameter $W = 0.01$, that is sufficiently small to make the smallest number of relocations. We do not set $W = 0$, since in this case the model finds an arbitrary solution with the smallest number of relocations neglecting the secondary objective. We use the same sequence of incidents for all four relocation strategies. To compute the performance of the system, we keep track of the response times for all the incidents. Then we limit ourselves to those incidents such that at least one of the four relocation strategies results in a different response time from the others. This is done in order to isolate those incidents that are affected by the coverage gap left by the major incident and the relocation decision made by one of the algorithms. We call these incidents the *decisive subset* of all incidents. The performance metrics are calculated using this decisive subset. In our experiments, the decisive subset constitutes 33.3% of all incidents that occurred simultaneously with a major incident. And the incidents that happen simultaneously with a major incident constitute 3.4% of all incidents.

We use the following notation to refer to the performance measures. ART for the average response time (2.14) and $FLAR_T$ for the fraction of late arrivals (2.15) given a single time threshold T. Using different time thresholds T_l for different demand locations $l \in \mathcal{L}$, we also compute $FLAR$ and the three versions of the compromise penalty function (2.16) CPF_c, CPF_d and CPF_e from Figures 2.2c, 2.2d and 2.2e, respectively. For the time threshold T we choose the four values used by FDAA, being 5, 6, 8 and 10 minutes. These performance metrics, computed over the decisive set of incidents, are presented in Table 2.2. The results show that the MCRP model outperforms all other approaches, and making no relocations is the worst strategy. Improvement made by MCRP over the NR scenario is 19.2% in terms of ART, 42.6% in terms of $FLAR$, and 15.3% to 57.2% in terms of $FLAR_T$. The KW model performs quite close to MCRP, with the biggest difference observed in terms of $FLAR_T$ for time threshold T equal to 5 and 6.

Table 2.2: Aggregate results computed over 200 years simulation run

Performance Metric	**MCRP**	**KW**	**CP**	**NR**
$ART\ (sec)$	413	417	466	511
$FLAR_5$	75.1%	77.2%	84.7%	88.7%
$FLAR_6$	56.8%	58.3%	71.2%	79.4%
$FLAR_8$	29.7%	29.9%	42.4%	53.3%
$FLAR_{10}$	12.6%	12.8%	20.9%	29.4%
$FLAR$	32.2%	32.7%	45.3%	56.1%
CPF_c	0.330	0.335	0.457	0.561
CPF_d	0.330	0.335	0.458	0.562
CPF_e	0.298	0.300	0.418	0.520



(a) ART

(b) FLAR

Figure 2.4: Performance as a function of the number of busy fire trucks

Impact of the number of busy trucks

Table 2.2 compares the four scenarios over all incidents that occur when there are at least three trucks already busy. Next, we break down the same decisive subset of incidents by the number of trucks already busy upon arrival of an incident. Figure 2.4 shows relative improvement over the NR relocation strategy as a function of the number of trucks occupied elsewhere.

We can see that the KW and MCRP models perform approximately the same until the number of busy trucks reaches 7. If 7 trucks or more are already occupied, MCRP significantly outperforms KW. The reason is in the objective function of KW. Each cost coefficient in the KW-model objective is an estimate of the average response time during the major incident if the corresponding relocation is made (see Appendix 2.8). The average response time depends on the configuration of all the trucks, and therefore, on all the relocations made. Hence, the effect of every single relocation depends on whether

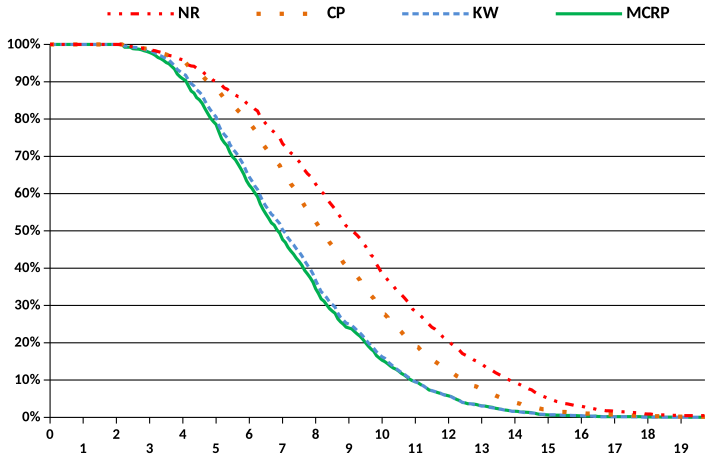


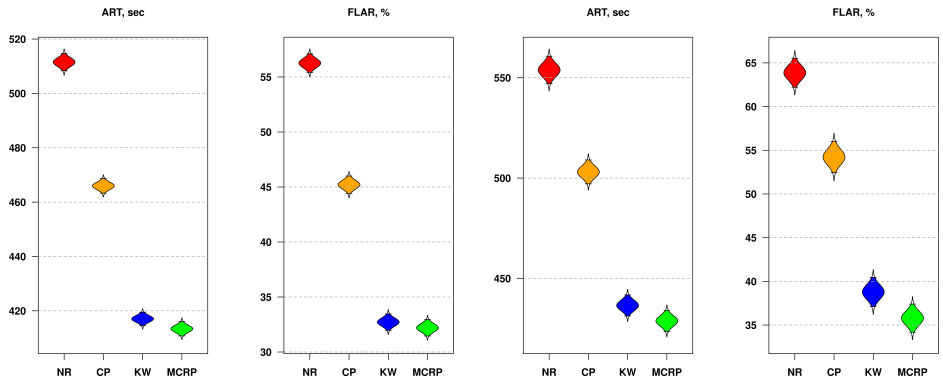
Figure 2.5: $FLAR_T$ plotted as a function of time threshold T

and how other trucks are relocated. This dependency is not taken into account in the KW objective. Hence, the more relocations we make, the less accurate the estimates are. When bigger incidents happen, we have to relocate more trucks to satisfy the coverage constraints, and this increases inaccuracy of the objective of KW.

For the subset of incidents occurring when there are at least 7 trucks busy, Figure 2.5 plots the $FLAR_T$ performance measure as a function of time threshold T, ranging from 0 to 20 minutes with the step of 5 seconds. The MCRP and KW lines are significantly below the other two methods. They are close to each other, but MCRP is consistently better for the time thresholds between 3 and 10 minutes. For T between 7 and 9 minutes, $FLAR_T$ is at least 5% better with the MCRP model than the corresponding value with the KW model.

Confidence intervals

Next, we split the 200 years incidents sequence into 400 intervals of 6 months length. We compute the ART and $FLAR$ performance measures over each interval for every scenario, and calculate the 95% confidence intervals for the obtained values. We do this first for the incidents that occur when there are at least 3 trucks busy, and then for the subset with at least 7 trucks already occupied. These confidence intervals are plotted in Figure 2.6. Again, MCRP shows the best performance, with both sides of the confidence intervals having the lowest values. The most significant improvement over the other methods is



(a) At least three trucks are busy

(b) At least seven trucks are busy

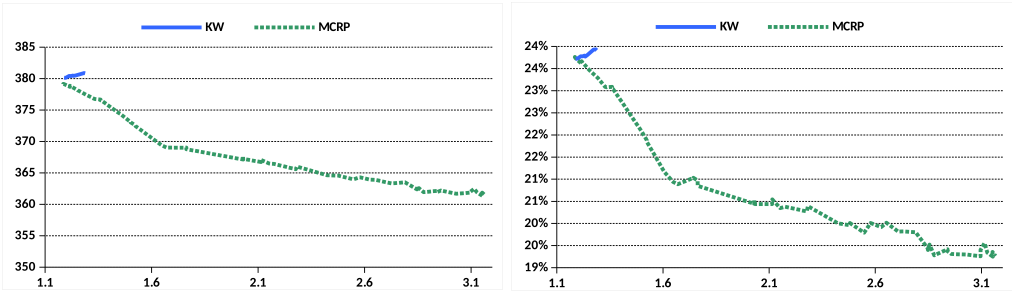
Figure 2.6: Confidence intervals

observed in terms of $FLAR$ when there are at least 7 trucks busy.

Varying parameter W

So far, we measured the performance with the two models KW and MCRP making the smallest number of relocations required to cover every response neighborhood. Now we show how the performance changes for the KW and MCRP models if we change the value of parameter W to allow for more relocations. We generate a sequence of incidents over 50 years time horizon. Then we run the two scenarios using the KW and MCRP models for different values of W so that the number of relocations made per major incident gradually increases from the minimum required to the maximum possible with both models. We vary W in the range between 0.01 and 0.999, and for each value of W we report the average number of relocations made per major incident and the average performance over the generated 50 years incidents sequence. In Figure 2.7, ART and $FLAR$ are plotted against the number of relocations made per major incident.

First, since the KW model associates costs with each relocation, and the objective is to minimize the total costs, it does not make many more relocations than the minimum required, even if we set the parameter W equal to 1 (see Appendix 2.8) and allow for as many relocations as possible. The minimum number of relocations needed to satisfy the constraints is 1.2 per major incident with both models. The maximum obtained is 1.3 with the KW model, and 3.2 with the MCRP model.



(a) ART

(b) FLAR

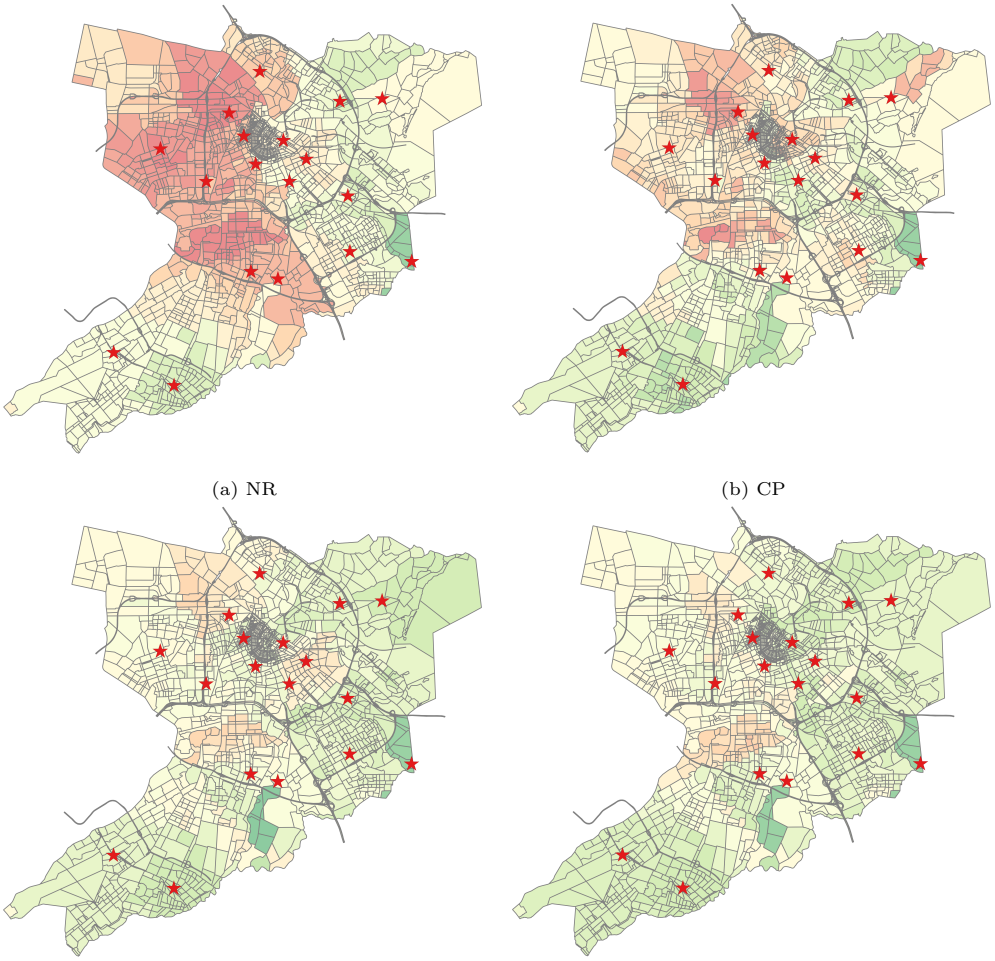
Figure 2.7: Change in performance if gradually increasing the number of relocations made per major incident

Secondly, we can see that making more relocations boosts the performance of the MCRP model. Making just about 0.5 relocations more than the minimum required decreased *ART* by 2.6% (10 seconds), and *FLAR* by 11.8%. In contrast, allowing for more relocations does not improve the performance of the KW model. In fact, when making more relocations the performance of the KW model slightly decreases, most likely due to the negative effects on the KW model's objective accuracy. The reasons behind this decrease in accuracy are discussed in more detail in Section 2.8.

Risk maps

So far, we looked at the overall performance over the entire region. We may also construct the risk maps of the region shown in Figure 2.8. To do this, we simulate 100 major incidents for each demand location. The size of each incident is sampled from the empirical distribution for the corresponding service area, and the duration is sampled from a Weibull distribution, as described in Section 2.6.1. For each major incident the relocations are made using one of the four strategies, with both KW and MCRP making the minimum number of relocations required to satisfy the constraints ($W = 0.01$). The new incidents are then generated until the major incident is resolved. We keep track of response times to those simultaneous incidents, and compute the *ART* for each demand location over all the incidents.

In the end, we get four values of the *ART* for each of the 2663 demand locations, so 10652 measurements in total. We use these values to color the demand locations in Figure 2.8. We pick an interval between 355 and 455 seconds containing about 98% of all 10652 observations excluding very small and very large *ART* values. We then divide



(a) NR

(b) CP

(c) KW

(d) MCRP

Figure 2.8: *ART* computed conditioning on the location of a major incident over all simultaneous events. Colors range from dark green (below 330 seconds) to dark red (above 449 seconds). The red stars represent the fire stations

this interval into subintervals of 5 seconds, and color the demand locations gradually changing from dark green (ART below 355 seconds) to dark red (ART above 455 seconds).

Figure 2.8 shows that the KW and MCRP models provide a significantly higher level of coverage during the major incidents overall and a much more fair coverage across the region. Comparing these two models, MCRP showed a better overall performance than KW while keeping a fair coverage of the whole region. For example, ART and $FLAR$ computed over all simultaneous incidents are 378 seconds and 24.6%, respectively, with the MCRP model against 383 seconds and 25.5% with the KW model.

2.7 Conclusion

In this chapter we considered the problem of relocating fire trucks during major incidents, to compensate for gaps in the coverage arising from the large number of trucks required for the incident. We proposed a novel relocation algorithm that solves a Maximum Coverage Relocation Problem (MCRP) whenever a major incident arises, in order to find the best relocations. The MCRP model is then tested by applying it to the operations of the FDAA. We calibrated the model based on ten years of historical incident data from the fire department, and used discrete-event simulation to evaluate its performance. We demonstrated that MCRP shows massive gains compared to not doing any relocations at all, and also provides significant improvement over the current practice at the FDAA. We also compared MCRP with the state-of-the-art as proposed by Kolesar and Walker [59]. We showed that MCRP performs better for larger incidents and, unlike the KW model, benefits from increasing willingness to make relocations. Moreover, MCRP is argued to be more flexible and easier to implement than KW.

For future work one could test MCRP on data from different fire departments, to better evaluate its performance across a wide range of possible scenarios. In addition, the framework presented here can be extended in various ways. First, our definition of coverage can be modified to include the risk in certain demand locations, in addition to the rate at which new incidents arise. For instance, a fire at a chemical plant may prove disastrous if not responded to in a timely manner, so a demand location housing a chemical plant should be weighted heavier than a demand location corresponding to farm land. Other extensions include dealing with incidents that require a mixture of different vehicle types (such as a ladder truck and a pumper) and explicitly modelling stochastic effects, such as random travel times and incident durations.

2.8 Appendices

In this section we provide supplementary materials to the main body of the chapter.

2.8.1 Algorithm Pseudocode

In this section we outline the relocation algorithm from Section 2.4 in a pseudocode. The algorithm uses the MCRP and LBAP formulations, and is launched whenever an incident occurs involving at least n_0 trucks. The fire department decides up front on the proper value of n_0 as discussed in the beginning of Section 2.4. Let $MCRP(n)$ be interpreted as a function that takes parameter n as an argument, solves MCRP with response neighborhoods of size n, and outputs $|\mathcal{N}| \times |\mathcal{N}|$ matrix X with elements $X_{ij} = x_{ij}$, that is, the solution of MCRP. Assume, $MCRP(n)$ outputs the empty matrix in case the corresponding MCRP is infeasible. Let $LBAP(X)$ be the function that takes the matrix $X = MCRP(n)$ as an argument, solves LBAP constructed as described in Section 2.4.2, and outputs $|\mathcal{N}| \times |\mathcal{N}|$ matrix $\hat{X}$ with elements $\hat{X}_{ij} = \hat{x}_{ij}$ (i.e., the solution of LBAP). We summarize the relocation algorithm in Algorithm 1.

Algorithm 1 Relocation algorithm

function RELOCATE(n_0)
  $n \leftarrow n_0$
  $X \leftarrow MCRP(n)$
  while X is empty **do**
    $n \leftarrow n + 1$
    $X \leftarrow MCRP(n)$
  $\hat{X} \leftarrow LBAP(X)$

2.8.2 MCRP Generalization

In this section we formulate the MCRP generalization that can incorporate various RN structures, potentially of different cardinality and coverage requirements. Let $\mathcal{K}$ be a set of response neighborhoods, where $k \in \mathcal{K}$ is a collection of demand locations for which at least b_k idle fire trucks are required to be at stations $\mathcal{N}_k \subseteq \mathcal{N}$. The following is the generalized formulation of MCRP:

$$\max W\Big(\sum_{i\in\mathcal{S}}\sum_{j\in\mathcal{E}} x_{ij}(d_j - d_i) + \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{E}} x_{ij}d_j - \sum_{i\in\mathcal{M}} z_i d_i\Big) \tag{2.17}$$
$$- \big(1-W\big)\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{E}} x_{ij}$$

$$\text{s.t.} \sum_{i\in\mathcal{N}} a_{ik}^n y_i \geq b_k, \qquad k\in\mathcal{K} \tag{2.18}$$

$$\sum_{j\in\mathcal{N}} x_{ij} \leq f_i, \qquad i\in\mathcal{N} \tag{2.19}$$

$$\sum_{j\in\mathcal{N}} x_{ji} \leq 1, \qquad i\in\mathcal{E} \tag{2.20}$$

$$1 - z_i \leq y_i, \qquad i\in\mathcal{M} \tag{2.21}$$

$$y_i = f_i + \sum_{j\in\mathcal{N}} x_{ji} - \sum_{j\in\mathcal{N}} x_{ij}, \qquad i\in\mathcal{N} \tag{2.22}$$

$$x_{ij} = 0, \qquad i\in\mathcal{N},\ j\in\mathcal{S}\cup\mathcal{M} \tag{2.23}$$

$$x_{ij}, z_i \in \{0,1\}, \qquad i,j\in\mathcal{N} \tag{2.24}$$

$$y_i \in \{0,1,...\}, \qquad i\in\mathcal{N} \tag{2.25}$$

This formulation differs from the formulation (2.1)-(2.9) in constraints (2.2). Here, instead of partitioning the region into the RNs $\mathcal{K}_n$ of the same cardinality n, any partitioning $\mathcal{K}$ of the region is possible. A partition $k\in\mathcal{K}$ is required to be covered by at least $b_k \in \mathbb{N}$ fire trucks instead of 1, as in (2.2).

2.8.3 Kolesar & Walker Formulation

Here we provide an extended version of the approach from Kolesar and Walker [59]. As mentioned in Section 2.3 the original definition of RN used in [59] implied a fixed size depending on the type of vehicle. We parametrize the size of RN to be able to extend it in case the model is infeasible for a given value of RN size. It allows us to use the KW model in the algorithm presented in Section 2.4 instead of the MCRP model. We also introduce the W parameter in the KW objective in the same manner as for the MCRP model to see how the model performs if we increase willingness to relocate (see Section 2.6.3).

In the KW formulation we use the same notations as in the MCRP formulation. The KW model can be formulated as follows:

$$\min\ W \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{N}} c_{ij}x_{ij} + (1-W) \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{N}} x_{ij} \tag{2.26}$$

$$\text{s.t.} \sum_{i\in\mathcal{N}} a^n_{ik} y_i \geq 1, \qquad k \in \mathcal{K}_n \tag{2.27}$$

$$y_i = f_i + \sum_{j\in\mathcal{N}} x_{ji} - \sum_{j\in\mathcal{N}} x_{ij}, \qquad i \in \mathcal{N} \tag{2.28}$$

$$\sum_{j\in\mathcal{N}} x_{ij} \leq f_i, \qquad i \in \mathcal{N} \tag{2.29}$$

$$x_{ij} \in \{0,1\}, \qquad i,j \in \mathcal{N} \tag{2.30}$$

$$y_i \in \{0,1,...\}, \qquad i \in \mathcal{N} \tag{2.31}$$

The objective function (2.26) consists of two parts. The first part is an indication of the expected total response time during the major incident multiplied by parameter W, as we discuss in detail in Section 2.8 below. The second part is the number of relocations made, multiplied by $1 - W$. This objective is equivalent to the original one if the W parameter is close enough to 0, so the minimum number of relocations is made to satisfy the constraints. Constraints (2.27) require every response neighborhood to be covered by at least one truck, and constraints (2.29) ensure not more than available trucks are relocated from every station. Note that the KW formulation does not have constraints (2.4). Those constraints prevent relocating more than one fire truck to the same station, which otherwise could happen in case of $W > 0$, as in MCRP each relocation is associated with a positive gain in the objective function. In the KW formulation, each relocation is associated with a cost. Relocating trucks beyond the first to an empty station can only make a feasible solution infeasible, by uncovering one or more response neighborhoods, while not increasing coverage. Hence, it is never optimal in the KW formulation to relocate more than one truck to the same empty station.

The main difference between the KW and the MCRP formulations is in the first component of the objective function. While the MCRP model maximizes the gains in coverage obtained by making relocations, the KW model minimizes the total costs $\sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{N}} c_{ij}x_{ij}$ incurred by making relocations. The c_{ij}'s themselves do not have a clear interpretation, but the difference in the objective between two candidate relocation solutions is an estimation of the difference in the expected total response time to the incidents arriving during the fire that triggered the relocation. In other words, the solution to the KW model minimizes an approximation of the expected total response time to incidents occurring during the major incident.

Computing these c_{ij} factors is a complex task that requires more detailed data and computations compared to MCRP. Below, we provide a procedure for computing the c_{ij} along the lines of [59]. This is intended both to clarify the interpretation mentioned above, as well as to illustrate why the performance of the KW model decreases for larger incidents, when more relocations are required to satisfy constraints (2.27).

Computing the coefficients c_{ij}

The definition of the c_{ij} is based on the square root law, which was first stated in [58] as a way to approximate the expected traveling time to an incident. Consider a region with area A that is served by N fire stations. By the square root law, the expected distance between the locations of the incidents and the fire stations closest to those incidents can be approximated as $D = K\sqrt{A/N}$, where K is some constant. In the remainder of this subsection, we describe how the authors in [59] propose to use the square root law to define and compute the c_{ij}.

Denote by A_i a physical area of the service area of station i, and by d_i the arrival rate of incidents in the service area of station i. Constants c_1 and c_2 are chosen such that $c_1\sqrt{A_i}$ is a good estimate of the expected response distance $D_i^{(1)}$ of the closest fire truck to the incidents in service area i, and $c_2\sqrt{A_i}$ is an estimate of the expected response distance $D_i^{(2)}$ of the second closest truck to the incidents in service area i. We will discuss choosing the c_1 and c_2 in Appendix 2.8 below.

Denote the average response velocity in the service area of station i by v_i. These can be evaluated using the distance and travelling time data. Let $i_j^{(1)}$ denote the station where the closest truck to j is located. The arrival rate of incidents in the service area of station i is computed as $d_i = \sum_{j:i_j^{(1)}=i} \lambda_j$. Let t denote the duration of the major incident, then the aggregate response time over all incidents in the service area of station i during the time interval $[0, t]$ can approximated with $c_1\sqrt{A_i}d_it/v_i$ if i has a truck available, and with $c_2\sqrt{A_i}d_it/v_i$ if it does not.

The KW model was developed with the main objective to cover all response neighborhoods with minimum number of relocations. In their iterative approach, the empty fire stations to be covered were defined first, and then the trucks were chosen for relocation to those empty fire stations. Assume that station $j \in \mathcal{J}$ is to be covered, and a set of stations $\mathcal{I}$ have a truck available for relocation. We need to decide from which station $i \in \mathcal{I}$ to relocate a truck to station j. Denote $\alpha_i = d_i\sqrt{A_i}/v_i$ and let r_{ij} be the driving time from station $i \in \mathcal{I}$ to station j. Let $T > t$ denote the time when the major incident is finished,

and all trucks have returned to their original stations, then the aggregate response time over all incidents during $[0, T]$ in the response area of the region $\mathcal{I} \cup \{j\}$, given that a truck from station $i \in \mathcal{I}$ is relocated to the empty station j, can be approximated with

$$(c_2 - c_1)[\alpha_i(t + r_{ij}) + \alpha_j r_{ij}] + c_1 T \sum_{k \in \mathcal{I} \cup \{j\}} \alpha_k.$$

The second term $c_1 T \sum_{k \in \mathcal{I} \cup \{j\}} \alpha_k$ in the expression above indicates the total response time in case all the station had an idle truck, and the first term accounts for the fact that demand locations in the service areas of stations i and j are served by the second closest truck during $t + r_{ij}$ and r_{ij} time units, respectively. As the second term is the same for any potential relocation, it is then omitted, and the cost c_{ij} of relocating an available fire truck from station i to an empty station j is approximated by

$$c_{ij} = (c_2 - c_1)[\alpha_i(t + r_{ij}) + \alpha_j r_{ij}]. \tag{2.32}$$

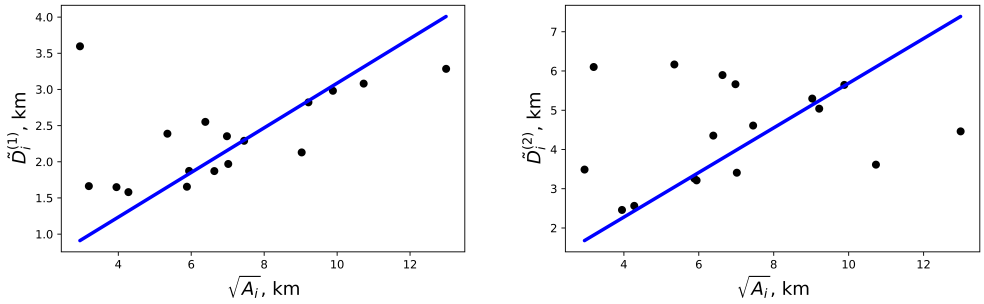
In our implementation of the KW model, the value t in (2.32) for the duration of a major incident is picked as a sample average over the historical incidents data, and is equal to three hours.

Fitting historical data

Recall that c_1 (c_2) denotes a constant such that $c_1\sqrt{A_i}$ ($c_2\sqrt{A_i}$) is a good approximation for the expected response distance in region i if the closest (second-closest) truck is dispatched. In order to estimate the parameters c_1 and c_2 we use linear regression based on the following data of the FDAA. The arrival rates λ_j of new incidents for every demand location j, the distance d_{ij} and the travel time t_{ij} between any pair of a demand location j and fire station i.

Based on the given travel times, the service areas are constructed for every station i, and the physical area A_i is computed for a corresponding service area. The expected distance of the closest and the second closest trucks to incidents arriving in a service area i is computed based on the provided arrival rates, travel times (for $D_i^{(2)}$ to define which truck is second closest for every demand location in a given service area), and distances. Remember that $i_j^{(1)}$ denotes the station where the closest truck to j is located. Let also $i_j^{(2)}$ indicate the fire station with the second closest truck to demand location j. Given the data mentioned above, we can estimate the expected travelling distance of the closest and the second closest truck in a service area of station i as

$$\tilde{D}_i^{(1)} = \frac{\sum_{j:i_j^{(1)}=i} \lambda_j d_{i_j^{(1)} j}}{\sum_{j:i_j^{(1)}=i} \lambda_j} \quad \text{and} \quad \tilde{D}_i^{(2)} = \frac{\sum_{j:i_j^{(1)}=i} \lambda_j d_{i_j^{(2)} j}}{\sum_{j:i_j^{(1)}=i} \lambda_j},$$



(a) c_1 (b) c_2

Figure 2.9: Linear regression for c_1 and c_2 parameters.

respectively. Based on the obtained estimations $\tilde{D}_i^{(1)}$ and $\tilde{D}_i^{(2)}$ for 17 fire stations, and the corresponding data on physical areas A_i, a simple linear regression is fit to model the relationships $D_i^{(1)} = c_1\sqrt{A_i}$ and $D_i^{(2)} = c_2\sqrt{A_i}$. The obtained linear regression is shown on Figure 2.9.

As the graphs show, the linear regression does not fit the data well. Specifically, the coefficient of determination R^2 is equal to -0.58 for the c_1 regression and to -1.19 for the c_2 model. The root-mean-sqare error (RMSE) is 0.77 and 1.79, respectively. The coefficient of determination is computed as $R^2 = 1 - SS_{res}/SS_{tot}$, where SS_{res} is the residual sum of squares and SS_{tot} is the total sum of squares. Hence, the negative value of R^2 means that a horizontal line that is the mean of the data provides a better fit than does the fitted function. We conjecture that this poor fit is due to the irregular road network in the FDAA coverage area, which is in sharp contrast with the grid-like network in NY, for which the approach in [58, 59] was developed.

Implementing the KW model

In order to implement the KW model, we require the following data:

- the arrival rate of new incidents λ_j per demand location;
- the traveling times r_{ij} between each pair of demand location and fire station;
- the traveling distances d_{ij} between each pair of demand location and fire station;
- the physical area A_i of each service area;

- the duration of major incidents.

In contrast, to implement the MCRP model, we require only the following:

- the arrival rate of new incidents d_i per service area;

- the traveling times r_{ij} between each pair of demand location and fire station.

Note that obtaining the arrival rate per service area d_i is much easier than finding the arrival rate λ_j per demand location, since the latter is much more granular.

Clearly, the data requirements for MCRP are much lighter compared to KW. Moreover, the computations required to implement KW outlined in Sections 2.8 and 2.8 are more complex than those for MCRP, and require expert knowledge to execute. Consequently, the threshold for implementing MCRP should be much lower than for KW.

Looking at the KW formulation and the computation of the c_{ij}, we observe that the authors of [59] make a number of significant assumptions and approximation steps that may result in inaccuracies, in particular as the size of the major incident grows. For instance, it requires an estimate up front for the duration t of the major incident. Given the substantial variability of these durations (for FDAA the historical duration of major incidents ranges from 1 hour to a full day), requiring a single point estimate for the duration has significant impact on the objective function and the accuracy. Moreover, KW leans heavily on the square root law from [58], which as we have seen in Section 2.8 is not accurate in the coverage area of FDAA. We conjecture that its successful usage in NY is due to that city's regular road network. Both these errors compound when the size of the major incident grows.

Upon closer inspection of the c_{ij} components, we see that in computing these it is always assumed that the second-closest truck is dispatched in case that the closest truck is not available. This is of course not true in practice, since sometimes both the first and second-closest trucks are unavailable. This is particularly likely during large incidents, which explains why KW becomes less accurate in that regime. Furthermore, when it comes to the c_{ij}, the authors of [59] write "*each relocation cost c_{ij} [...] depends on the resultant configuration of houses to be filled and to be left empty [...]. However, we can approximate the c_{ij} by taking an average configuration*". So, in [59], the authors use some 'default' configuration rather than the current one, the gap between which again grows with the incident size.

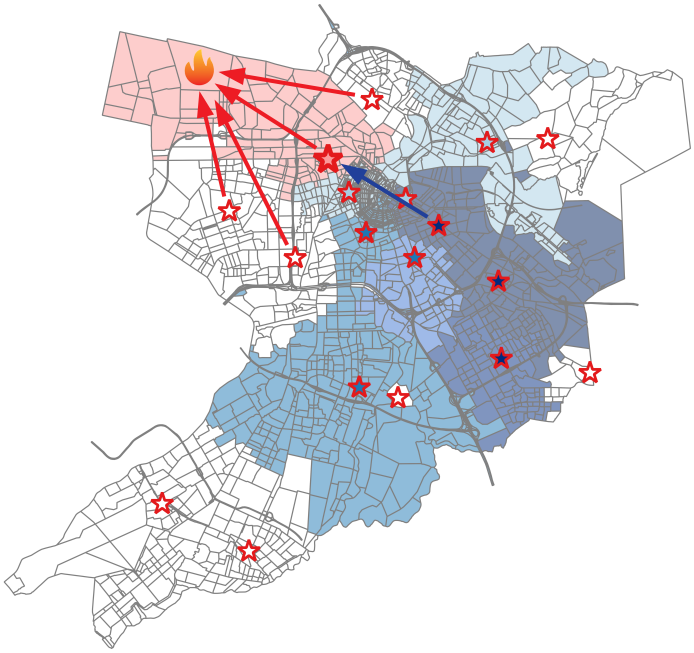


Figure 2.10: Current Practice algorithm example

2.8.4 'Current Practice' Algorithm

In this section, we describe the CP algorithm using the example from Figure 2.10. The service area of the fire station corresponding to the major incident's demand location is painted red. The service areas of the other empty and volunteer stations are painted white, and the service areas of the fire stations with available professional trucks are painted blue. If a major incident happened, and several trucks are dispatched to its location (flame icon), the CP algorithm relocates one of the available professional trucks to the fire station (big star) servicing the major incident's demand location. The procedure identifying which truck to relocate is as follows. The available professional trucks are first ordered according to their mean response time corresponding to the incident's demand location. Then these trucks are divided into three groups. Assume there are N trucks available for relocation. The first $\lfloor N/3 \rfloor$ trucks from the ordered list are put into the first group (light blue), the next $\lfloor N/3 \rfloor$ trucks are put into the second group (blue), and the last $N - 2\lfloor N/3 \rfloor$ trucks are put into the third group (dark blue). The first truck from the third group is then chosen for relocation.

3

Dispatching Fire Trucks under Stochastic Driving Times

In this chapter we discuss optimal dispatching of fire trucks under stochastic driving times. The study is motivated by a particular dispatching problem that arises at the FDAA, where two fire trucks are sent to the same incident location for a quick response. We formulate the dispatching problem as a Markov Decision Process, and numerically obtain the optimal dispatching decisions using policy iteration. We show that the fraction of late arrivals can be significantly reduced by deviating from current practice of dispatching the closest available trucks, with a relative improvement in terms of fraction of late arrivals of on average about 20%, and over 50% for certain instances. We also show that driving-time correlation has a non-negligible impact on decision making, and if ignored may lead to performance decrease of over 20% in certain cases. As the optimal policy cannot be computed for problems of realistic size due to the computational complexity of the policy iteration algorithm, we propose a dispatching heuristic based on a queueing approximation for the state of the network. We show that the performance of this heuristic is close to the optimal policy, and requires significantly less computational effort.

The work in this chapter is based on [106]: D. Usanov, P.M. van de Ven, and R.D. van der Mei. Dispatching fire trucks under stochastic driving times. *Computers & Operations Research*, 114, 2020.

3.1 Introduction

When a new fire arises, one or more trucks are dispatched from the fire stations close to the fire in order to facilitate a quick response. However, sending closest trucks may lead to gaps in coverage for the duration of an incident which may have adverse effect on the response time to incidents that happen simultaneously. This is particularly true for large fires that require multiple trucks and take longer to put out. In this chapter, we study how to dispatch multiple fire trucks under stochastic driving times in order to respond quickly to the present fire, while maintaining good coverage for possible simultaneous incidents.

To illustrate this tradeoff, we consider the example of FDAA, which operates 19 fire stations spread across the city of Amsterdam and surrounding areas. When a small fire occurs in the city center of Amsterdam that only requires a single fire truck to address, the FDAA nevertheless dispatches two trucks from different fire stations. These incidents are of the highest (of three) priority level, and constitute about 70% of all fires. When the first truck arrives at the fire, the second truck returns to its fire station. This type of redundancy is used by FDAA in order to mitigate delays that may arise when a fire truck encounters a traffic jam or a large garbage truck. Intuitively, the dispatcher would want to ensure that these two trucks are relatively close to the fire, but still sufficiently spaced out so that the remaining trucks retain good coverage. Moreover, we would want the trucks to approach the fire from different directions, so that when one truck gets stuck in traffic, the other can still get to the fire quickly. We refer to the latter phenomenon as *driving-time correlation*, and observe that this adds yet another layer of complexity to the optimal dispatching problem.

Although the problem of dispatching a single vehicle to incidents has been studied extensively in the literature on emergency services, to our knowledge very little work has been done on dispatching multiple vehicles, and we are the first to consider driving-time correlation in this context. Moreover, we are not aware of any studies into driving-time correlation in the transportation literature either. The current practice of the FDAA is to dispatch a truck each from the two fire stations closest to the incident. However, it is unclear whether this leads to the fastest response (given the correlated driving times), and leaves the best coverage. Naturally, driving-time correlation also plays a role when considering incidents that require more than two trucks, but for ease of presentation we limit ourselves to the case with two trucks. While this problem is motivated by the situation of the FDAA, we believe other major cities with busy traffic use, or could benefit from, similar dispatching methods.

To study this problem, we model the city as a graph, where the vertices correspond

to demand locations where incidents may occur, and an edge indicates that two locations can be reached directly. Fire stations are positioned at some of the vertices, and new fires arise at random times and locations. Similar to the current practice of the FDAA, we assume that fires are addressed by sending two fire trucks, the first of which to arrive will engage the fire.[1] The response time of a truck dispatched from a fire station to a fire is the sum of travel times over all edges traversed on the graph, and the travel time over each edge is modelled as random variable. When two trucks dispatched to the same fire use the same edge they may incur the same travel times, capturing the driving time correlation. Fires last for some random time, after which the trucks become idle again. In order to determine the optimal dispatching policy we model this system as a Markov decision process (MDP).

We first use policy iteration to numerically determine the optimal dispatching policy, and show that significant improvements can be made over the current practice of sending the two closest idle trucks. We also use this approach to demonstrate that it is important to take into account driving-time correlation in the model, since dispatch decision and performance metrics may be incorrect otherwise. For realistic-sized instances such as the coverage area of FDAA we cannot use policy iteration due to its computational complexity, and we develop novel heuristics instead.

Inspired by the results in [101], we develop these heuristics using the idea of one-step improvement. This approach was developed in [80, 82], and has for instance been applied to call centers [10], control of traffic lights [36], routing in queueing networks [11] and loss networks [40]. To do this, we first obtain an approximation for the fraction of late arrivals under the policy of sending the closest trucks, assuming that all fire stations are independent from each other. We then apply a single policy-iteration step to these results in order to obtain an improved policy. We show that the resulting policy significantly outperforms closest-first. The computational complexity of this approach is much better than that of the full policy iteration algorithm needed to obtain the optimal dispatching policy, yet its performance is remarkably close to optimal.

To summarize, in this chapter we make the following contributions:

1. We develop the first model for dispatching multiple trucks in an emergency service network setting, possibly in the presence of correlated (stochastic) driving times;

2. We show that the current fire department practice of sending the closest trucks is far from optimal, the optimality gap grows with the number of trucks in the system

[1] Note that we limit ourselves to the case of two trucks for simplicity, but we expect that our approach, heuristics and insights hold for larger fires that require more trucks.

and can be as large as 50% for certain problem instances;

3. We show that taking into account driving time correlation has a significant impact on the response time and the optimal dispatch policy, and ignoring correlation when deriving a policy may lead to performance loss of more than 20%;

4. To circumvent computational issues for obtaining the optimal dispatch policy, we propose a new heuristic based on 1-step policy improvement that has a small optimality gap, but only requires a fraction of its computation time.

The remainder of this chapter is organized as follows. In Section 3.2, we provide a review of the relevant literature, and in Section 3.3, we give a description of the model studied in the chapter, explain how we account for driving-time correlation, and formulate the MDP. In Section 3.4 we discuss one-step improvement policy and introduce our heuristics. In Section 3.5 we numerically investigate the impact of correlation, compare the performance of the optimal policy, closest-first and the heuristics. Conclusions and suggestions for further research are made in Section 3.6.

3.2 Literature Review

Operations Research related to fire departments can be traced back to the seminal RAND fire project, which ran from 1968 to 1975 and addressed a range of issues related to the New York City fire department. This includes for instance developing a simulation model for fire fighting services [15], a square root law for fire fighting response times [58], and algorithms for relocations during major incidents [59]. We refer to [34] for an overview of this project and its research output. Since then the research literature on fire department operations has been limited in both scope and quantity, focussing mostly on facility location problems. The goal of this stream of literature is to determine the optimal location of the fire stations (see, e.g., [69, 44, 16, 21]).

To our knowledge, the only papers that deal specifically with dispatching of fire trucks are [100] and [43], both originating from the RAND fire project. In [100], the authors consider whether to dispatch one or two fire trucks to incidents of unknown severity, and show that the optimal policy has a threshold structure, where one dispatches two trucks only if sufficient trucks are available. However, this work ignores the spatial component and does not determine *which* trucks to dispatch. The work closest to ours is perhaps [43], where the authors propose an algorithm for *how many* (one or two) and *which* trucks to dispatch. The objective of the algorithm is to minimize response time to serious incidents, those requiring at least two ladder trucks. The algorithm performs a grid search, where the first truck is picked for dispatching based on a certain loss approximation, assuming

that only that truck is dispatched. Then, given the choice of the first truck, the second truck is decided on based on another loss function. Finally, the decision is made whether to send only the first truck, or both of them, based on the corresponding estimated costs. In contrast to our work, [43] relies on heuristic arguments for determining the future costs of current dispatching decisions, and ignores driving-time correlation. Moreover, the used loss functions do not seem to have an intuitive interpretation, and dispatching of the first truck is done independently of whether the second truck will be dispatched or not. In contrast, our approach is to jointly pick the two trucks to be dispatched such that the fraction of late arrivals is minimized, allowing to incorporate driving-time correlation.

An area that is closely related to fire truck dispatching is that of dispatching ambulances to accidents and other emergencies. We will discuss the most relevant literature below, but emphasize that to our knowledge most of this work only considers dispatching a single vehicle to incidents, and does not take into account driving-time correlation. While results on the optimal dispatching of a single ambulance are not directly applicable to our setting, we now provide a brief discussion of some recent developments in this area. See for instance [45, 25, 7] for a more complete overview of this field. In [2], a dispatching heuristic was proposed based on the notion of preparedness, measuring the ability of the system to respond quickly to future incidents. The heuristic suggests to dispatch an ambulance resulting in the smallest decrease in preparedness. The algorithm was further studied in [65]. It was shown that the preparedness algorithm performs significantly worse than sending the closest ambulance in terms of average response time. The authors noted, however, that the poor performance of the preparedness algorithm is due to the fact that it ignores the current response time when making a dispatching decision. They introduced a modified version of the algorithm that balances between the decrease in preparedness and the response time to the current incident. In their experiments, the extended algorithm outperformed the closest-first dispatching policy.

In [67], the authors consider a setting with multiple incident priority levels, and compare a range of dispatching policies based on the closest-first policy. Modifications include possibilities to reroute busy ambulances to more urgent incidents and to reassign incidents to ambulances that become idle. The authors conclude that the relative performance of each policy depends on the parameters and available infrastructure. In [48], the authors formulate the problem of ambulance dispatching as an MDP, and then present a heuristic which is shown to perform close-to-optimal, and in certain cases outperforms closest-first. In [5] and [6], *patient survivability* is used as an objective for the problem with different incident priority levels. The authors formulate the problem as an MDP, and observe that dispatching closest vehicle is only optimal for the most urgent incidents. They also indicate that the optimal policy is most beneficial when the spatial distribution of incidents is skewed, which is the case in most real-life applications. Using the insights

obtained from the optimal policy, the authors introduce a heuristic that outperforms the closest-first policy. The authors of [72, 73] provide an MDP formulation of the ambulance dispatching problem under certain fairness constraints, and numerically compute the optimal policy for small instances. The problem of possibly sending two different types of emergency vehicles is considered in [98], where the authors propose a heuristic for this purpose.

In addition to dispatching, substantial work in recent years has focused on relocation as well as joint dispatching and relocation of ambulances, in order to maintain better coverage. The relocation decisions imply proactive repositioning of idle vehicles within the region with the aim to reduce response time to future incidents. In [71, 93, 70, 79], the joint problem was addressed using approximate dynamic programming. In [25], the authors use stochastic programming to solve this problem, while ensuring that the workload due to relocations remains limited. The optimization method in [24] is designed to make relocations that maximize coverage under personnel's workload limitations. Low computation costs of the approach allow to make decisions in real time, in contrast to the earlier methods described, which require offline computations.

As mentioned earlier, the research on ambulance dispatching is mostly focused on the setting where exactly one vehicle is required to serve an incident. To understand why results for the single-vehicle case cannot easily be applied in our multiple-vehicle setting, consider the following. First, any dispatching action is a trade-off between a quick response, and ensuring that the remaining coverage is sufficient, should another incident arise while the first incident is still ongoing. Decomposing a multiple-vehicle dispatching formulation into a sequence of independent single-vehicle problems, one may not be able to carefully navigate this tradeoff, since every dispatching decision is made in a greedy way (assuming it is the only such decision). To illustrate this, consider the easier problem having to remove k trucks: which set of k trucks would result in the best coverage? It is easy to see that solving the problem sequentially would likely result in a substantially different solution with a worse coverage compared with solving the problem jointly for all trucks.

The second reason why algorithms for dispatching a single vehicle cannot be easily applied in our setting is due to the driving-time correlation. If applying single-vehicle policies for dispatching multiple trucks, one would be unable to take into account this correlation. As we shall show in this chapter, driving-time correlation has a significant impact on the optimal dispatching policy, and ignoring it substantially reduces performance.

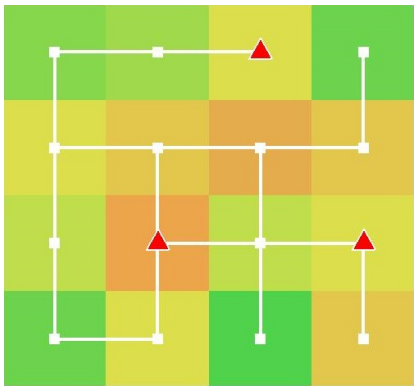


Figure 3.1: Graph representation of a region served by a fire department

3.3 Model

We consider a city represented by a connected, undirected graph $(\mathcal{J}, E)$, see Figure 3.1. The set of vertices $\mathcal{J} = \{1, ..., J\}$ represents the neighborhoods, or *demand locations*. Two vertices are connected if it is possible to travel directly between these two demand locations. A subset $\mathcal{I} \subseteq \mathcal{J}$ of demand locations contain a fire station (marked with triangles in Figure 3.1), and we denote $I = |\mathcal{I}|$. Fire station $i \in \mathcal{I}$ houses C_i fire trucks, and all fire trucks are assumed to be identical.

We assume that new fires arise at each demand location $j \in \mathcal{J}$ according to a Poisson process with rate λ_j. Fire trucks can be either idle (i.e., waiting at a fire station) or busy (i.e., travelling or fighting a fire), and whenever a new fire starts, two idle fire trucks have to be dispatched. If a fire starts and fewer than two idle trucks are available, we request the missing truck(s) from a neighboring region. We assume the neighboring regions have ample capacity, so there are always trucks available. For tractability, we assume that when a truck is dispatched, it remains busy for an exponential time with rate μ, independent from the other truck dispatched and from the location of the fire and fire station. Independence from the location of the fire and fire station is a reasonable assumption as in practice the traveling time is negligible compared to the on-scene service time. Note that the independence assumption allows us to consider any travel time distribution, although we shall focus mostly on Erlang-distributed travel times, for ease of presentation. The assumption that both trucks have the same busy-time distribution will result in an upper bound on the real-life busy fraction, since only the first truck to arrive will stay to resolve the incident. However, given the relatively low busy fraction for the

fire truck application domain, we expect our model to be sufficiently accurate. Returning trucks can be dispatched once they reach their station. Although an idealization, this assumption has negligible impact given the relatively low busy fraction of fire trucks seen in practice, and given the traveling time is small compared to the on-scene service time. This means it is rarely the case a fire truck is driving upon arrival of a new incident.

The state of the system can be represented by a vector $\boldsymbol{f} = (f_1, \ldots, f_I)$, where f_i denotes the number of idle trucks at station i. Let the vector $\boldsymbol{a}(\boldsymbol{f}, j) = (a_1(\boldsymbol{f}, j), ..., a_I(\boldsymbol{f}, j))$ represent the dispatch action taken if a new fire starts at a location j when in state $\boldsymbol{f}$. Here $0 \leq a_i(\boldsymbol{f}, j) \leq f_i$ denotes the number of trucks dispatched from station $i \in \mathcal{I}$. Given that exactly two trucks are dispatched to every fire we have that $|\boldsymbol{a}(\boldsymbol{f}, j)| \leq 2$, where the remaining $2 - |\boldsymbol{a}|$ trucks are sent from neighboring regions.

We denote by $\boldsymbol{F}^{\boldsymbol{a}}(t)$ the state of the system at time t under decision rule $\boldsymbol{a}$. Observe that, due to the exponentiality assumptions, the process $\{\boldsymbol{F}^{\boldsymbol{a}}(t)\}_{t\geq 0}$ is a continuous-time Markov process, with state space $\mathcal{S} = \{(f_1, ..., f_I) | 0 \leq f_i \leq C_i \ \forall i \in \mathcal{I}\}$. Let $\boldsymbol{e}_i$ denote a vector of length I with ith element equal to 1, and all other elements equal to zero. The transition rates q of this process are given by

$$\begin{aligned} q(\boldsymbol{f}, \boldsymbol{f} - \boldsymbol{a}(\boldsymbol{f}, j)) &= \lambda_j, & j \in \mathcal{J},\ \boldsymbol{f} \in \mathcal{S}, \\ q(\boldsymbol{f}, \boldsymbol{f} + \boldsymbol{e}_i) &= (C_i - f_i)\mu, & i \in \mathcal{I},\ \boldsymbol{f} \in \mathcal{S}. \end{aligned}$$

The first transition corresponds to trucks being dispatched upon the start of a new fire at location j, where the number of trucks at each location i is reduced from f_i to $f_i - a_i(\boldsymbol{f}, j)$. These transitions occur at rate λ_j, the rate at which new incidents start at demand location j. The second transition corresponds to a truck returning to its fire station and becoming idle. This happens at rate μ for each individual truck not at its station, so the rate of trucks returning to station i is equal to $(C_i - f_i)\mu$. This model is an example of the hypercube model from [63]. The hypercube model consists of a multiserver queueing model with distinguishable servers, corresponding to fire trucks in our setting. In [50], the authors numerically compute the optimal assignment policy of servers to requests in the hypercube model, and show that assigning the lowest-cost (closest in our setting) server is only optimal for small loads. The model is of relatively limited use in our setting, however, in that it cannot fully take into account the spatial component of our problem, and is only concerned with allocating a single server (dispatching a single truck). In [41], a hypercube model was proposed used to analyze a system with particular dispatching policies including multiple dispatch and partial backup. This model was further embedded into a genetic algorithm in [42] to optimize the service areas of ambulance bases.

3.3.1 Traveling and Response Time

When a truck is dispatched from fire station $i \in \mathcal{I}$ to demand location $j \in \mathcal{J}$, it travels along a shortest path on the graph. Let $s(i,j) \subseteq E$ denote the set of edges constituting the path. Since we assume that the graph is connected, such a path always exists. In case multiple shortest paths exist, we select one at random. The travel time along edge $e \in E$ is denoted by $X_e \sim \exp(1)$, and follows an independent exponential random variable with unit mean. So the marginal traveling time of a fire truck dispatched from i to j is given by $T_{i,j} = \sum_{e \in s(i,j)} X_e$, an Erlang-distributed random variable with $|s(i,j)|$ phases of unit mean.

For fire trucks dispatched from neighboring regions, we assume a traveling time T_0 independent of the demand location of the fire, as typically those trucks are located relatively far away, and the driving time is dominated by the time it takes to reach the city in the first place. We assume that T_0 has an Erlang distribution with $2 \max_{i \in \mathcal{I}, j \in \mathcal{J}} |s(i,j)|$ phases of unit mean. That is, the expected traveling time for a truck from a neighboring region is twice the maximum expected traveling time between any fire station-demand location pair on the graph, to reflect the fact that these trucks have to travel further.

The performance of a fire department is measured based on the response time to incidents, i.e., the time between the moment a fire reported and when the first truck arrives on scene. We consider two cases for computing the response time: *uncorrelated* and *correlated*. In the first case, we use the simplifying assumption that the driving time on the same edge is independent between both dispatched fire trucks. In the correlated case, we assume that both trucks incur the same driving time realization for each shared edge. We now discuss both cases in more detail.

Case 1: Uncorrelated driving times

In order to model the fact that in the uncorrelated case the response times of the two trucks that are dispatched are completely independent, we introduce two independent copies of the driving time random variable over each edge. To do this, we introduce an index $v = 1, 2$, which is used to distinguish between the two trucks that are dispatched, and is distinct from the index $i \in \mathcal{I}$ we use to index over all trucks. We denote by $X_e^{(v)}$ the driving time of truck v over edge $e \in E$, for $v = 1, 2$, and we assume that $X_e^{(1)}$ and $X_e^{(2)}$ are independent. We first treat the case where no trucks are sent from outside, and truck v is dispatched from location i_v, $v = 1, 2$. In this case, the total traveling time of the v-th truck to j can be written as $T_{i_v,j}^{(v)} = \sum_{e \in s(i_v,j)} X_e^{(v)}$, $v = 1, 2$. These $T_{i_v,j}^{(v)}$ are mutually independent because the $X_e^{(v)}$ are, even when $i_1 = i_2$. The $T_{i_v,j}^{(v)}$ follow an

Erlang distribution with $|s(i_v, j)|$ phases of unit mean.

In case one truck is dispatched from outside, we assume its traveling time is independent from the truck dispatched from inside the system; if two trucks are dispatched from outside their traveling times are assumed to be mutually independent. We denote by $T_0^{(1)}$ and $T_0^{(2)}$ two i.i.d. copies of the Erlang-distributed random variable T_0.

Summarizing, in the uncorrelated case, given a dispatch decision $\boldsymbol{a}$ for a fire at location j, the response time can be expressed as

$$R(\boldsymbol{a}, j) = \begin{cases} \min\{T_{i,j}^{(1)}, T_{i,j}^{(2)}\} & \text{if } a_i = 2, \\ \min\{T_{i_1,j}^{(1)}, T_{i_2,j}^{(2)}\} & \text{if } a_{i_1} = a_{i_2} = 1,\ i_1 \neq i_2, \\ \min\{T_{i,j}^{(1)}, T_0^{(1)}\} & \text{if } a_i = 1,\ |\boldsymbol{a}| = 1, \\ \min\{T_0^{(1)}, T_0^{(2)}\} & \text{if } |\boldsymbol{a}| = 0. \end{cases} \tag{3.1}$$

The first two entries correspond to the case where two trucks are dispatched from inside the network with the first covering the case where both trucks are sent from the same location, and the second the case with different locations. Note that if the trucks are dispatched from the same station, they follow the same shortest path in a graph. This is a reasonable assumption, as it is unlikely that in reality there are two independent shortest paths. Moreover, an alternative solution of sending the trucks via two different paths is difficult to sell at the fire department, as it is counterintuitive to the goal of getting to the incident as quickly as possible. The third and fourth entry in (3.1) correspond to the case where one and two trucks are dispatched from outside, respectively.

Case 2: Correlated driving times

In the correlated case the traveling times are no longer independent from each other, and we denote by X_e the shared random traveling time over edge $e \in \mathcal{E}$ for both trucks. In contrast to the uncorrelated case, we need not distinguish between both trucks to compute the traveling time, and we denote $T_{i,j} = \sum_{e \in s(i,j)} X_e$ as the traveling time from i to j over $s(i, j)$, which is an Erlang distributed random variable with $|s(i, j)|$ phases of unit mean. The traveling time of trucks dispatched from outside the network are still assumed to be independent from traveling times inside the network and from each other. Thus, in the correlated case the response time is given as follows:

$$R(\boldsymbol{a}, j) = \begin{cases} T_{i,j} & \text{if } a_i = 2, \\ \min\{T_{i_1,j}, T_{i_2,j}\} & \text{if } a_{i_1} = a_{i_2} = 1,\ i_1 \neq i_2, \\ \min\{T_{i,j}, T_0^{(1)}\} & \text{if } a_i = 1,\ |\boldsymbol{a}| = 1, \\ \min\{T_0^{(1)}, T_0^{(2)}\} & \text{if } |\boldsymbol{a}| = 0. \end{cases} \tag{3.2}$$

The entries correspond to the same decisions as in (3.1) (respectively: two trucks from the same location, two trucks from different locations, one truck from outside the network, both trucks from outside the network). Note that in comparison to (3.1), the first entry no longer contains a minimum operator, since both trucks will have the same driving time realization as they are dispatched from the same location and there is correlation. The second entry is no longer necessarily a minimum between two independent Erlang-distributed random variables, as the routes of the two trucks may share one or more edges on the graph, for which they will see the same driving time realization.

We emphasize that our approach described above for modeling driving-time correlation is certainly not the only possibility, and this work should be seen as the first attempt to take this phenomenon into account when making dispatching decisions. For instance, we assume complete correlation between the driving time on each shared edge, whereas a smaller but still positive correlation coefficient may be more realistic. We briefly discuss this extension in Section 3.6.

For each incident, we are interested in whether the response time is within some time limit t^*, and we say a late arrival occurred otherwise. Our goal is to minimize the fraction of late arrivals. This is one of the most widely used performance metrics in emergency services, and is for instance used by the FDAA and the Dutch government to measure the performance of fire fighting services.

3.3.2 MDP Formulation

We are interested in finding the dispatch decisions $\boldsymbol{a}(\boldsymbol{f}, j)$ that minimize the fraction of late arrivals. In order to determine these, we describe the system as an infinite-horizon average-cost Markov decision process (MDP). To do this, we first uniformize our Markov process $\{\boldsymbol{F}^{\boldsymbol{a}}(t)\}_{t\geq 0}$ by adding the following dummy transitions: $q(\boldsymbol{f}, \boldsymbol{f}) = \mu \sum_{i\in\mathcal{I}} f_i$. This ensures that transitions out of any state happen at rate $\tau = \sum_{j\in\mathcal{J}} \lambda_j + \sum_{i\in\mathcal{I}} C_i$, without altering the dynamics of the network.

We are now in position to formulate our infinite-horizon average-cost MDP. Note that when a new fire starts and the network is in state $\boldsymbol{f}$, we can make any of the following decisions $\boldsymbol{a}$:

$$\mathcal{A}(\boldsymbol{f}) = \{\boldsymbol{a} \in \mathbb{N}_0^I \mid 0 \leq a_i \leq f_i,\ \min\{2, |\boldsymbol{f}|\} \leq \sum_{i=1}^{I} a_i \leq 2\},$$

i.e., we dispatch at most two trucks from inside the region, and we only dispatch outside trucks if fewer than two idle trucks are available. This description also states that we cannot dispatch more trucks from each station than available. Let $h^*(\boldsymbol{f})$ denote the

relative cost incurred over an infinite time horizon when starting in state $\boldsymbol{f} \in \mathcal{S}$, compared to paying the average cost g^* every time unit. Since our process is unichain and has a finite state space and action space, we know from [89, Theorem 8.4.3] that there exists an optimal deterministic policy that satisfies the Bellman equations:

$$\begin{aligned} h^*(\boldsymbol{f})\tau = &- g^* + \mu \sum_{i \in \mathcal{I}} (C_i - f_i) h^*(\boldsymbol{f} + \boldsymbol{e}_i) + \mu \sum_{i \in \mathcal{I}} f_i h^*(\boldsymbol{f}) \\ &+ \sum_{j \in \mathcal{J}} \lambda_j \min_{\boldsymbol{a} \in \mathcal{A}(\boldsymbol{f})} \{\mathbb{P}(R(\boldsymbol{a}, j) > t^*) + h^*(\boldsymbol{f} - \boldsymbol{a})\}, \quad \boldsymbol{f} \in \mathcal{S}. \end{aligned} \tag{3.3}$$

The first summation on the right-hand side of (3.3) corresponds to fire trucks returning to their fire station, and the second to dummy transitions needed for uniformization. In neither case do we incur a cost or have to make a decision. The third summation corresponds to new fires that occur, in which case we have to make a dispatch decision $\boldsymbol{a}$, and incur some costs $\mathbb{P}(R(\boldsymbol{a}, j) > t^*)$ equal to the probability of exceeding the response time threshold t^*, given the dispatch decision and location of the fire. The value function g^* has an interpretation of the rate of late arrivals, that is, the average number of arrivals per time unit that were later than the time threshold t^*. To measure the performance of the dispatching policies we use the *fraction of late arrivals*, which is equal to $\frac{g^*}{\sum_{j \in \mathcal{J}} \lambda_j}$.

To compute the immediate costs $\mathbb{P}(R(\boldsymbol{a}, j) > t^*)$, we must take a closer look at the distribution of the response time $R(\boldsymbol{a}, j)$, presented in (3.1) and (3.2) for uncorrelated and correlated driving times, respectively. For uncorrelated driving times, in all four cases of (3.1), the response time is the minimum of two independent Erlang distributed random variables. The same holds for cases 3 and 4 of (3.2), for correlated driving times.

The most challenging setting to compute is case 2 of (3.2), where two trucks are dispatched to node j from different locations i_1 and i_2 under correlated driving times. This may be rewritten as the sum of an independent Erlang-distributed random variable and the minimum of two others, i.e.,

$$R(\boldsymbol{a}, j) = \sum_{e \in s(i_1,j) \cap s(i_2,j)} X_e + \min \left\{ \sum_{e \in s(i_1,j) \setminus s(i_2,j)} X_e, \sum_{e \in s(i_2,j) \setminus s(i_1,j)} X_e \right\}, \tag{3.4}$$

where $a_{i_1} = a_{i_2} = 1,\ i_1 \neq i_2$. This kind of driving time correlation captures the fact that two fire trucks that take the same route may be delayed by the same incident or traffic, and encourages dispatching trucks over non-overlapping routes.

Thus, in order to compute the immediate costs $\mathbb{P}(R(\boldsymbol{a}, j) > t^*)$, we require the following result.

Proposition 3.3.1. *Let $Y_0 \sim Er(1, w_0)$, $Y_1 \sim Er(1, w_1)$ and $Y_2 \sim Er(1, w_2)$ be independent Erlang-distributed random variables with phases of unit mean, $w_i > 0$, $i = 1, 2, 3$. Then*

$$P(\min\{Y_1, Y_2\} > t^*) = e^{-2t^*} \sum_{n=0}^{w_1-1} \sum_{m=0}^{w_2-1} \frac{t^{*n+m}}{n!m!}$$

and

$$\begin{aligned} P(Y_0 + \min\{Y_1, Y_2\} > t^*) = &\sum_{n=0}^{w_1-1} \sum_{m=0}^{w_2-1} \sum_{l=0}^{n+m} \frac{e^{-2t^*} t^{*l} (-1)^{n+m-l}}{n!m!(w_0-1)!} \binom{n+m}{l} \\ &\times \int_{y_0=0}^{t^*} y_0^{n+m-l+w_0-1} e^{y_0} \mathrm{d}y_0 \\ &+ \sum_{n=0}^{w_0-1} \frac{t^{*n}}{n!} e^{-t^*}. \end{aligned}$$

The proof of Proposition 3.3.1 can be found in Appendix 3.7.

3.3.3 Closest-First Dispatching

The main benchmark throughout this chapter is the current practice of FDAA, which is to always send the two closest (in terms of expected travel time) fire trucks, which we refer to as the *closest-first* (CF) policy. We consider this as part of a larger class of static dispatching policies, where fire trucks are dispatched according to a fixed order per demand location. Such policy can be represented by a list $\sigma_j(k)$, $j \in \mathcal{J}, k \in \{1, \ldots, \sum_i C_i\}$, where $\sigma_j(k) \in \mathcal{I}$ represents the fire station from which to send the kth truck for an incident at location j. Let $\boldsymbol{a}^{CF}(\boldsymbol{f}, j)$ denote the action taken in state $\boldsymbol{f}$ given a new incident at location j, then

$$\boldsymbol{a}^{CF}(\boldsymbol{f}, j) = \boldsymbol{e}_{\sigma_j(k_1)} + \boldsymbol{e}_{\sigma_j(k_2)},$$

where

$$k_1 = \min\{k : f_{\sigma_j(k)} \geq 1\}, \quad k_2 = \min\{k : f_{\sigma_j(k)} - \mathrm{I}_{\{k=k_1\}} \geq 1\},$$

denote the number of the first and second truck dispatched, respectively. That is, truck k_1 is the closest fire truck to demand location j that is available, and k_2 the second-closest. If $C_i = 1$ for all i, then σ_j reduces to a permutation over all fire stations. In case $k_i, i = 1, 2$ does not exist (because there are insufficient trucks available), we set $\sigma_j(k_i) = 0$ and

define $\boldsymbol{e}_0$ as the all-zero vector, to ensure trucks are sent from outside.

The long-term average costs under this CF policy can be obtained by limiting the Bellman equations (3.3) to only those actions $\boldsymbol{a}^{CF}(\boldsymbol{f}, j)$, i.e.,

$$\begin{aligned} h^{CF}(\boldsymbol{f})\tau = &- g^{CF} + \mu \sum_{i \in \mathcal{I}} (C_i - f_i) h^{CF}(\boldsymbol{f} + \boldsymbol{e}_i) + \mu \sum_{i \in \mathcal{I}} f_i h^{CF}(\boldsymbol{f}) \\ &+ \sum_{j \in \mathcal{J}} \lambda_j \left(\mathbb{P}(R(\boldsymbol{a}^{CF}(\boldsymbol{f}, j), j) > t^*) + h^{CF}(\boldsymbol{f} - \boldsymbol{a}^{CF}(\boldsymbol{f}, j)) \right), \quad \boldsymbol{f} \in \mathcal{S}. \end{aligned} \tag{3.5}$$

Here, g^{CF} and $h^{CF}(\boldsymbol{f})$ denote the long-term average and relative costs under the CF policy, respectively. Thus, (3.5) is a system of $|S|$ linear equations, with $|S|+1$ unknowns g^{CF} and $h^{CF}(\boldsymbol{f})$, $f \in \mathcal{S}$. The costs can be obtained by fixing $h^{CF}(\boldsymbol{f})$ for one state $\boldsymbol{f}$, and solving the remaining system of equations.

3.4 Dispatching Heuristics

As we shall see from the experiments in Section 3.5.2, the optimal dispatching policy significantly outperforms closest-first, both in the correlated and uncorrelated cases. However, it is well-known that solving the Bellman equations (3.3) can be computationally infeasible for large instances. In this section, we present two heuristics to approximate the optimal dispatching policy.

3.4.1 The One-Step Improvement Heuristic

The first heuristic we consider is based on the idea of one-step improvement, and we refer to the policy obtained this way as to the *one-step improvement* (OSI) policy. This approach was developed in [80, 82], and the key idea is to first determine the (relative) costs $\tilde{h}(\boldsymbol{y})$ for some sub-optimal policy, and then applying a single policy iteration step to find improved actions. That is, we replace the future costs $h^*(\boldsymbol{y})$ in (3.3) by some function $\tilde{h}(\boldsymbol{y})$. The maximizing action for this approximation of the Bellman equations can then be determined without iteration, significantly reducing the computational complexity compared to the full policy iteration algorithm. As pointed out in [80, 82], the first policy iteration step typically yields the biggest gains, so the result from one-step improvement is often close to optimal.

Here we use the CF policy to approximate the future optimal relative costs. We first compute the relative costs $h^{CF}(\boldsymbol{f})$ from (3.5), and then substitute these into the

right-hand side of the Bellman equations (3.3). Ignoring the part that does not depend on the actions, the decision made by the OSI policy can be found as:

$$\boldsymbol{a}^{OSI}(\boldsymbol{f}, j) \in \arg\min_{\boldsymbol{a} \in A(\boldsymbol{f})} \left(\mathbb{P}(R(\boldsymbol{a}, j) > t^*) + h^{CF}(\boldsymbol{f} - \boldsymbol{a}) \right), \quad \boldsymbol{f} \in \mathcal{S}. \tag{3.6}$$

3.4.2 The One-Step Improvement Approximation Heuristic

To derive the OSI policy from (3.6), we first need to solve the CF policy's Bellman equations (3.5) to determine the $h^{CF}(\boldsymbol{f})$. This is still computationally expensive for large problem instances. In this section, we present an algorithm that approximates the CF policy costs $h^{CF}(\boldsymbol{f})$, which can then in turn be used as a basis for the one-step improvement in (3.6). We will refer to the policy obtained using one step improvement with the CF policy cost approximation as the *one-step improvement approximation* (OSIA). This constitutes our second heuristic.

In order to approximate $h^{CF}(\boldsymbol{f})$, we assume that each fire station has exactly one truck. This assumption does not limit the applicability of the algorithm, as we can always treat each truck as a separate station in the same location, and adjust the states and actions accordingly. Let $J(\boldsymbol{f}, t)$ denote the expected total cost under the CF policy during the time interval $[0, t]$ starting from state $\boldsymbol{f}$. Then the relative cost $h^{CF}(\boldsymbol{f})$ can be defined as

$$h^{CF}(\boldsymbol{f}) = \lim_{t \to \infty} \left(J(\boldsymbol{f}, t) - g^{CF} t \right),$$

where g^{CF} denotes the cost per time unit under CF from (3.5).

Assume that after some time $T > 0$ the system is in steady state, so the difference between the relative costs and the average costs is incurred in the interval $[0, T]$ only. In this case, we can approximate $h^{CF}(\boldsymbol{f})$ as

$$\begin{aligned} h^{CF}(\boldsymbol{f}) = &\lim_{t \to \infty} J(\boldsymbol{f}, t) - g^{CF} t = J(\boldsymbol{f}, T) - g^{CF} T \\ &+ \lim_{t \to \infty} \left(J(\boldsymbol{f}, t) - J(\boldsymbol{f}, T) \right) - \left(g^{CF} t - g^{CF} T \right) \\ \approx &J(\boldsymbol{f}, T) - g^{CF} T. \end{aligned} \tag{3.7}$$

Substituting (3.7) into (3.6) we obtain the equations for the OSIA policy:

$$\begin{aligned} \boldsymbol{a}^{OSIA}(\boldsymbol{f}, j) &\in \arg\min_{\boldsymbol{a} \in A(\boldsymbol{f})} \mathbb{P}(R(\boldsymbol{a}, j) > t^*) + J(\boldsymbol{f} - \boldsymbol{a}, T) - g^{CF} T \\ &= \arg\min_{\boldsymbol{a} \in A(\boldsymbol{f})} \mathbb{P}(R(\boldsymbol{a}, j) > t^*) + J(\boldsymbol{f} - \boldsymbol{a}, T), \quad \boldsymbol{f} \in \mathcal{S},\ j \in \mathcal{J}. \end{aligned}$$

Here, we omit the $g^{CF}T$ term because it appears for all actions $\boldsymbol{a}$.

So in order to derive the OSIA policy, we need to estimate $J(\boldsymbol{f}, T)$, for all $\boldsymbol{f} \in S$, the total costs incurred in the interval $[0, T]$, starting from state $\boldsymbol{f}$. Following the approach taken in from [101], we decompose the network into individual $M/M/1/1$ queues associated with individual fire stations. By doing this, we essentially decouple the network into individual fire stations, for each we can now compute an approximation for the probability of the corresponding fire truck to be busy (the so-called *busy probability*). We combine these to obtain an approximation for $J(\boldsymbol{f}, T)$.

Let us first consider a fire station i in isolation, and compute its busy probability. Denote by D_i the given demand arrival rate for the truck at station i. Recall that the steady-state busy probability of an $M/M/1/1$ queue with load ρ_i is given by $B(\rho_i) = \rho_i/(1+\rho_i)$, and thus the steady-state rate of rejected requests is $D_i B(\rho_i)$. Denote by $N(\rho_i, f_i, t)$ the expected number of rejected requests in the $M/M/1/1$ queue during $[0, t]$ starting with f_i trucks at time 0. Finally, let $\Delta(\rho_i, f_i)$ be the difference in rejected requests between starting from steady state and starting from f_i: $\Delta(\rho_i, f_i) = \lim_{t\to\infty} \big(N(\rho_i, f_i, t) - D_i t B(\rho_i)\big)$.

Assuming as above that the system is in steady state after time T, we have that

$$\Delta(\rho_i, f_i) \approx \big(N(\rho_i, f_i, T) - D_i T B(\rho_i)\big). \tag{3.8}$$

The busy probability p_i can be obtained by dividing the expected total number of rejections $N(\rho_i, f_i, T)$ by the expected number of arrivals $D_i T$. Observe that in our case $\rho_i = D_i/\mu$, since each request will occupy the server (i.e., fire truck) for an expected duration μ^{-1}. Using the identity in (3.8) and bounding between 0 and 1 to obtain a probability (since we are using approximations), we get

$$p_i = \frac{N(\rho_i, f_i, T)}{D_i T} = \max\left\{0, \min\left\{1, B(D_i/\mu) + \frac{\Delta(D_i/\mu, f_i)}{D_i T}\right\}\right\}. \tag{3.9}$$

Observe that in order to evaluate (3.9) we need to approximate $\Delta(D_i/\mu, f_i)$, the difference in total number of rejected calls between steady-state and starting from state f_i. To do this, we formulate the queue representing station i as an average-cost MDP, where the state is the number of idle trucks at the fire station. Transitions happen when either a request for a truck arrives, or an idle truck returns from an incident. If there is an idle truck, it is always dispatched. The cost for a rejection is 1, and 0 for an accepted job. This results in the following system of two Bellman equations and a normalizing equation:

$$h_0 = \frac{D_i}{D_i + \mu} - \frac{D_i B(\frac{D_i}{\mu})}{D_i + \mu} + \frac{D_i}{D_i + \mu} h_0 + \frac{\mu}{D_i + \mu} h_1, \tag{3.10}$$

$$h_1 = -\frac{D_i B(\frac{D_i}{\mu})}{D_i} + h_0, \tag{3.11}$$

$$\frac{1}{1+\frac{D_i}{\mu}} h_0 + \frac{\frac{D_i}{\mu}}{1+\frac{D_i}{\mu}} h_1 = 0. \tag{3.12}$$

Solving (3.10)-(3.12), we obtain $\boldsymbol{h} = (h_0, h_1)$, the relative costs starting from state $f_i = 0$ or $f_i = 1$, respectively. We use $\Delta(D_i/\mu, f_i) = h_{f_i}$, and compute p_i using (3.9).

Having determined the busy probability p_i for a given arrival rate D_i, our next step is to update the values of D_i using the busy probabilities obtained. Here, we again consider all fire stations jointly. According to the CF policy, the closest two idle trucks are dispatched to an incident. Recall that the lists $\sigma_j(k)$, $j \in \mathcal{J}, k \in \{1, \ldots, I\}$, represent the dispatching order corresponding to the CF policy. So as each station has exactly one truck, $\sigma_j^{-1}(i)$ denotes the position held by station i in the dispatching order of demand location j. For instance, $\sigma_j^{-1}(i) = 1$ means that station i is the closest to demand location j.

Let p_0 correspond to the probability of an outside truck being unavailable, and set $p_0 = 0$. After p_i is computed for each station i according to (3.9), we calculate the probability $p^j_{\{i_1,i_2\}}$ of a newly arrived incident at demand location j requests trucks at i_1 and i_2. Note that a single incident can generate requests at multiple pairs of fire stations, since some of them might be occupied. By conditioning on the availability of the fire trucks we obtain:

For $j = 1, \ldots, J$, $i_1 = 2, \ldots, I$, $i_2 = 1, \ldots, (i_1 - 1)$ (both trucks are from inside):

$$p^j_{\{i_1,i_2\}} = \begin{cases} 1, & \text{if } \sigma_j(i_1) = 1,\ \sigma_j(i_2) = 2, \\ & \text{or } \sigma_j(i_1) = 2,\ \sigma_j(i_2) = 1, \\ \prod_{i \neq i_1, i_2,\ \sigma_j(i) < max\{\sigma_j(i_1), \sigma_j(i_2)\}} p_i, & \text{otherwise.} \end{cases} \tag{3.13}$$

For $j = 1, \ldots, J$, $i_1 = 1, \ldots, I$, $i_2 = 0$ (one truck is from outside):

$$p^j_{\{i_1,i_2\}} = \prod_{i \neq i_1} p_i. \tag{3.14}$$

For $j = 1, .., J$, $i_1 = 0$, $i_2 = 0$ (both trucks are from outside):

$$p^j_{\{i_1,i_2\}} = \prod_{i=1}^{I} p_i. \tag{3.15}$$

Algorithm 2 CF cost approximation

Initialization

$p^j_{\{i_1,i_2\}} = \begin{cases} 1, & \text{if } \sigma_j(i_1) = 1,\ \sigma_j(i_2) = 2 \text{ or } \sigma_j(i_1) = 2,\ \sigma_j(i_2) = 1 \\ 0, & \text{otherwise} \end{cases}$

$D_{\{i_1,i_2\}} = \sum_{j\in\mathcal{J}} \lambda_j p^j_{\{i_1,i_2\}} \quad \forall i_1, i_2 \in \{0, 1, ..., I\}$

$D_i = \sum_{k\neq i} D_{\{i,k\}} \quad \forall i \in \mathcal{I}$

while true **do**

  Compute $\Delta(D_i, \mu, f_i) = h_{f_i}$ using (3.10)-(3.12)

  Compute p_i using (3.9)

  Compute $p^j_{\{i_1,i_2\}}$ using (3.13)-(3.14)

  Compute $D_{\{i_1,i_2\}}$ using (3.16)

  $\hat{D}_i = \sum_{k\neq i} D_{\{i,k\}} \quad \forall i \in \mathcal{I}$

  if $|D_i - \hat{D}_i|/D_i < \epsilon \quad \forall i \in \mathcal{I}$ **then**

    $D_i = \hat{D}_i \quad \forall i \in \mathcal{I}$

    break

  $D_i = \hat{D}_i \quad \forall i \in \mathcal{I}$

$J(\boldsymbol{f}, T) = T \sum_{j\in\mathcal{J}} \lambda_j \sum_{i_1=0}^{I} \sum_{i_2=0}^{\max\{0,i_1-1\}} p^j_{i_1 i_2} C^j_{i_1 i_2} (1 - p_{i_1})(1 - p_{i_2})$

The probability $p^j_{\{i_1,i_2\}}$ is equal to 1 if trucks at i_1 and i_2 are the closest to j. Otherwise, it is a product of the busy probabilities of those trucks that are closer than either i_1 or i_2. Trucks from inside of the region are always closer than those from outside.

Denote $D_{\{i_1,i_2\}}$ the demand arriving for trucks from stations i_1 and i_2. Given the probabilities $p^j_{\{i_1,i_2\}}$, we compute $D_{\{i_1,i_2\}}$ for $i_1 = 2, ..., I$, and $i_2 = 1, ..., (i_1 - 1)$:

$$D_{\{i_1,i_2\}} = \sum_{j\in\mathcal{J}} \lambda_j p^j_{\{i_1,i_2\}}. \tag{3.16}$$

Finally, by summing over all pairs $\{i, k\}$, $k \neq i$, we can obtain the arrival rate of incidents at station i as $D_i = \sum_{k\neq i} D_{\{i,k\}}$.

Let $C^j_{i_1 i_2}$ indicate the expected penalty related to sending trucks i_1 and i_2 to location j. It is equivalent to the cost $\mathbb{P}(R(\boldsymbol{a}, j) > t^*)$ where the action $\boldsymbol{a}$ corresponds to sending the trucks from stations i_1 and i_2 to location j, given that those are idle. Computing these costs is discussed in Section 3.3.2. We now summarize the algorithm that approximates $J(\boldsymbol{f}, T)$ for a given state $\boldsymbol{f} \in \mathcal{S}$ in pseudocode Algorithm 2.

3.5 Numerical Results

We now present the results of our numerical experiments. In Section 3.5.1 we describe the setup of our numerical experiments. The results are separated into two parts: in Section 3.5.2 we compare the CF and OPT policies, and use this to understand how much improvement over CF can be obtained, and what is the impact of driving-time correlation on the policies and their performance. In Section 3.5.3, we then evaluate the performance of our heuristics OSI and OSIA relative to CF and OPT, both in terms of fraction of late arrivals and computation time.

3.5.1 Experimental Setup

All experiments were run in MATLAB R2017b on a computer with an Intel Core i5-5250U 1.6 GHz processor, 8 GB RAM, running Linux Fedora 26. In order to evaluate the performance of a policy for a given network and set of parameters, we numerically solve the Bellman equations (3.3) for OPT policy and the restricted Bellman equations (3.5) for CF policy. This way we obtain g^{OPT} and g^{CF}, the long-term expected number of late arrivals per time unit for OPT and CF, respectively. The dispatching order $\sigma_j(k)$ for CF is determined by ordering for each demand location j the fire stations k based on the length of their shortest path to j. Ties are broken arbitrarily.

In order to compute the performance of OSI we first determine the relative costs for closest first $h^{CF}(\boldsymbol{f})$ from (3.5), and substitute these into (3.6) to determine the actions $\boldsymbol{a}^{OSI}$. These are then substituted into the Bellman equations (3.3), which we solve numerically to obtain the rate of late arrivals for OSI, g^{OSI}. For OSIA, we repeat this procedure, except that instead of computing the exact relative costs for closest first $h^{CF}(\boldsymbol{f})$, we compute $J(\boldsymbol{f}, T)$ from Algorithm 2 and use the approximation for $h^{CF}(\boldsymbol{f})$ from (3.7). This way, we obtain g^{OSIA}, the rate of late arrivals under OSIA. In order to compute the fraction of late arrivals (FLAR) for any of these policies, we divide the long-term expected number of late arrivals per time unit g by the total arrival rate, i.e., $g/\sum_{j\in\mathcal{J}}\lambda_j$.

For our experiments, we randomly generate grid-like graphs, as outlined below. For some parameter $d \in \mathbb{N}$, we generate a grid of $d \times d$ vertices (see Figure 3.2a), placed at unit distance. We then connect each pair of vertices within unit distance from each other, so a vertex away from the boundary is connected to its four immediate neighbors (see Figure 3.2b) and we obtain a graph with $|\mathcal{J}| = d^2$ nodes and $|E| = 2d(d-1)$ edges. We then remove edges uniformly at random, until the number of removed edges is below $2d(d-1)s$ (see Figure 3.2c), where $s \in (0,1)$ is some desired level of sparseness. The s parameter is drawn from a uniform distribution $\mathcal{U}(0.4, 1)$. While removing the edges, we

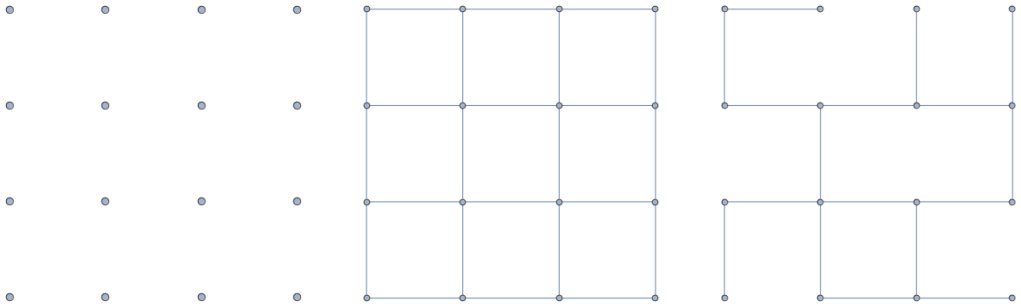


Figure 3.2: Random graph construction

check if the graph remains connected. In case the graph becomes disconnected, a new random edge is selected for removal. If after a certain number of attempts no edge is found that can be removed without disconnecting the graph, the procedure stops, and the obtained graph is used.

In our experiments, we assume each station has exactly one truck. This does not affect methodology, but makes it easier to visualise and understand the difference in actions taken by different policies. We allocate stations (or trucks) to vertices sequentially in a randomized manner. Each of the I trucks is positioned on a vertex not yet occupied by other trucks uniformly at random.

3.5.2 Comparison of Closest-First and Optimal Dispatching

In this section we study OPT and its performance relative to the CF heuristic. Recall that OPT is computed from the Bellman equations (3.3) through policy iteration, and it is here that we run into the infamous curse of dimensionality, which states that the state space and action space of the MDP become too big to solve in an efficient manner. Specifically, our action space grows as $\mathcal{O}(I^2)$ since each action consists of sending two trucks. The state space grows as $\mathcal{O}(2^I \times d^2)$, since there are 2^I possible combinations of available trucks, and the next fire can occur on any of the $J = d^2$ demand locations. Although the complexity of each step of policy iteration is polynomial in the size of the state space and action space, there is no universal polynomial bound on the complexity of the algorithm, due to the uncertainty in the number of steps required [68]. In practical terms, this means that we can only compute the optimal policy for instances of small-to-moderate size. In Section 3.5.3 we restrict ourselves to suboptimal policies, and consider instances of real-life size (in the case of FDAA, there are roughly $I = 13$ trucks and $J \approx 400$ demand

locations). Due to the relatively low load seen in the FDAA practice ($\rho = 0.02$) and used in our experiments, the number of incidents that requires trucks from outside is negligible.

Relative improvement of the optimal policy over closest-first

We are interested in assessing the current practice of dispatching the two closest trucks, and to see whether there is any room for improvement (i.e., reducing the fraction of late arrivals) by dispatching in a smarter way. To do this, we consider the relative improvement of OPT over CF, which is computed as

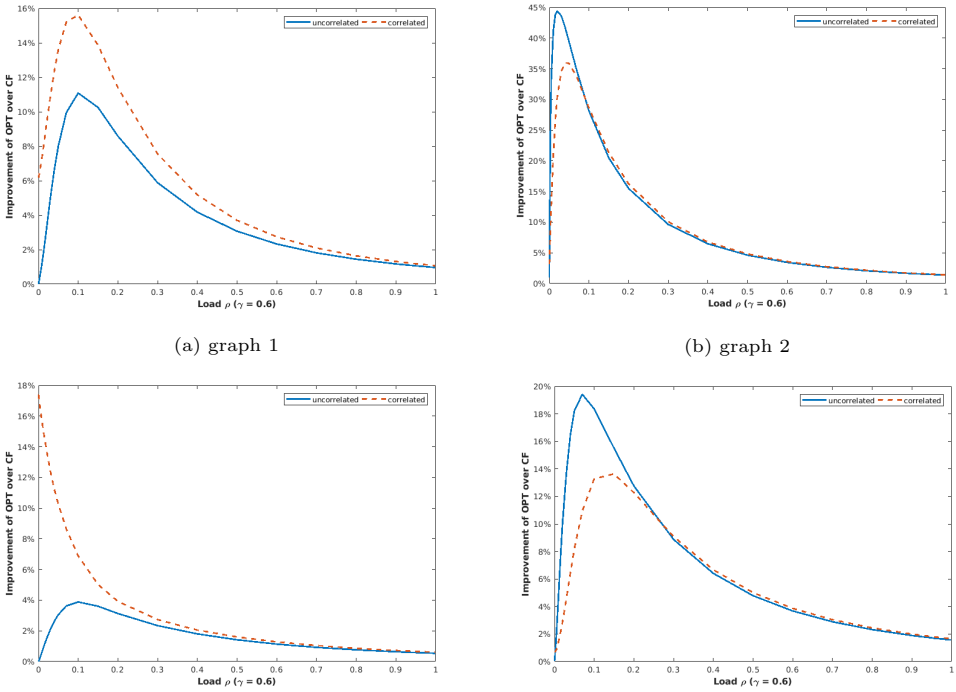
$$\delta^{OPT} = \frac{g^{CF} - g^{OPT}}{g^{CF}} \times 100\%.$$

In Figure 3.3 we plot the relative improvement against the load of the system $\rho = \frac{\sum_{j \in \mathcal{J}} \lambda_j}{I\mu}$, which represents the amount of work per truck arriving each time unit. We do this for four different randomly generated graphs, and show the improvement both in uncorrelated and correlated cases. We choose the time threshold for late arrivals as $t^* = \gamma \max_{i \in \mathcal{I}, j \in \mathcal{J}} |s(i,j)|$, to ensure that it scales with the graph size, and set $\gamma = 0.6$.

We see that in both cases the relative improvement ranges from 0% to 50%, depending on the load and on the graph. This suggests that in the right circumstances, significant gains can be found by dispatching in a clever way. In the uncorrelated case, the relative difference is small when ρ is small or large. This is because if the load is close to 0, the system is almost always in the state with all the trucks being idle, and when ρ is close to 1, there is no room for improvement irrespective of whether there is correlation or not, because the system is almost always in the state with no idle trucks.

When correlation is introduced, however, we see from Figure 3.3 that sending two closest trucks does not necessarily minimize response time, even for small loads. Hence, in this case the OPT policy may improve upon the CF policy even for very small values of ρ, as illustrated in Figures 3.3c and 3.3d. However, we see in all cases in Figure 3.3 that as ρ grows, the improvement curve for correlated driving times converges to the one corresponding to uncorrelated case.

The influence of the time threshold t^* (through the parameter γ) is studied in Figure 3.4. Four arbitrary random graphs are chosen, and for each the relative improvement is plotted against γ, with $\rho = 0.1$. We again observe that significant gains can be made compared to the closest-first policy, and that the extent of this improvement depends on the network parameters. Here we can see that the behaviour is similar in both the correlated and uncorrelated cases. If γ is close to zero (and hence, so is t^*), the OPT



(a) graph 1

(b) graph 2

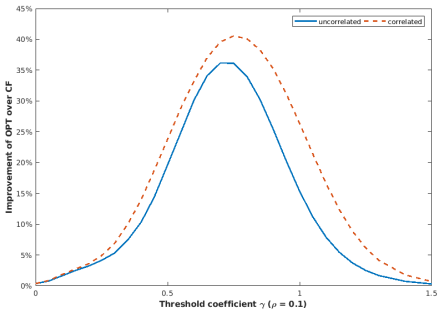
(c) graph 3

(d) graph 4

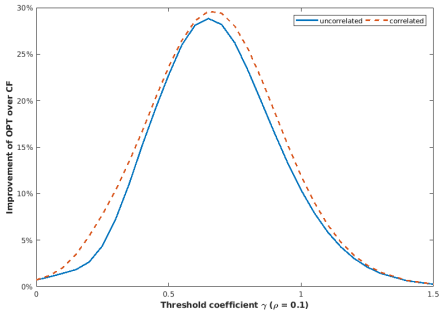
Figure 3.3: δ^{OPT} as a function of ρ for four random graphs ($I = 5$, $d = 7$, $\gamma = 0.6$)

policy cannot improve upon the CF policy. The time threshold is too low to meet, unless the location of a fire coincides with the location of one of the idle trucks. As a result, the fraction of late arrivals is close to 1 independent of which trucks are sent. As γ grows, there is more room for improvement. However, when γ approaches 1, the relative improvement of OPT drops to zero again. The reason is that in this case the time threshold t^* is so large it can always be met, even if the dispatching policy is far from optimal.

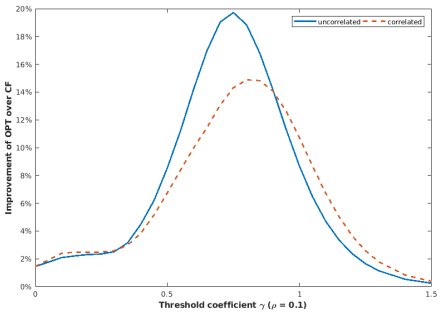
For a more detailed review of the relative improvement of OPT over CF we turn to Table 3.1. This shows the relative improvements a function of the graph size parameter d and the number of trucks I, for $\rho = 0.1$ and $\gamma = 0.6$. For every combination of I and d, we generate 150 random graphs. The values in Table 3.1 represent the minimum, mean and maximum over these 150 random graphs for each parameter set. We can see a modest



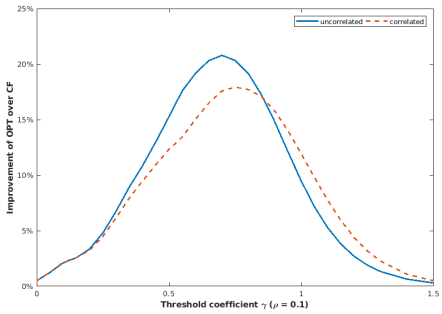
(a) graph 1



(b) graph 2



(c) graph 3



(d) graph 4

Figure 3.4: δ^{OPT} as a function of γ for four random graphs ($I = 5$, $d = 7$, $\rho = 0.1$)

increase in relative improvement in d, and a significant improvement in I, reaching an average improvement of over 20% with $I = 6$ trucks, and over 50% for certain instances with driving time correlation.

In Figure 3.5 we show the fraction of late arrivals for OPT for the same set of experiments discussed above. That is, for different values of d and I we plot the confidence interval over all 150 graphs considered. Although we observed from Table 3.1 that the average relative improvement of OPT over CF is not significantly affected by whether we consider driving-time correlation, we see from Figure 3.5 that the fraction of late arrivals increases when correlation is taken into account. This indicates that in this case it is more important to deviate from the CF policy in order to limit the fraction of late arrivals. Since in practice there is always some degree of driving-time correlation, these

Table 3.1: Minimum, maximum and mean δ^{OPT} evaluated over 150 random graphs ($\rho = 0.1$, $\gamma = 0.6$)

		uncorrelated			correlated		
I	***d***	**min**	**mean**	**max**	**min**	**mean**	**max**
3	**4**	0.00%	5.50%	20.49%	0.00%	5.77%	19.69%
	5	0.00%	7.41%	25.08%	0.00%	7.46%	25.16%
	6	0.01%	6.70%	24.79%	0.00%	6.60%	26.56%
	7	0.03%	7.09%	32.43%	0.00%	7.06%	34.51%
4	**4**	0.15%	9.61%	25.66%	0.04%	11.36%	37.27%
	5	0.99%	10.28%	34.56%	0.92%	11.39%	42.50%
	6	1.10%	11.02%	31.16%	1.18%	12.32%	37.11%
	7	1.38%	11.32%	39.17%	1.15%	12.19%	42.49%
5	**4**	2.24%	16.03%	46.65%	2.53%	18.58%	50.59%
	5	2.25%	16.62%	40.96%	2.29%	17.91%	43.17%
	6	2.36%	16.84%	37.36%	3.49%	17.94%	39.77%
	7	2.68%	19.72%	52.42%	1.63%	20.22%	54.95%
6	**4**	4.94%	20.70%	46.21%	5.13%	23.35%	51.01%
	5	7.12%	21.79%	43.17%	6.10%	24.08%	48.80%
	6	4.03%	22.75%	49.18%	4.68%	24.63%	52.34%
	7	6.04%	24.84%	53.57%	5.40%	26.48%	54.60%

results suggest that when dispatching multiple trucks it is valuable to deviate from CF dispatching. This is in contrast to the case with a single truck, when CF is close to optimal [48].

Impact of correlation on the optimal policy.

To illustrate the difference between the optimal policies without correlation ($\boldsymbol{a}_{uc}^{OPT}$) and with correlation ($\boldsymbol{a}_{c}^{OPT}$) we select a random graph with $d = 6$ ($J = 36$ demand locations) and $I = 4$ trucks, see Figure 3.6. The demand locations are colored according to the arrival rates of new incidents, with green corresponding to low rates. We are looking at the state $\boldsymbol{f} = \boldsymbol{C}$ with all four trucks available. The background of each location j is colored according to the corresponding policy $\boldsymbol{a}^{OPT}(\boldsymbol{C}, j)$. For example, if a new incident happens at a demand location with green background, then trucks 1 and 2 will be dispatched.

While for this particular choice of graph and parameters the impact of correlation is relatively small, it is useful for illustrating how the optimal policy changes when

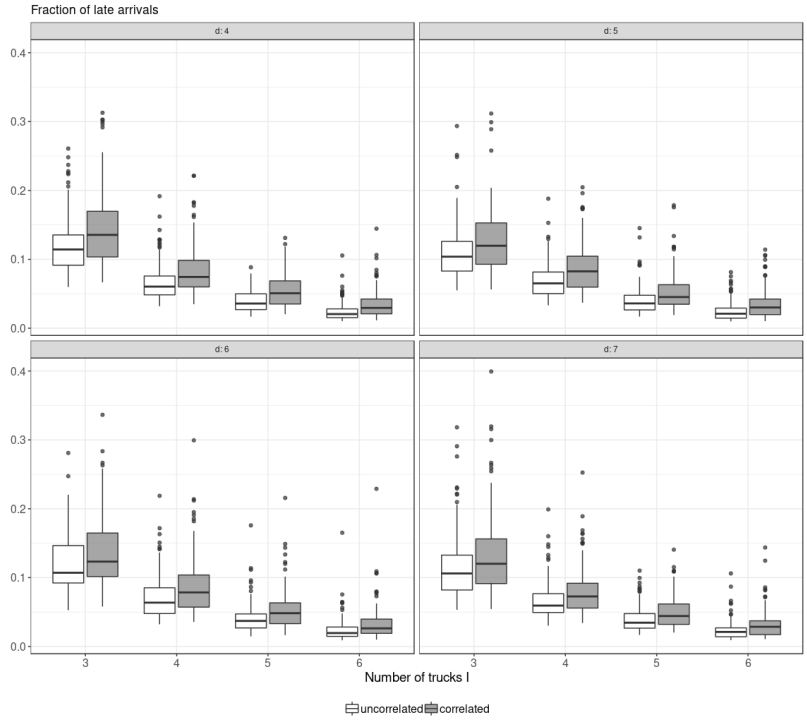
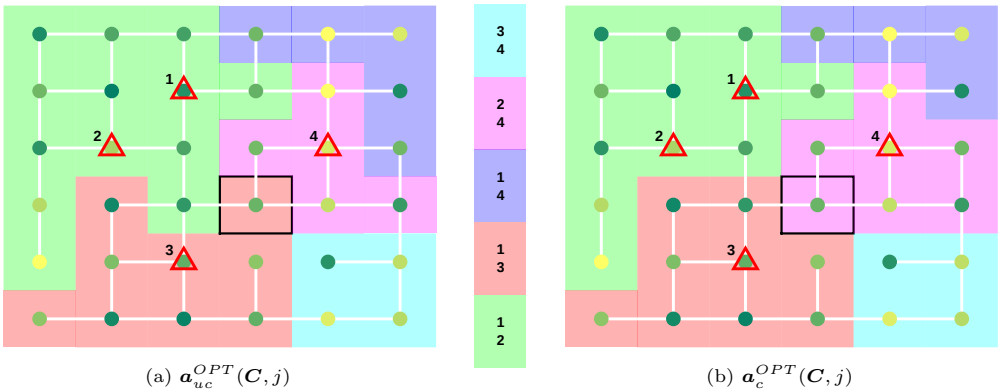


Figure 3.5: Confidence intervals for fraction of late arrivals of the OPT policy for different values of d ($\rho = 0.1$, $\gamma = 0.6$)

correlation is introduced. For instance, to the demand location highlighted in black in the middle of the graph, the policy $\boldsymbol{a}_{uc}^{OPT}$ dispatches trucks 1 and 3 that share one edge on their way to that location. The policy $\boldsymbol{a}_{c}^{OPT}$ instead dispatches trucks 2 and 4 that share no edges in their shortest paths, as shared edges imply higher probability of being late in the presence of driving-time correlation.

The other two changes in this example, as well as those in other instances we evaluated, follow a similar pattern: the optimal policy with correlation may be different from the optimal policy without correlation for those demand locations where $\boldsymbol{a}_{uc}^{OPT}$ dispatches two trucks with overlapping routes. However, this need not be the case, and the example in Figure 3.6 also includes such demand locations where $\boldsymbol{a}_{c}^{OPT}$ remains unchanged compared to $\boldsymbol{a}_{uc}^{OPT}$, because in these cases the decrease in expected response time when changing actions does not outweigh the coverage reduction resulting from this change. This illus-



(a) $\boldsymbol{a}_{uc}^{OPT}(\boldsymbol{C}, j)$ (b) $\boldsymbol{a}_{c}^{OPT}(\boldsymbol{C}, j)$

Figure 3.6: Example of difference between $\boldsymbol{a}_{uc}^{OPT}(\boldsymbol{f}, j)$ and $\boldsymbol{a}_{c}^{OPT}(\boldsymbol{f}, j)$ on a random graph

trates the complexity of finding the optimal policy for this model, and the difficulties one would encounter when trying to generalize the observations obtained from Figure 3.6 into some kind of heuristic. One main reason for this is the complex interactions encountered in this model. For instance, changing the arrival rate in one part of the network may affect the optimal policy elsewhere.

To see the extent to which driving-time correlation affects the optimal policy for a broader range of instances we conduct the following experiment. We generate 150 random graphs, and for every graph we compute $\boldsymbol{a}_{c}^{OPT}$ and $\boldsymbol{a}_{uc}^{OPT}$. In order to study the impact of ignoring driving-time correlation, we look at what happens with the system performance if we use $\boldsymbol{a}_{uc}^{OPT}$ in a setting with driving-time correlation. To do this, we substitute the policy $\boldsymbol{a}_{uc}^{OPT}$ into the Bellman equations (3.5) for a fixed policy with the costs corresponding to the correlated case, and measure the relative increase in value function compared to the policy $\boldsymbol{a}_{c}^{OPT}(\boldsymbol{f}, j)$. Note that the relative increase in value function is equivalent to the relative increase in the fraction of late arrivals.

Table 3.2 shows the aggregate results of this experiment with minimum, maximum and mean relative increase in fraction of late arrivals computed over 150 random graphs. We observe that the importance of taking driving-time correlation into account grows with the number of trucks in the system. With more vehicles available there are more options for making a dispatching decision to avoid potential traffic jams for the current and upcoming incidents. The average decrease in performance when using the policy derived under the assumption of uncorrelated driving times in a setting with driving-time

Table 3.2: Relative increase in fraction of late arrivals when ignoring correlation ($d = 6$, $\rho = 0.1$, $\gamma = 0.6$)

I	min	mean	max
3	0.0%	1.3%	7.4%
4	0.2%	2.8%	12.2%
5	0.3%	4.8%	16.3%
6	1.1%	7.1%	21.3%

correlation reached 7.1% for 6 trucks, and for some instances was over 20%.

3.5.3 Performance of the Heuristics

In this section, we compare the performance of the two heuristics OSI and OSIA to the optimal policy OPT, both in terms of fraction of late arrivals and computation time.

Improvement over closest-first

First, we look at the heuristics' performance. Table 3.3 shows the relative difference of OSI and OSIA with CF, in addition to that of OPT. The values of δ^{OSI} and δ^{OSIA} are computed the same way as δ^{OPT}. The numbers presented in the table are the mean values of the corresponding metrics evaluated over 150 randomly generated graphs. The values of d and γ are fixed, and we vary the load ρ in the range $\{0.02, 0.04, 0.1, 0.4, 0.6\}$ and the number of trucks I in the range 3 to 6. For every combination of I and ρ the minimum, mean and maximum over 150 randomly generated graphs is presented. The improvement over CF for all three policies first increases with ρ followed by a decrease for high loads. Both OSI and OSIA policies show significant improvement over the CF policy for lower values of ρ, and are relatively close to the performance of OPT. The improvement over CF grows with I and is larger in the presence of driving-time correlation, similar to what we observed in Table 3.1. As it can be seen from the more detailed Tables 3.7, 3.6 and 3.8 in Appendix 3.7, the heuristics performance also improves as d increases, suggesting that their performance is better for larger networks. Appendix 3.7 also includes Table 3.9, which shows the relative improvement of OSIA over CF for $\rho = 0.02$ and $I = 7$ for larger values of d. Here we see that as I and d grow larger, the gap with CF increases as well.

Note that, in our setting, the fraction of late arrivals under the CF policy $FLAR^{CF}$ is relatively low. This would mean that the improvement over CF in the number of late arrivals is low compared to the total number of incidents. However, this improvement should not be understated. From the emergency services perspective, any improvement in late arrivals is considered significant. In the case of FDAA, for example, the original

Table 3.3: Aggregate performance evaluated over multiple random graphs ($d = 6$, $\gamma = 0.6$)

		uncorrelated				correlated			
I	ρ	$FLAR^{CF}$	δ^{OPT}	δ^{OSI}	δ^{OSIA}	$FLAR^{CF}$	δ^{OPT}	δ^{OSI}	δ^{OSIA}
3	**0.02**	0.39%	4.83%	4.83%	2.81%	0.51%	5.54%	5.54%	4.12%
	0.04	0.48%	6.36%	6.36%	4.88%	0.60%	6.52%	6.52%	5.43%
	0.1	0.77%	6.70%	6.70%	6.36%	0.89%	6.60%	6.60%	6.33%
	0.4	2.12%	2.57%	2.57%	2.51%	2.19%	2.59%	2.59%	2.52%
	0.6	2.74%	1.48%	1.48%	1.45%	2.79%	1.50%	1.50%	1.47%
4	**0.02**	0.20%	9.49%	9.49%	5.10%	0.31%	12.24%	12.24%	9.53%
	0.04	0.25%	11.55%	11.54%	8.51%	0.37%	13.53%	13.52%	11.43%
	0.1	0.47%	11.02%	10.99%	10.45%	0.59%	12.32%	12.29%	11.87%
	0.4	1.82%	3.77%	3.75%	3.28%	1.90%	4.13%	4.12%	3.50%
	0.6	2.50%	2.14%	2.14%	1.94%	2.56%	2.33%	2.32%	2.05%
5	**0.02**	0.10%	15.71%	15.62%	8.56%	0.18%	17.38%	17.30%	13.50%
	0.04	0.14%	18.57%	18.35%	13.60%	0.21%	19.56%	19.32%	16.22%
	0.1	0.29%	16.84%	16.49%	14.35%	0.38%	17.94%	17.58%	16.00%
	0.4	1.60%	4.76%	4.68%	3.90%	1.68%	5.22%	5.12%	4.06%
	0.6	2.34%	2.56%	2.53%	2.21%	2.40%	2.79%	2.75%	2.32%
6	**0.02**	0.05%	20.45%	20.15%	11.73%	0.11%	22.00%	21.70%	17.33%
	0.04	0.07%	24.90%	24.37%	17.89%	0.13%	25.91%	25.30%	21.64%
	0.1	0.18%	22.75%	22.05%	18.44%	0.25%	24.63%	23.75%	21.57%
	0.4	1.43%	5.99%	5.84%	4.92%	1.50%	6.67%	6.49%	5.17%
	0.6	2.20%	3.16%	3.10%	2.70%	2.26%	3.47%	3.40%	2.87%

idea of dispatching two trucks instead of one is targeted at reducing the risk of a possible delay, despite additional operational costs. This shows the importance of any decrease in response time. The further gains that can be achieved by changing the dispatching strategy are particularly valuable, given that it does not involve any extra operational costs.

Given the value functions g^{OPT} and g^{OSIA} of the OPT and OSIA policies, respectively, we compute the *OSIA optimality gap* as

$$\frac{g^{OSIA} - g^{OPT}}{g^{OPT}} \times 100\%.$$

We compute the optimality gap for OSI in a similar way. Table 3.4 shows the average optimality gap of the OSI and OSIA policies computed over 150 random graphs for each

Table 3.4: Average optimality gap of OSI and OSIA ($d = 6$, $\gamma = 0.6$)

		uncorrelated		correlated	
I	ρ	OSI	OSIA	OSI	OSIA
3	**0.02**	0.00%	2.36%	0.00%	1.68%
	0.04	0.00%	1.72%	0.00%	1.24%
	0.1	0.00%	0.37%	0.00%	0.29%
	0.4	0.00%	0.06%	0.00%	0.07%
	0.6	0.00%	0.03%	0.00%	0.03%
4	**0.02**	0.00%	6.01%	0.01%	3.57%
	0.04	0.01%	3.91%	0.01%	2.70%
	0.1	0.03%	0.66%	0.04%	0.52%
	0.4	0.01%	0.51%	0.02%	0.66%
	0.6	0.01%	0.21%	0.01%	0.28%
5	**0.02**	0.18%	11.67%	0.16%	5.98%
	0.04	0.36%	7.24%	0.40%	4.79%
	0.1	0.47%	3.05%	0.49%	2.41%
	0.4	0.09%	0.90%	0.11%	1.22%
	0.6	0.04%	0.36%	0.04%	0.49%
6	**0.02**	0.67%	16.26%	0.58%	8.13%
	0.04	1.04%	11.72%	1.12%	7.04%
	0.1	1.01%	5.73%	1.33%	4.18%
	0.4	0.16%	1.13%	0.20%	1.60%
	0.6	0.06%	0.47%	0.07%	0.63%

combination of I and ρ. The performance of both OSI and OSIA stays within a few percent of OPT. The optimality gap grows with I. The OSI policy performs slightly better in a setting without correlation, while the opposite is true for OSIA. The optimality gap of both OSI and OSIA decreases in ρ, suggesting that these approximations perform best in the high load regime. Note that while the optimality gap of these heuristics grows in the network size, we have seen from Tables 3.3, 3.7, 3.6, 3.8 and 3.9 that the improvement over CF also does. So while neither OSI nor OSIA is asymptotically optimal, their performance in fact improves as the network grows larger.

Computation time

Next, we take a look at the computation time of the various policies. If computation time would not be an issue, then using OPT is an obvious choice. However, solving MDP exactly quickly becomes problematic when the instance size grows, as the size of the state space grows exponentially in I. In our experiments, the main issue with solving the MDP

exactly for larger instances was not the running time of policy iteration, but the size of the array with transition probabilities (i.e., $|\mathcal{S}| \times |\mathcal{S}| \times |\mathcal{A}|$). As a result, computing the OPT policy breaks down for even moderate-sized networks (e.g., $I = 7$, $d = 6$).

To compare the computational performance of OSI and OSIA, we plot the computation time for determining these policies against I (Figure 3.7) and the number of demand locations $J = d^2$ (Figure 3.8). Here, we use a single randomly generated graph for each data point. The OSI policy is computed faster then the optimal, but still requires solving a set of $|\mathcal{S}+1|$ Bellman equations. Storing a $|\mathcal{S}+1| \times |\mathcal{S}+1|$ matrix of coefficients for the system of Bellman equations becomes infeasible, which is why we can only determine the OSI policy for small values of I and J. The computation time of the OSIA policy shows significantly slower growth in I and J than that of OSI. Moreover, it does not require storing large data structures, and makes it feasible to obtain a good policy for problem instances of realistic size.

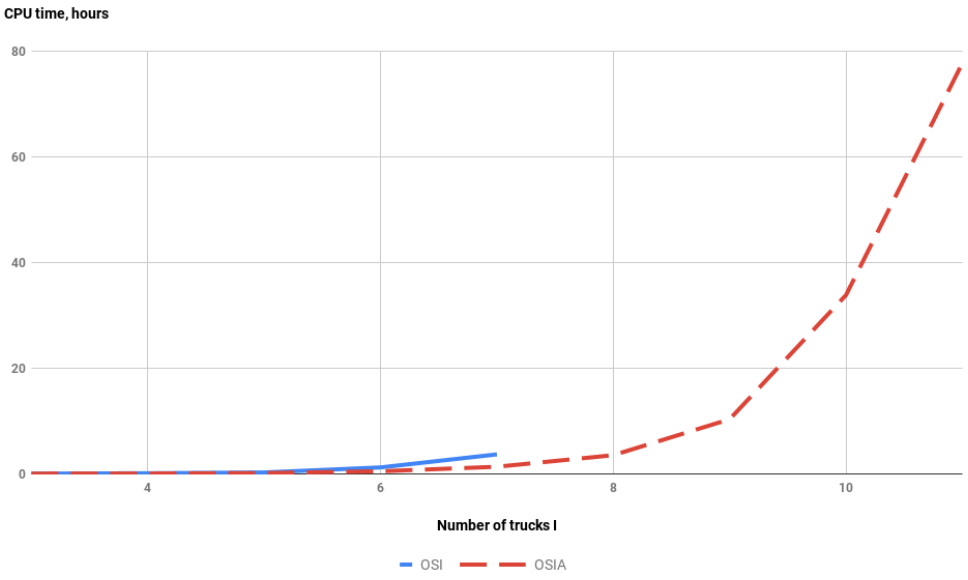


Figure 3.7: Computation time as a function of the number of trucks I ($J = 625$)

The computation time of the OSIA heuristic is reasonable for the systems used in our

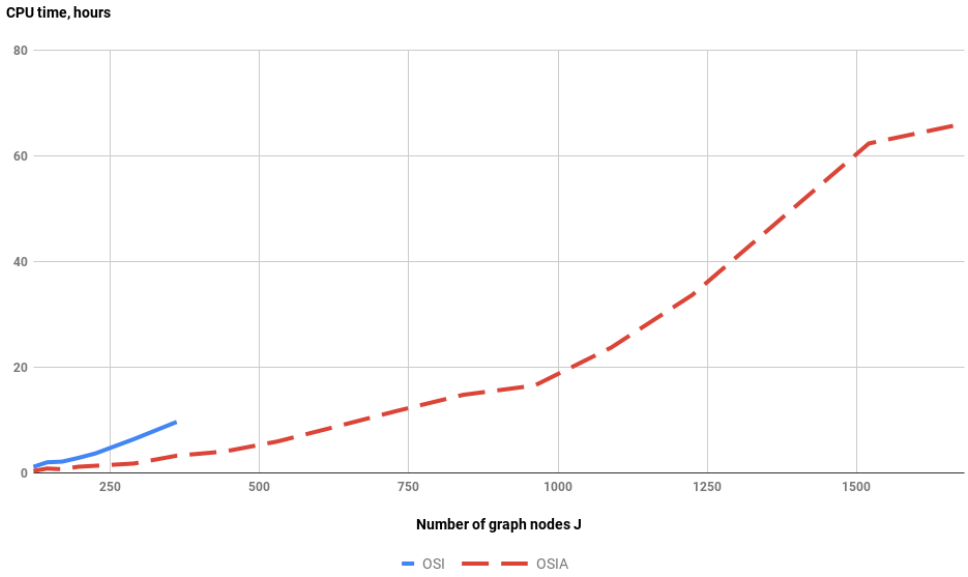


Figure 3.8: Computation time as a function of the number of demand locations J ($I = 7$)

numerical experiments. The algorithm is meant to be used in the offline regime, only once for a given network, and produces look-up tables indicating the dispatching decision to be made for each state of the system. Moreover, in our experiments we ran approximation Algorithm 2 sequentially for each state. In real-life applications, the OSIA computation time can be significantly decreased by means of parallelization.

3.6 Conclusion

In the present chapter we studied a dispatching problem in a fire department where two trucks have to be dispatched to an incident location, and the decision is to be made on which idle trucks to send. We modelled the region served by a fire department as a connected graph and formulated the dispatching problem as an MDP. The optimal policy was obtained by solving the MDP exactly using policy iteration.

Using small problem instances, we showed that the current practice of sending the two closest trucks can be far from optimal, with optimality gap reaching 50% in certain

cases. As obtaining the optimal policy for large problem instances is computationally infeasible, we also derived a one-step-improvement OSI policy, that can be obtained faster and for larger problem instances than OPT. In our experiments, however, OSI still remained computationally infeasible for problem instances of realistic sizes. Therefore, we introduced the OSIA policy that incorporates an approximation scheme into the OSI policy computation procedure. The OSIA policy performed close to the optimal performance with optimality gap of about 2%, and significantly lower computation time that allows for solving problem instances of realistic size.

We considered two types of stochastic behaviour in driving time when two trucks are dispatched to the same incident location. If two trucks traverse the same edge in a graph we assume their travelling times to be either independent of each other (uncorrelated), or the same (correlated). Our experiments show that introducing correlation makes a difference compared to sending two closest trucks, even if the load is small. Since performance is measured based on response time, sending two closest is not necessarily optimal.

As discussed in Section 3.5.2, analytically characterizing the optimal policy for general networks seems untractable, due to the complex network dynamics that may propagate even small perturbations throughout the network. However, we are optimistic that for small network instances or specific network structures (such as linear networks), one may be able to obtain structural results on the optimal policy. Doing this for both the case with and without correlation may lead to interesting insights into where and how these two optimal policies differ.

The work in this chapter can be extended in several interesting ways. First, instead of only considering perfect or no correlation between the driving times, we could allow for intermediate levels of correlation by assuming that the driving time on a single edge is hyperexponential instead of exponential. By coupling only one of the branches of this distribution we can accommodate any correlation coefficient. Second, one could allow for driving time distributions beyond exponential. Changing the driving time distribution does not affect the MDP formulation, but rather the immediate costs. So one would have to generalize Proposition 3.3.1. Note that if we use a heavy-tailed distribution, the results can potentially show a more significant advantage of using the OPT policy instead of CF. We expect a larger optimality gap for CF in the case of heavy-tailed driving time distribution since the larger variance in response time necessitates more careful dispatching. Third, the MDP formulation itself can be enhanced by allowing more than two trucks to be dispatched to an incident, and we can generalize the definition of the response time accordingly. This would entail changing the action space from all actions that dispatch at most 2 trucks to those that dispatch at most k trucks. The main difficulty

in making this extension lies in computing the immediate cost $\mathbb{P}(R(\boldsymbol{a}(\boldsymbol{f}, j), j) > t^*)$ for those actions $\boldsymbol{a}$ that dispatch more than two trucks. If only the first truck to arrive is relevant, the costs can be computed along the lines of Proposition 3.3.1, by conditioning on the realizations of the driving times of all trucks. If the performance metric depends on more than just the first truck to arrive, generalizing the results obtained here may be more complex. Finally, when two trucks are dispatched from the same station, we may assume that each takes a different path in order to avoid driving-time correlation. Including this in the model may result in the optimal policy and heuristics to dispatch trucks from the same station more often.

3.7 Appendices

In this section we provide supplementary materials to the main body of the chapter.

3.7.1 List of Notations

$\mathcal{J} = \{1, ..., J\}$	Set of demand locations / nodes in a graph
E	Set of edges in a graph
$\mathcal{I} \subseteq \mathcal{J}$	Demand locations containing a fire station
$I = \lvert\mathcal{I}\rvert$	Number of fire stations
C_i	Number of fire trucks with the base station $i \in \mathcal{I}$
λ_j	Arrival rate of new fires at a location $j \in \mathcal{J}$
$1/\mu$	Expected time a truck remains busy after being dispatched
$\rho = \frac{\sum_{j \in \mathcal{J}} \lambda_j}{I\mu}$	Load of the system, that is, the amount of work per fire truck per time unit
f_i	Number of idle trucks at a station $i \in \mathcal{I}$
$\boldsymbol{e}_i$	Vector of length I with ith element equal to 1, and all other elements equal to zero
$\boldsymbol{f} = (f_1, ..., f_I)$	Vector representing the state of the system
$\boldsymbol{a}(\boldsymbol{f}, j) = (a_1(\boldsymbol{f}, j), ..., a_I(\boldsymbol{f}, j))$	The dispatch action taken if a new fire starts at a location j when in state $\boldsymbol{f}$
$0 \leq a_i(\boldsymbol{f}, j) \leq f_i$	Number of trucks dispatched from station $i \in \mathcal{I}$
$\mathcal{S} = \{(f_1, ..., f_I) \lvert 0 \leq f_i \leq C_i \ \forall i \in \mathcal{I}\}$	The system state space

$s(i,j)$	Shortest path between nodes i and j in a graph
$T_{i,j} = \sum_{e \in s(i,j)} X_e$	Traveling time between nodes i and j, where $X_e \sim \exp(1)$
T_0	Traveling time from a neighboring region to any demand location
$R(\boldsymbol{a}, j)$	Response time to a fire at a location j given a dispatch decision $\boldsymbol{a}$
$\tau = \sum_{j \in \mathcal{J}} \lambda_j + \mu \sum_{i \in \mathcal{I}} C_i$	Transition rate out of any state
$\mathcal{A}(\boldsymbol{f})$	Actions space in state $\boldsymbol{f} \in \mathcal{S}$
g^*	Average cost incurred per time unit
$h^*(\boldsymbol{f})$	Relative cost incurred over infinite time horizon when starting in state $\boldsymbol{f} \in \mathcal{S}$ compared to paying g^* every time unit
$\sigma_j(k) \in \mathcal{I}$	Fire station that is the base station for the kth closest truck to location j, assuming that truck is idle
$k_i, i = 1, 2$	Number of the closest and the second-closest idle truck in the list $\sigma_j(k)$, $j \in \mathcal{J}, k \in \{1, \ldots, \sum_i C_i\}$
$J(\boldsymbol{f}, t)$	Expected total cost under the CF policy during the time interval $[0, t]$ starting from state $\boldsymbol{f}$
T	Parameter of the OSIA heuristic indicating the time it takes for the system to get into the steady state by assumption

D_i	Arrival rate of requests for the truck at station i
$\rho_i = D_i/\mu$	Load of the $M/M/1/1$ queue representing fire station i
p_i	Busy probability of station i
t^*	Response time threshold
γ	Parameter that defines the response time threshold t^* for a given graph as a fraction of the maximum traveling time between two nodes

3.7.2 Proof of Proposition 3.3.1

Proof. The first statement can be readily proven by using the independence of Y_1 and Y_2:

$$P(\min\{Y_1, Y_2\} > t^*) = P(Y_1 \geq t^*)P(Y_2 \geq t^*).$$

Substituting in the distribution of Y_1 and Y_2 we obtain the desired result.

For the second statement we condition on the value of Y_0 to obtain the following expression:

$$\begin{aligned} P(Y_0 + \min\{Y_1, Y_2\} > t^*) &= \int_{y_0=0}^{\infty} f_{Y_0}(y_0) P(\min\{Y_1, Y_2\} > t^* - y_0) \mathrm{d}y_0 \\ &= \int_{y_0=0}^{t^*} f_{Y_0}(y_0) P(Y_1 > t^* - y_0) P(Y_2 > t^* - y_0) \mathrm{d}y_0 \\ &\quad + \int_{y_0=t^*}^{\infty} f_{Y_0}(y_0) \mathrm{d}y_0. \end{aligned}$$

By substituting the distribution function of Y_0, Y_1 and Y_2, and exchanging the order of integration and summation we obtain

$$\begin{aligned} P(R(\boldsymbol{a}, j) > t^*) &= \int_{y_0=0}^{t^*} f_{Y_0}(y_0) \sum_{n=0}^{w_1-1} \frac{(t^* - y_0)^n}{n!} e^{-t^*+y_0} \sum_{m=0}^{w_2-1} \frac{(t^* - y_0)^m}{m!} e^{-t^*+y_0} \mathrm{d}y_0 \\ &\quad + \sum_{n=0}^{w_0-1} \frac{t^{*n}}{n!} e^{-t^*} \\ &= \sum_{n=0}^{w_1-1} \sum_{m=0}^{w_2-1} \int_{y_0=0}^{t^*} f_{Y_0}(y_0) \frac{(t^* - y_0)^{n+m}}{n!m!} e^{-2t^*+2y_0} \mathrm{d}y_0 + \sum_{n=0}^{w_0-1} \frac{t^{*n}}{n!} e^{-t^*}. \end{aligned}$$

Expanding $(t^* - y_0)^{n+m}$ yields

$$
\begin{aligned}
&P(R(\boldsymbol{a}, j) > t^*) \\
&= \sum_{n=0}^{w_1-1} \sum_{m=0}^{w_2-1} \int_{y_0=0}^{t^*} \frac{y_0^{w_0-1}}{(w_0-1)!} e^{-y_0} \frac{1}{n!m!} \sum_{l=0}^{n+m} \binom{n+m}{l} t^{*l} (-y_0)^{n+m-l} e^{-2t^*+2y_0} \mathrm{d}y_0 \\
&\quad + \sum_{n=0}^{w_0-1} \frac{t^{*n}}{n!} e^{-t^*} \\
&= \sum_{n=0}^{w_1-1} \sum_{m=0}^{w_2-1} \sum_{l=0}^{n+m} \frac{e^{-2t^*} t^{*l} (-1)^{n+m-l}}{n!m!(w_0-1)!} \binom{n+m}{l} \int_{y_0=0}^{t^*} y_0^{n+m-l+w_0-1} e^{y_0} \mathrm{d}y_0 \\
&\quad + \sum_{n=0}^{w_0-1} \frac{t^{*n}}{n!} e^{-t^*},
\end{aligned}
$$

completing the proof. □

3.7.3 Additional Numerical Results

In this section we provide computational results for a wider range of parameters, supporting the findings in the main text. Tables 3.7, 3.6 and 3.8 show the relative improvement of OPT, OSI and OSIA over the CF policy in terms of fraction of late arrivals, depending on the number of fire trucks I and the size of the network d. The heuristics performance improves as both I and d increase, suggesting that their performance is better for larger networks and with more trucks. Table 3.9 shows the relative improvement of OSIA over CF for $\rho = 0.02$ and $I = 7$ for larger values of d. We see that as I and d grow larger, the gap with CF continues to increase as well.

model

Table 3.6: Aggregate performance evaluated over 150 random graphs ($\rho = 0.02$, $\gamma = 0.6$)

			uncorrelated			correlated		
I	d	Policy	min	mean	max	min	mean	max
3	4	OPT	0.0%	3.4%	27.8%	0.0%	4.8%	25.6%
		OSI	0.0%	3.4%	27.8%	0.0%	4.8%	25.6%
		OSIA	-2.5%	1.9%	20.5%	-2.9%	4.0%	20.3%
	5	OPT	0.0%	5.7%	36.5%	0.0%	6.8%	34.3%
		OSI	0.0%	5.7%	36.5%	0.0%	6.8%	34.3%
		OSIA	-0.2%	2.9%	26.3%	-0.8%	5.2%	31.7%
	6	OPT	0.0%	4.8%	32.8%	0.0%	5.5%	37.4%
		OSI	0.0%	4.8%	32.8%	0.0%	5.5%	37.4%
		OSIA	-4.0%	2.8%	20.6%	-5.4%	4.1%	31.6%
	7	OPT	0.0%	5.7%	49.2%	0.0%	6.0%	45.0%
		OSI	0.0%	5.7%	49.2%	0.0%	6.0%	45.0%
		OSIA	-1.4%	3.6%	36.4%	-2.1%	4.8%	35.4%
4	4	OPT	0.1%	8.4%	54.5%	0.0%	11.1%	62.5%
		OSI	0.1%	8.4%	54.5%	0.0%	11.1%	62.5%
		OSIA	-3.4%	4.6%	25.5%	-3.9%	8.7%	62.4%
	5	OPT	0.3%	9.0%	63.4%	0.4%	11.3%	67.9%
		OSI	0.3%	9.0%	63.3%	0.4%	11.3%	67.9%
		OSIA	-0.7%	4.5%	44.1%	-2.0%	8.7%	58.8%
	6	OPT	0.2%	9.5%	56.9%	0.3%	12.2%	60.7%
		OSI	0.2%	9.5%	56.9%	0.3%	12.2%	60.7%
		OSIA	0.1%	5.1%	41.6%	-2.6%	9.5%	59.1%
	7	OPT	0.4%	10.3%	74.7%	0.6%	12.0%	77.3%
		OSI	0.4%	10.3%	74.6%	0.6%	12.0%	77.3%
		OSIA	-0.5%	6.0%	55.3%	-0.6%	9.1%	71.4%
5	4	OPT	0.3%	13.6%	72.1%	1.1%	17.6%	78.2%
		OSI	0.3%	13.6%	72.0%	1.1%	17.4%	78.0%
		OSIA	-0.8%	8.1%	33.1%	-1.5%	14.7%	76.5%
	5	OPT	0.3%	14.4%	58.7%	1.3%	15.8%	63.7%
		OSI	0.3%	14.4%	58.6%	1.3%	15.7%	63.0%
		OSIA	0.1%	7.5%	37.3%	-3.3%	11.9%	59.9%
	6	OPT	0.5%	15.7%	69.2%	1.0%	17.4%	73.4%
		OSI	0.5%	15.6%	68.4%	1.0%	17.3%	73.1%
		OSIA	0.2%	8.6%	57.7%	-3.7%	13.5%	66.5%
	7	OPT	0.7%	17.7%	81.3%	0.7%	18.0%	81.3%
		OSI	0.7%	17.6%	80.5%	0.7%	17.9%	80.6%
		OSIA	0.3%	11.2%	59.3%	-9.1%	14.0%	79.1%
6	4	OPT	1.3%	18.0%	66.7%	1.4%	20.0%	76.7%
		OSI	1.3%	17.9%	65.2%	1.4%	19.8%	75.7%
		OSIA	0.5%	11.5%	42.9%	-1.2%	16.7%	69.8%
	5	OPT	1.3%	19.1%	72.9%	1.2%	20.9%	75.0%
		OSI	1.3%	19.0%	69.4%	1.2%	20.7%	74.8%
		OSIA	0.7%	11.2%	51.8%	0.8%	17.2%	67.3%
	6	OPT	0.6%	20.5%	74.4%	1.0%	22.0%	73.2%
		OSI	0.6%	20.2%	73.5%	1.0%	21.7%	72.6%
		OSIA	-0.3%	11.7%	61.6%	-14.3%	17.3%	64.5%
	7	OPT	1.2%	22.8%	81.0%	1.8%	24.1%	82.3%
		OSI	1.2%	22.5%	79.5%	1.8%	23.9%	81.4%
		OSIA	0.7%	14.2%	62.9%	1.5%	19.3%	73.6%

Table 3.7: Aggregate performance evaluated over 150 random graphs ($\rho = 0.04$, $\gamma = 0.6$)

			uncorrelated			correlated		
I	d	Policy	min	mean	max	min	mean	max
3	4	OPT	0.0%	4.9%	27.9%	0.0%	5.7%	25.9%
		OSI	0.0%	4.9%	27.9%	0.0%	5.7%	25.9%
		OSIA	-3.0%	3.4%	21.6%	-3.8%	4.8%	23.0%
	5	OPT	0.0%	7.3%	34.6%	0.0%	7.7%	33.7%
		OSI	0.0%	7.3%	34.6%	0.0%	7.7%	33.7%
		OSIA	-0.3%	5.4%	32.6%	-1.3%	6.5%	33.6%
	6	OPT	0.0%	6.4%	32.0%	0.0%	6.5%	35.5%
		OSI	0.0%	6.4%	32.0%	0.0%	6.5%	35.5%
		OSIA	-3.9%	4.9%	24.3%	-5.4%	5.4%	32.8%
	7	OPT	0.0%	7.2%	45.7%	0.0%	7.3%	43.7%
		OSI	0.0%	7.2%	45.7%	0.0%	7.3%	43.7%
		OSIA	-1.7%	5.6%	38.6%	-0.7%	6.2%	38.6%
4	4	OPT	0.1%	10.2%	44.2%	0.0%	12.5%	57.4%
		OSI	0.1%	10.2%	44.2%	0.0%	12.5%	57.4%
		OSIA	-3.3%	6.8%	36.9%	-7.6%	10.1%	57.0%
	5	OPT	0.6%	10.8%	52.7%	0.7%	12.4%	59.0%
		OSI	0.6%	10.8%	52.5%	0.7%	12.4%	58.9%
		OSIA	-0.9%	7.8%	45.4%	-0.3%	10.4%	58.5%
	6	OPT	0.4%	11.5%	51.1%	0.6%	13.5%	57.4%
		OSI	0.4%	11.5%	51.1%	0.6%	13.5%	57.4%
		OSIA	0.3%	8.5%	41.8%	-0.9%	11.4%	52.9%
	7	OPT	0.8%	12.2%	63.3%	1.0%	13.4%	66.7%
		OSI	0.8%	12.2%	63.2%	1.0%	13.4%	66.7%
		OSIA	0.5%	9.3%	56.9%	-1.2%	11.1%	65.4%
5	4	OPT	0.8%	16.8%	65.9%	1.7%	19.9%	70.6%
		OSI	0.8%	16.7%	65.5%	1.7%	19.7%	69.6%
		OSIA	-0.7%	11.8%	57.6%	-1.0%	17.2%	68.0%
	5	OPT	0.9%	17.8%	56.5%	1.2%	18.8%	61.9%
		OSI	0.9%	17.6%	56.3%	1.2%	18.6%	61.4%
		OSIA	0.4%	11.9%	39.8%	-2.0%	15.1%	56.0%
	6	OPT	1.1%	18.6%	57.7%	1.9%	19.6%	59.5%
		OSI	1.1%	18.4%	57.6%	1.9%	19.3%	59.4%
		OSIA	0.9%	13.6%	52.6%	-0.9%	16.2%	58.2%
	7	OPT	1.4%	21.2%	75.1%	0.9%	21.0%	76.5%
		OSI	1.4%	20.9%	74.2%	0.9%	20.8%	75.5%
		OSIA	1.1%	16.5%	68.1%	0.4%	18.1%	74.4%
6	4	OPT	2.5%	22.4%	62.8%	2.7%	24.1%	69.3%
		OSI	2.5%	22.0%	62.0%	2.7%	23.6%	69.1%
		OSIA	1.3%	15.2%	42.7%	-1.8%	20.1%	67.3%
	5	OPT	3.1%	23.7%	65.1%	2.8%	24.9%	69.4%
		OSI	3.0%	23.3%	63.0%	2.8%	24.5%	67.3%
		OSIA	2.4%	16.5%	50.3%	1.9%	20.8%	62.5%
	6	OPT	1.5%	24.9%	68.4%	2.1%	25.9%	70.3%
		OSI	1.5%	24.4%	65.0%	2.1%	25.3%	68.9%
		OSIA	0.8%	17.9%	56.0%	-15.0%	21.6%	63.8%
	7	OPT	2.9%	27.3%	73.1%	3.7%	28.1%	73.6%
		OSI	2.9%	26.9%	72.2%	3.7%	27.5%	72.0%
		OSIA	1.4%	20.5%	66.3%	3.4%	23.7%	67.7%

Table 3.8: Aggregate performance evaluated over multiple random graphs ($\rho = 0.1$, $\gamma = 0.6$)

			uncorrelated			correlated		
I	d	Policy	min	mean	max	min	mean	max
3	4	OPT	0.0%	5.5%	20.5%	0.0%	5.8%	19.7%
		OSI	0.0%	5.5%	20.5%	0.0%	5.8%	19.7%
		OSIA	-2.6%	5.2%	20.5%	-3.5%	5.5%	19.7%
	5	OPT	0.0%	7.4%	25.1%	0.0%	7.5%	25.2%
		OSI	0.0%	7.4%	25.1%	0.0%	7.5%	25.2%
		OSIA	-0.6%	7.1%	24.8%	-1.4%	7.2%	25.0%
	6	OPT	0.0%	6.7%	24.8%	0.0%	6.6%	26.6%
		OSI	0.0%	6.7%	24.8%	0.0%	6.6%	26.6%
		OSIA	-2.6%	6.4%	24.0%	-3.7%	6.3%	26.6%
	7	OPT	0.0%	7.1%	32.4%	0.0%	7.1%	34.5%
		OSI	0.0%	7.1%	32.4%	0.0%	7.1%	34.5%
		OSIA	-1.9%	6.8%	31.9%	-2.3%	6.9%	34.4%
4	4	OPT	0.1%	9.6%	25.7%	0.0%	11.4%	37.3%
		OSI	0.1%	9.6%	25.7%	0.0%	11.3%	37.3%
		OSIA	-0.1%	8.9%	24.8%	-0.2%	10.8%	37.2%
	5	OPT	1.0%	10.3%	34.6%	0.9%	11.4%	42.5%
		OSI	1.0%	10.2%	34.3%	0.9%	11.3%	42.4%
		OSIA	-0.5%	9.7%	34.4%	-0.7%	10.9%	42.4%
	6	OPT	1.1%	11.0%	31.2%	1.2%	12.3%	37.1%
		OSI	1.1%	11.0%	31.2%	1.2%	12.3%	37.1%
		OSIA	1.1%	10.4%	31.0%	1.1%	11.9%	37.0%
	7	OPT	1.4%	11.3%	39.2%	1.2%	12.2%	42.5%
		OSI	1.4%	11.3%	39.0%	1.2%	12.1%	42.4%
		OSIA	1.1%	10.8%	38.9%	0.3%	11.7%	42.4%
5	4	OPT	2.2%	16.0%	46.6%	2.5%	18.6%	50.6%
		OSI	2.2%	15.8%	46.1%	2.5%	18.2%	49.8%
		OSIA	-3.0%	13.2%	45.6%	-2.5%	16.6%	49.7%
	5	OPT	2.3%	16.6%	41.0%	2.3%	17.9%	43.2%
		OSI	2.3%	16.3%	40.7%	2.3%	17.6%	42.8%
		OSIA	-0.2%	13.9%	38.8%	-0.3%	15.9%	40.9%
	6	OPT	2.4%	16.8%	37.4%	3.5%	17.9%	39.8%
		OSI	2.3%	16.5%	37.1%	3.5%	17.6%	39.5%
		OSIA	0.8%	14.4%	35.6%	-1.0%	16.0%	37.5%
	7	OPT	2.7%	19.7%	52.4%	1.6%	20.2%	54.9%
		OSI	2.7%	19.3%	51.6%	1.6%	19.8%	54.4%
		OSIA	0.9%	17.4%	49.5%	-0.4%	18.5%	54.2%
6	4	OPT	4.9%	20.7%	46.2%	5.1%	23.4%	51.0%
		OSI	4.9%	20.2%	45.7%	5.1%	22.6%	50.3%
		OSIA	1.0%	15.8%	43.0%	-0.5%	20.3%	49.2%
	5	OPT	7.1%	21.8%	43.2%	6.1%	24.1%	48.8%
		OSI	7.0%	21.3%	41.9%	6.0%	23.4%	47.1%
		OSIA	2.2%	17.3%	40.2%	2.8%	20.9%	44.7%
	6	OPT	4.0%	22.7%	49.2%	4.7%	24.6%	52.3%
		OSI	4.0%	22.0%	47.5%	4.7%	23.7%	50.9%
		OSIA	1.4%	18.4%	45.2%	2.3%	21.6%	50.8%
	7	OPT	6.0%	24.8%	53.6%	5.4%	26.5%	54.6%
		OSI	6.0%	24.1%	52.4%	5.3%	25.7%	53.2%
		OSIA	2.8%	20.7%	48.8%	2.3%	23.4%	49.8%

Table 3.9: Aggregate performance of OSIA over 50 random graphs ($\rho = 0.02$, $\gamma = 0.6$, $I = 7$)

	uncorrelated			**correlated**		
d	**min**	**mean**	**max**	**min**	**mean**	**max**
7	0.5%	18.8%	81.4%	1.9%	24.1%	76.1%
8	1.2%	16.0%	59.0%	0.9%	19.9%	57.6%
9	0.5%	20.3%	88.4%	2.0%	28.5%	81.4%
10	1.0%	22.3%	89.0%	2.0%	26.2%	83.7%

4

Real-time Dispatching and Relocation of Emergency Service Engineers

In this chapter we consider a network of geographically distributed capital goods, maintained by a set of service engineers who are expected to respond quickly to machine breakdowns. We are interested in the question which service engineers to dispatch to what breakdowns, and how to relocate these engineers to maintain good coverage. We propose and evaluate a range of scalable dispatching and relocation heuristics inspired by an extensive research literature in the domain of emergency medical services. We compare the proposed heuristics against each other using comprehensive simulation experiments, and benchmark the best combination of dispatching and relocation heuristics against the optimal policy. We find that this combined heuristic performs close to optimal, while easily scaling to realistic-sized networks, making it suitable for practical applications.

The work in this chapter is based on [85]: A. Pechina, D. Usanov, P.M. van de Ven, and R.D. van der Mei. Real-time Dispatching and Relocation of Emergency Service Engineers. *European Journal of Operational Research*, 2019. In revision.

4.1 Introduction

Capital goods are both expensive and essential to the business of their users, and frequent unplanned downtime may have significant repercussions. Consider for instance the reputation damage suffered by train companies unable to maintain the train schedule, or the potentially life-threatening consequences of a broken MRI scanner. In order to ensure continuous operation of these capital goods, manufacturers of such products typically provide post-sale support, which may include installation, warranties, spare parts supply and maintenance services. Providing good post-sale support is an important revenue source and competitive advantage for manufacturers [77]. In 2006, after-sales services accounted for 40% of the profit of a sample of 120 large US manufacturing companies [61].

An essential component of post-sale support is *corrective* maintenance, i.e., repairing machines that have suffered breakdowns. In practice, a certain service level (e.g., the percentage of failures that should be fixed within a certain time threshold) is determined in the service contract between a manufacturer and the user. Failure to meet these service levels may result in penalties for the manufacturer.

To meet these ever-tightening service level agreements for corrective maintenance, a manufacturer must be able to quickly dispatch the necessary resources to the site of a breakdown. Since the capital goods are geographically dispersed (e.g., located in many different hospitals), this requires a carefully planned network of spare part warehouses and service engineers. Establishing such networks requires striking a delicate balance between maintaining customer satisfaction and achieving low operational costs [76]: building a large number of warehouses would for instance guarantee fast response times, but this is very costly. Similarly, maintaining many service facilities and employing a large number of service engineers is costly for a manufacturer. Thus, creating a cost-effective maintenance network requires both careful planning of the service facilities and warehouses, and managing the scarce resources in an efficient manner.

While the problem of managing a spare parts network has been extensively studied in research literature (see, e.g., [117] for an overview), managing service engineers has received little attention so far. Moreover, certain service networks (e.g., software systems support) do not require spare parts at all. Because of this, we focus here on the problem of how to best manage service engineers.

Service engineers are typically located at geographically dispersed base stations, from which they can be quickly dispatched to the site of a breakdown. Here we assume that the location of these base stations and the number of service engineers are given, and we are interested in the question of how to manage the service engineers. This includes the

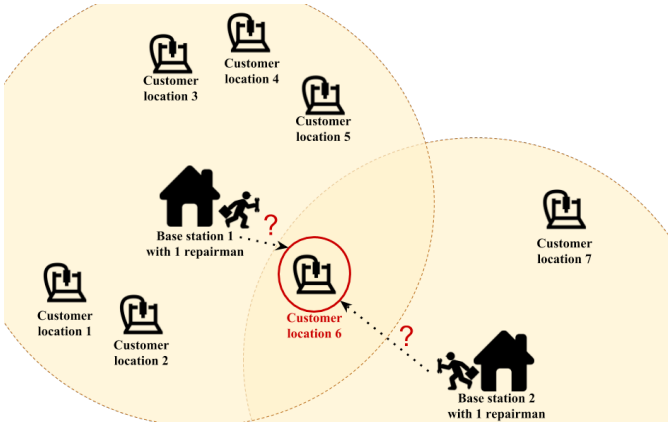


Figure 4.1: Example of a dispatching decision

following decisions: (i) which service engineer to dispatch to a breakdown; (ii) should we dispatch one right away or wait for a nearby engineer to become available; (iii) to which base station should we send an engineer who just finished a repair; and (iv) once a service engineer is dispatched to repair a machine, should we relocate idle service engineers to improve the coverage of the region.

Together, these decisions form a trade-off between responding quickly to a current breakdown on the one hand, and maintaining good coverage for future breakdowns on the other hand. The goal is to find a dispatching-and-relocation policy that minimizes the long-term costs caused by violations of the service level agreements with customers. This is a complicated problem due to the fact that dispatching and relocation decisions always increase coverage of one part of the region, but at the same time decrease coverage of another part. Making a decision requires finding a good balance between the possible costs from failures that are already reported and potential costs from future failures.

Consider, for example, a situation depicted in Figure 4.1. The yellow circles centered around the base stations represent the areas reachable within a given time limit from those base stations. If a call arrives from a customer location 6, the decision should be made which service engineer (from base station 1 or from base station 2) to dispatch. The first service engineer is closer to the customer and can arrive earlier. However, if the closest service engineer is dispatched, customer locations 1-5 are too far from the remaining idle engineer. If a failure occurs in one of these locations, it cannot be fixed in time.

We divide the policy for managing service engineers into two parts: *dispatching* and *relocation*. The dispatching policy is responsible for dispatching service engineers to emergency calls, so it answers the following two questions:

(i) Should a newly reported failure be assigned to one of the idle service engineers or should it be served by one of the busy engineers after he finishes his current job?

(ii) Which service engineer should be dispatched to the customer?

The relocation policy prescribes the base location of the idle service engineers according to the system state. It answers two questions:

(i) Should idle service engineers be relocated from their current base stations to different ones to improve coverage?

(ii) To which base station should we send a service engineer that just finished a job?

Traditionally, Markov decision theory can be used to find optimal policies. However, for realistic-sized problem instances this approach is often infeasible due to computational complexity and high memory usage. Instead, we look for scalable dispatching and relocation heuristics that perform close to optimal. Although management of service engineers is not well-studied in the research literature, we observe that the problem is close to that of Emergency Medical Services (EMS), which also deals with dispatching and relocation of resources [108]. Recently, much progress has been made in dispatching and relocation for EMS. In this chapter we adapt both dispatching and relocation heuristics from that domain to our setting, and show that these perform surprisingly well in this setting as well.

We compare the performance of these heuristics by means of simulation over a wide range of parameters, to assess whether there is an approach that performs best for any type of system. In addition, we formulate the model as a Markov Decision Process (MDP), and benchmark the best performing heuristic against the optimal policy for a small instance.

To summarize, in this chapter we make the following contributions:

1. We introduce a new model for dynamic dispatching and relocation of service engineers, and formulate it as an MDP;

2. We adapt a number of dispatching and relocation heuristic from the EMS literature, and conduct an extensive computational studies, where we compare the performance of the proposed algorithms.

The remainder of the chapter is organized as follows. Section 4.2 gives an overview of related literature. In Section 4.3, we present the model and formulate it as a Markov decision process. Different dispatching and relocation heuristics are discussed in Sections 4.4 and 4.5, respectively. Numerical experiments are presented in Section 4.6. Finally, Section 4.7 contains conclusions and suggestions for further research.

4.2 Literature Review

The problem of real-time management of service engineers arises in the context of spare parts management in a service logistics network. There is extensive research in the area of spare parts management (see [117] for an overview). Most of it focuses on operations for spare parts. One such recent work is by Tiemessen et al. [101], where the authors study the problem of dynamic dispatching of spare parts on a network with multiple customer classes. To our knowledge, there is very limited research on real-time service engineer management. In [8] and [55] the authors study optimal repairmen allocation policies in a simplified setting, for instance ignoring the geographical locations of the engineers. The work closest to ours is by Drent et al. [23], where the authors showed the benefit of deviating from the closest-first dispatching policy, as well as proactively relocating service engineers. However, the authors used a stylized grid-like type of network with deterministic repair and traveling times, where service engineers could reside anywhere on the grid.

The field of EMS is well-studied and is closely related to our setting. There are however a number of crucial differences between these two application areas. For instance, in our setting there is only a finite number of machines that can break down, and each breakdown affects the rate at which new ones occur. This is in contrast to EMS, where the ongoing incidents do not affect the arrival rate of new incidents. Moreover, many crucial parameters such as the load, coverage area, target response times and service times differ between these two settings, necessitating the present study. It is also worth noting that to our knowledge there is no comprehensive comparison between heuristics in the EMS area, and this study is a first step towards that as well. In this chapter, we introduce a new model for dynamic management of service engineers, and develop several heuristics inspired by the research in the field of EMS.

In the remainder of this section we outline the most relevant results from the area of EMS operations. For an extensive overview of recent work on location, relocation and dispatching of ambulances, we refer to [7]. We organize our literature review according to the type of methods used. First, we discuss Integer Linear Programming (ILP) based approaches. Then we discuss results obtained using MDP theory, followed by several

heuristics that were successfully used for dispatching and relocation. Finally, we cover those references that use Approximate Dynamic Programming (ADP) for dynamic EMS management.

ILP-based approaches. A compliance table is a policy that precomputes the optimal locations depending on the number of available service units. Every time this number changes (e.g., when a call arrives, or when a service is finished), idle service units are repositioned according to the compliance table. A method called Maximum Coverage Relocation Problem (MCRP) was introduced in [31] to compute compliance tables, where for each number of available servers the coverage was maximized. The algorithm was later extended in [108] to the Maximal Expected Coverage Relocation Problem (MEXCRP). MEXCRP compliance tables incorporate the busy fraction of the ambulances.

The problem of choosing a service unit that can be best dispatched to a new emergency call can also be formulated as an ILP. According to the computational study of Jagtenberg *et al.* [49], it can even outperform other dispatching policies that use more information about the state of the system.

MDP-based approaches. A common way to find the optimal policy is to model the system as an MDP with either continuous or discrete time. For small systems the optimal policy can be found, for instance, using policy or value iteration [48, 124]. For real-life systems, however, the state space is often too large and the problem is computationally intractable. One way to address this problem was considered in [48]: instead of finding the exact optimal policy, the authors perform a limited number of value iteration steps, and compare the results for different numbers of steps with other policies.

Heuristics. Another approach to tackle the problem of large state space is to make decisions in real time rather then precomputing the best desicion for each possible state. This applies to both relocation and dispatching problems. The first real-time relocation model was proposed in [30]. It is based on the Double Standard Model [29], and maximizes the demand covered by at least two vehicles. It also minimizes the relocation costs, so the relocation history is taken into account. Another relocation model, maximizing the preparedness of the system (i.e., the capacity of the system to answer future demands), was introduced in [2]. The authors proposed a method to find a relocation policy that minimizes travel times.

In [47] the Dynamic Maximal Expected Coverage Location Problem (DMEXCLP) heuristic was proposed for redeployment of service units that just finished their service. The heuristic is based on calculating the expected covered demand and choosing the new location of the service units accordingly. This research was later extended in [109],

where relocation was allowed not only after the service completion but also right after dispatching a service unit. The authors also studied how different restrictions, such as a restriction on the maximum distance of a relocation, influence the performance of the system. Two types of regions (rural and urban) were considered and it was shown that the optimal strategy depends on the type of the region.

The same ideas can be used for making dispatching decisions. In [30] the authors proposed choosing, among all service unites that can reach the incident location in time, the one that leads to the minimal relocation time. Dispatching a service unit that causes the smallest decrease in preparedness was proposed in [2]. In [49] the expected covered demand was used instead, and the obtained dispatching policy outperformed the commonly used closest-first dispatching policy. Further research also incorporated the possibility of waiting for a busy service unit to finish its service, instead of dispatching an idle one [110].

ADP-based approaches. In [71] an ADP approach using so-called approximate policy iteration was proposed for dynamic ambulance management. The authors considered the problem of ambulance redeployment upon completion of their job, for a system with no other types of relocation and a fixed closest-first dispatching policy. More recently, in [93] the same framework was used to optimize both the dispatching policy and the redeployment of ambulances upon service completion. Finally, in [79] the authors considered a general problem of ambulances dispatching and relocation, with the possibility to reposition idle ambulances and to put incoming calls into a queue.

4.3 Model

We consider a service region consisting of a set of identical machines $\mathcal{K} = \{1, ..., K\}$ and a set of base stations $\mathcal{R} = \{1, ..., R\}$. Locations of the machines and the base stations are fixed, and the traveling times between each pair of locations are deterministic and known. Let $\mathcal{M} = \{1, ..., M\}$ denote the set of service engineers. Each service engineer is rested at one of the base station when idle.

The time until the next breakdown of a working machine is exponentially distributed with rate λ. Upon the breakdown of a machine, exactly one service engineer is needed to repair it, and the repair time of each machine is exponentially distributed with rate μ. This repair time does not include the traveling time required for a service engineer to reach the location of a machine. However, the repair starts immediately after a service engineer arrives to the location of a broken machine, assuming all the necessary tools and spare parts needed for repair are available upon arrival. A service engineer can be

dispatched immediately after a failure occurred, or the failed machine can be put in a queue waiting for a service engineer to be assigned. The time between the failure and the arrival of a service engineer to the failure location is called *response time*. If the response time exceeds the time limit t^*, costs are incurred. We discuss the costs structure later in this section.

We consider the state of the system immediately after one of the following events happen:

1. a failure of a machine;
2. end of repair;
3. arrival of a service engineer at a base station;
4. arrival of a service engineer at a machine location.

The event e is described by the tuple (e_t, e_l), where $e_t \in \{1, 2, 3, 4\}$ indicates the type of the event, and $e_l \in \{1, ..., L\}$ the event location. The location can be either a broken machine or a base station, with the total number of locations $L = K + R$. Let l_k indicate the location of machine $k \in \mathcal{K}$, and l_r the location of base station $r \in \mathcal{R}$.

Let κ_k denote the state of machine $k \in \mathcal{K}$. Here $\kappa_k = 0$ if machine k is working, $\kappa_k = -1$ if machine k is in repair, and $\kappa_k = t$ if machine k has been waiting for repair for t time units. Note that immediately after failure of machine k, its state is $\kappa_k = 0$ despite the machine is broken, as its elapsed waiting time is 0 time units. This plays a role when defining the number of broken/working machines in a given state. That is, if the event type is $e_t = 1$, the number of broken machines is $\left|\left\{k \in \mathcal{K} : (\kappa_k \neq 0) \vee (e_l = l_k)\right\}\right|$. The state of all machines is represented by the vector $\boldsymbol{\kappa} = (\kappa_1, ..., \kappa_K)$.

We describe the state of all service engineers by the vector $\boldsymbol{\eta} = (\eta_1, .., \eta_M)$. Here $\eta_m = (l_m, d_m)$ describes the state of service engineer m, where l_m indicates the destination of that service engineer, and d_m indicates the remaining time left to reach the destination. If the service engineer m is residing at a location l (either doing a repair or waiting at a base station), then $l_m = l$ and $d_m = 0$.

The state of the system immediately after an event e is described by the tuple $(t, e, \boldsymbol{\kappa}, \boldsymbol{\eta})$, where t is the time of the event. We use $t(s)$, $e(s)$, $\boldsymbol{\kappa}(s)$, $\boldsymbol{\eta}(s)$ to represent each component of a given state s. The state space is infinitely large, with non-recurrent states, as the time is included in the state description.

4.3.1 Actions

Whenever an event occurs, an action is taken. The set of possible actions is defined by the type of event e_t. Below we formulate the action space per event type, but first we introduce additional notation and general assumptions regarding feasible actions. We assume that only idle service engineers are allowed to be dispatched to a broken machine. We denote the set of all idle service engineers in state s by $\mathcal{F}(s) = \big\{m \in \mathcal{M} \mid l_m(s) \in \mathcal{R}\big\}$. Note that the service engineer who is still on the way to the base station is considered idle, and can be dispatched for repair. In that case, we assume that the service engineer first reaches the base station and then immediately departs to the machine location. We assume, however, that relocation of traveling idle service engineers is not allowed. We do this to avoid situations when an idle service engineers is continuously relocated from one station to another, never reaching his/her destination. Let $\mathcal{F}_0(s) = \big\{m \in \mathcal{M} \mid l_m(s) \in \mathcal{R}, d_m = 0\big\}$ denote the subset of idle service engineers that are not traveling. Let also denote the set of broken machines that are waiting in the queue in state s by $\mathcal{Q}(s) = \big\{k \in \mathcal{K} \mid \{m \in \mathcal{M} \mid l_m(s) = k\} = \emptyset\big\}$.

Event type $e_t = 1$. In case of a new failure, the action consists of a dispatching decision and a relocation decision. At this point, dispatching is allowed only to the newly broken machine, but not to the machines in the queue. One of the idle service engineers $\mathcal{F}(s)$ can be dispatched to the new breakdown, or a machine can be put in the queue. If dispatching is done, a relocation is allowed of one of the remaining stationary idle service engineers from his/her current location to another base station. However, if the repair request is put in the queue, relocation is not allowed. Let the binary vector $\boldsymbol{X}$ of length M represent the dispatching decision, and the binary $M \times R$ matrix $\boldsymbol{Y}$ the relocation decision. Here $X_m = 1$ indicates that service engineer m is dispatched for repair, and $Y_{mr} = 1$ indicates that service engineer m is relocated to base station r. The action space can be formally described as

$$\begin{aligned}\mathcal{A}_1(s) = \Bigg\{(\boldsymbol{X},\boldsymbol{Y}) \,|&X_m = 0, \quad m \notin \mathcal{F}(s);\; Y_{mr} = 0, \quad m \notin \mathcal{F}_0(s),\; r \in \mathcal{R};\\ &\sum_{m\in\mathcal{M}} X_m \le 1;\; \sum_{\substack{m\in\mathcal{M},\\ r\in\mathcal{R}}} Y_{mr} \le 1;\\ &\sum_{\substack{m\in\mathcal{M},\\ r\in\mathcal{R}}} X_m Y_{mr} = 0;\; \Big(1 - \sum_{m\in\mathcal{M}} X_m\Big) \sum_{\substack{m\in\mathcal{M},\\ r\in\mathcal{R}}} Y_{mr} = 0\Bigg\},\end{aligned} \tag{4.1}$$

where the constraints ensure that at most one idle service engineer is dispatched, at most one other idle stationary service engineer is relocated, and relocation is done only upon dispatching.

Event type $e_t = 2$. When a service engineer finishes repairing a machine, we can either allocate that service engineer to one of the base stations or dispatch to one of the broken machines in the queue. No relocation of other service engineers is allowed in this case. In [85] the authors included the possibility of relocating one other idle service engineer for this type of event. However, the best performing heuristic in that study did not make use of such extra relocations. Assuming that the potential gain from one extra relocation is marginal, we exclude the possibility of extra relocation to reduce computational complexity of the ADP approach. Moreover, using the same action space for both approaches makes the comparison cleaner.

Let the binary vector $\boldsymbol{Z} = (Z_1, ..., Z_L)$ represent the redeployment decision. Here $Z_l = 1$ indicates that the service engineer that just finished a repair is redeployed to location l, where l is the location of either one of the base stations or one of the machines in the queue. The corresponding action space is given by

$$\mathcal{A}_2(s) = \left\{ \boldsymbol{Z} \mid Z_l = 0, \quad l \notin \{l_k : k \in \mathcal{Q}(s)\} \cup \{l_r : r \in \mathcal{R}\}, \ \sum_{l \in \mathcal{L}} Z_l = 1 \right\}, \tag{4.2}$$

where the constraints ensure that the service engineer is redeployed to either a base station or a machine from the queue.

Event type $e_t = 3$. When a service engineer arrives at a base station, he can be immediately dispatched to one of the machines in the queue or left idle at that base station. Relocation of any sort is not allowed in that case. Let the binary vector $\boldsymbol{U} = (U_1, ..., U_K)$ represent the dispatching decision. Here, $U_k = 1$ indicates that the service engineer that just arrived at a base station is dispatched to machine $k \in \mathcal{Q}(s)$. The action space can be written as

$$\mathcal{A}_3(s) = \left\{ \boldsymbol{U} \mid U_k = 0, \quad k \notin \mathcal{Q}(s), \ \sum_{k \in \mathcal{K}} U_k \leq 1 \right\}, \tag{4.3}$$

where the constraints ensure that the service engineer is dispatched to at most one machine from the queue. Note that $\mathcal{A}_3(s)$ is empty if and only if the queue is empty in state s.

Event type $e_t = 4$. Once a service engineer arrives at a broken machine, he immediately starts repairing the machine. In that case no action is taken, hence

$$\mathcal{A}_4(s) = \emptyset. \tag{4.4}$$

4.3.2 Transitions

We describe evolution of the system via the stochastic process $\{s_n\}_{n\in\mathbb{N}}$ embedded on decision epochs. To that end, we need to derive the distribution of time until the next transition for a given state s_n and action a_n, as well as the transition probabilities.

Given state s_n and action a_n, the next state s_{n+1} is defined by the function $s_{n+1} = \Phi(s_n, a_n, \omega(s_n, a_n))$, where the random element $\omega(s_n, a_n)$ determines the next event. The next event can be either a failure of one of the working machines, the end of an ongoing repair, or an arrival of one of the traveling service engineers to his/her destination. Let $d(s_n, a_n)$ denote the minimum remaining distance in time units over all traveling service engineers after action a_n is taken in state s_n. Let $d(s_n, a_n) = \infty$ if there are no traveling service engineers. Let also $\mathcal{W}(s_n)$ denote the set of all working machines and $\mathcal{H}(s_n)$ the set of machines in repair in state s_n. Note that those sets are not affected by the action a_n. Recall that the time until breakdown of each machine in $\mathcal{W}(s_n)$ is exponentially distributed with rate λ, and the time until the end of repair of each machine in $\mathcal{H}(s_n)$ is exponentially distributed with rate μ. Hence, the time until the next event is distributed as the minimum of $d(s_n, a_n)$ and an exponentially distributed random variable $\Gamma(s_n)$ with rate $\eta(s_n) = \lambda|\mathcal{W}(s_n)| + \mu|\mathcal{H}(s_n)|$.

The probability that the next event is of type $e_t(s_{n+1}) = 1$ or $e_t(s_{n+1}) = 2$ is equal to

$$P(\Gamma(s_n) < d(s_n, a_n)) = 1 - e^{-\eta(s_n)d(s_n,a_n)}.$$

Then the probability that the next event is

- the failure of machine $k \in \mathcal{W}(s_n)$ equals

  $$\frac{\lambda}{\eta(s_n)}(1 - e^{-\eta(s_n)d(s_n,a_n)}),$$

  if $\mathcal{W}(s_n) \neq \emptyset$, and 0 otherwise;
- the end of repair of machine $k \in \mathcal{H}(s_n)$ equals

  $$\frac{\mu}{\eta(s_n)}(1 - e^{-\eta(s_n)d(s_n,a_n)}),$$

  if $\mathcal{H}(s_n) \neq \emptyset$, and 0 otherwise;
- the arrival of a service engineer to his/her destination equals

  $$e^{-\eta(s_n)d(s_n,a_n)},$$

  if there are traveling service engineers after action a_n is taken in state s_n (i.e., $s(s_n, a_n) < \infty$), and 0 otherwise.

The state of all service engineers immediately after action a_n is taken in state s_n is updated in a straightforward manner by changing destinations and remaining distances accordingly. Upon realization of the next event $e(s_{n+1})$ and transition time $\min\{\Gamma(s_n), d(s_n, a_n)\}$, the state s_{n+1} is obtained as follows. The time is equal to $t(s_{n+1}) = t(s_n) + \min\{\gamma(s_n), d(s_n, a_n)\}$, where $\gamma(s_n)$ is the realization of $\Gamma(s_n)$. If the next event is the repair of machine k, its state is set to $\kappa_k = 0$. Waiting times of all machines in the queue are increased by $\min\{\gamma(s_n), d(s_n, a_n)\}$, and traveling times of all traveling service engineers are decreased by $\min\{\gamma(s_n), d(s_n, a_n)\}$. If the next even is an arrival of a service engineer at a machine k, the state of that machine is set to $\kappa_k = -1$.

4.4 Dispatching Heuristics

The dispatching policy determines *when* and *which* service engineer to assign to a call, based on the current state of the system. When a breakdown occurs, a customer wants a service engineer to arrive on scene as quickly as possible. However, when a service engineer is dispatched from one of the base stations, the coverage of the customers around this base station decreases, which may lead to high future costs. Thus, a good dispatching policy finds a balance between minimizing immediate costs and future costs.

In order to focus fully on the problem of dispatching, in this section relocation is not allowed. In Section 4.5 we consider the complementary problem of varying the relocation policy while keeping of fixed dispatching policy. So after service completion, the engineer returns to the his preallocated base station. The initial allocation of service engineers to base stations is made to maximize the expected covered demand and is a solution to the ILP problem (4.29) described in Appendix 4.8 below.

In our comparative study we consider the following five dispatching policies:

DP1: Closest-first dispatching (without waiting);

DP2: Maximal coverage dispatching (without waiting);

DP3: Maximal expected coverage dispatching (without waiting);

DP4: Minimal response time dispatching with unknown remaining repair time;

DP5: Minimal response time dispatching with known remaining repair time.

In Section 4.4.1, we consider dispatching heuristics DP1 - DP3 that put a call into the queue only if there are no idle service engineers (no waiting). Otherwise, a service engineer has to be dispatched immediately. In Section 4.4.2, we discuss DP4 and DP5 that allow calls to be put into the queue even if idle engineers are available. A comparison of all dispatching policies based on simulation can be found in Section 4.6.2.

4.4.1 Dispatching Policies without Waiting

Recall from (4.1) that in the event of type $e_t(s) = 1$, the set of all possible actions is described by a vector X that represents the dispatching decision, and a matrix Y that represents the relocation decision. As in this section relocation is not allowed, all elements of the matrix Y are always set to 0. We do the same for the relocation matrix Y for $e_t(s) = 2$, see (4.2).

In this section, when a call arrives in the system with at least one idle service engineer, an engineer is dispatched immediately. Recall that $\mathcal{F}(s)$ denotes the set of all idle service engineers in state s, so for any state s with $\mathcal{F}(s) \neq \emptyset$ and event $e_t(s) = 1$, the set of possible actions from (4.1) reduces to

$$\mathcal{A}_1(s) = \left\{ (X, Y) \mid \sum_{m \in \mathcal{F}(s)} X_m = 1, Y_{mr} = 0 \ \ \forall m \in \mathcal{F}(s), r \in \mathcal{R} \right\}.$$

If $\mathcal{F}(s) = \emptyset$, then the call is put in the queue and no decision should be made. Note also that under these restrictions the set of possible actions for $e(s) = 3$ is empty, as a call cannot be put in the queue if there is an idle service engineer in the system.

DP1. Consider state s with the event $e_t(s) = 1$ (i.e., a call arrives from machine k) and $\mathcal{F}(s) \neq \emptyset$. To describe the dispatching policy, we need to calculate the vector X depending on state s. The simplest and most widely used dispatching policy in practice is the so-called *closest-first* policy. Under this policy, the closest idle service engineer is always dispatched to a call. So

$$X_m = 1 \iff m = \underset{n \in \mathcal{F}(s)}{\arg\min} \left(d_{l_n k} + d_n \right),$$

where $d_{l_n k}$ is the distance in time between the destination of the service engineer n and the source of the call, the machine k.

DP2. One of the possible metrics of the expected system performance is *coverage*, i.e., the number of machines covered by at least one service engineer. When a call arrives, for each idle service engineer m that can reach machine k in time, the remaining coverage of the system without him/her is defined as:

$$coverage(m) = \sum_{k' : \kappa_{k'} = 0} \mathbb{I}\left\{ \exists n \in \mathcal{F}(s) : n \neq m, \ d_{l_n k'} \leq t^* \right\}.$$

Then in *coverage-based dispatching*, the service engineer that leaves most machines covered by others is dispatched. If there are no service engineers that can reach the source of the

call in time, then the remaining coverage is calculated for all idle service engineers and the one with the biggest remaining coverage is dispatched. In case there are multiple service engineers maximizing the remaining coverage, the closest one is dispatched.

DP3. Note that the coverage only estimates the performance of the system for the next call. Alternatively, one can calculate the *expected covered demand*, the fraction of calls that will be answered in time by the system. As the expected covered demand is hard to compute, for computational study we use the approximation described in Appendix 4.8. Similar to the coverage based approach, first we calculate the remaining expected covered demand after dispatching each of the idle service engineers that can reach the broken machine in time, and dispatch the one with the highest remaining expected covered demand. If there are no idle service engineers that can reach the machine in time, we instead do this procedure for all idle service engineers.

4.4.2 Dispatching Policies with Waiting

Under the dispatching policies discussed in the previous subsection, an idle service engineer is always dispatched when a call arrives. However, in practice it may occur that a service engineer close to the new breakdown will finish its repair earlier than any idle service engineer can reach the breakdown, in which case it may be better to wait for the busy service engineer to finish, and dispatch him/her instead.

Inspired by this, we extend the closest-first dispatching policy to the dispatching policy that chooses a service engineer with the smallest response time. For a service engineer m whose destination is a base station r (i.e., $l_m = r$) the response time to a call from machine k is calculated as $rt(k, m) = d_m + d_{l_m k}$. For a service engineer m whose destination is machine k' (i.e., $l_m = k'$) the response time consists of the distance left to machine k', the length of repair and the distance from machine k' to machine k. Then the response time equals $rt(k, m) = d_m + t_{repair} + d_{k'k}$. We consider two situations: when the length of repair can be estimated upon arrival of a service engineer to the machine (**DP5**) and when it stays unknown (**DP4**). If the length of repair t_{repair} is not known, it can be estimated from its distribution. We estimated it by an α^{th} percentile of the repair time distribution. Throughout the computational study we take $\alpha = 80\%$.

In the extended closest-first policy the service engineer $m = \arg\min_n rt(k, n)$, that minimizes response time, is assigned to the call. If $m \in \mathcal{F}(s)$ then the service engineer is dispatched immediately. If the service engineer m is busy, then the call is placed in the queue.

4.5 Relocation Heuristics

The relocation policy is responsible for the location of idle service engineers. The simplest relocation policy is the static policy, where each service engineer is assigned to a base station and resides there when idle. In Appendix 4.8 we discuss how to compute the static allocation of the service engineers that maximizes the expected covered demand. We assume that this policy is used when the dispatching policies are studied in isolation. However, when a service engineer is dispatched to a call, a large area of the region may become uncovered and it may be beneficial to reallocate other idle service engineers. In our system we allow one idle service engineer to change the destination either when one of the service engineers is dispatched or finishes a repair.

For our comparative study we choose the following five heuristic relocation policies:

RP1: Static policy;

RP2: MCRP compliance tables;

RP3: MEXCRP compliance tables;

RP4: DMEXCLP heuristic without constraints;

RP5: DMEXCLP heuristic with constraints.

In this section we consider four relocation policies, in addition to the static policy (RP1). Policies RP2 and RP3 are compliance tables constructed according to two different algorithms. The compliance table relocation policy is the one where location of service engineers depends only on the number of idle service engineers. The locations of the idle engineers are ignored, as well as the state of the machines. This allows us to reduce the number of considered situations and precompute relocation actions for larger systems. We present these approaches in Sections 4.5.1 and 4.5.2, respectively. Policies RP4 and RP5 are heuristic relocation policies based on the DMEXCLP heuristic introduced in [47]. The main difference between this heuristic and compliance tables is that decisions are made in real-time, that allows to use the information about the current location of service engineers and the state of the machines. We construct two versions of the DMEXCLP heuristic: one with no restrictions and one with restrictions on relocation, in Section 4.5.3.

4.5.1 MCRP Compliance Tables

The first type of compliance table is the Maximum Coverage Relocation Problem (MCRP) compliance table that aims to maximize the probability that the next call is answered in time. Consider a system with M service engineers. Then a compliance table consists of

M levels, where level m contains the allocation solution for m idle service engineers in the system. If one of them is dispatched, other service engineers are relocated according to level $m-1$. If one of the service engineers becomes idle after finishing a repair, then the idle service engineers are relocated according to level $m+1$.

Denote by z_{mk} the indicator of the fact that at the level with m idle service engineers the machine k is covered (meaning that at least one service engineer can reach it in time), Then at the level m we want to maximize $\sum_{k=1}^{K} z_{mk}$, i.e., the number of covered machines. If x_{mr} is the number of service engineers at the base station r at level m, then $z_{mk} \leq \sum_{r \in N_k} x_{mr},\ k = 1, \ldots, K$, where N_k is the set of all base stations from which the machine k can be reached in time.

Recall that we allow to relocate at most one service engineer at a time. To include this restriction in the ILP formulation, we introduce non-negative variables α_{mr} that represent the number of service engineers that arrived at base station r after going from level $m+1$ to level m. Then $x_{mr} - x_{m+1,r} \leq \alpha_{mr},\ r = 1, \ldots, R,\ m = 1, \ldots, M-1$, and $\sum_{r=1}^{R} \alpha_{mr} \leq 1,\ r = 1, \ldots, R,\ m = 1, \ldots, M-1$. As these constraints connect different levels of the compliance table, the ILP problems cannot be solved separately for each level, and we construct an ILP formulation for the whole table. Let S_m be the event of having m idle service engineers in the system. Then the objective function can be formulated as $\sum_{m=1}^{M} \mathbb{P}(S_m) \sum_{k=1}^{K} z_{mk}$. An accurate approximation of probabilities $\mathbb{P}(S_m)$ is provided in Appendix 4.8, equations (4.23). Finally, we are in position to provide the formulation for the MCRP compliance table:

$$\max \quad \sum_{m \in \mathcal{M}} \mathbb{P}(S_m) \sum_{k \in \mathcal{K}} z_{mk} \tag{4.5}$$

$$\text{s.t.} \quad z_{mk} \leq \sum_{r \in N_k} x_{mr}, \qquad k \in \mathcal{K}, m = \mathcal{M} \tag{4.6}$$

$$\sum_{r \in \mathcal{R}} x_{mr} = m, \qquad m \in \mathcal{M} \tag{4.7}$$

$$x_{mr} - x_{m+1,r} \leq \alpha_{mr}, \qquad r \in \mathcal{R}, \quad m = 1, \ldots, M-1 \tag{4.8}$$

$$\sum_{r \in \mathcal{R}} \alpha_{mr} \leq 1, \qquad m = 1, \ldots, M-1 \tag{4.9}$$

$$\alpha_{mr} \geq 0, \qquad r \in \mathcal{R}, \quad m = 1, \ldots, M-1 \tag{4.10}$$

$$x_{mr} \in \{0, 1, \ldots, m\}, \qquad r \in \mathcal{R}, \quad m \in \mathcal{M} \tag{4.11}$$

$$z_{mk} \in \{0, 1\}, \qquad k \in \mathcal{K}, \quad m \in \mathcal{M} \tag{4.12}$$

4.5.2 MEXCRP Compliance Tables

Consider a system where the number of idle service engineers is larger than the number required to cover all machines. In the MCRP approach presented in Section 4.5.1, if each machine is covered by a service engineer, the location of the remaining service engineers does not affect the coverage. This may lead to inefficient allocation of service engineers. The MEXCRP compliance tables introduced in [108] can solve this problem. In this algorithm, the main goal is to optimize expected covered demand, not the number of covered machines.

The problem can again be formulated as an ILP. Let the binary variable y_{mki}, $m = 1, \ldots, M$, $k = 1, \ldots, K$, $i = 1, \ldots, m$, equal 1 if and only if in the configuration for m idle service engineers machine k is covered by at least i service engineers. Denote by P_{mki}, $m = 1, \ldots, M$, $k = 1, \ldots, K$, $i = 1, \ldots, m$, the probability that in the configuration for m idle service engineers a call from machine k is responded to by the ith closest service engineer. For a given number of idle service engineers m, this probability can be approximated using equation (4.27) in Appendix 4.8. The MEXCRP compliance table formulation is as follows:

$$\max \quad \sum_{m \in \mathcal{M}} \sum_{k \in \mathcal{K}} \sum_{i=1}^{m} P_{mki} y_{mki} \tag{4.13}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} y_{mki} \leq \sum_{r \in N_k} x_{mr}, \qquad k \in \mathcal{K}, \quad m \in \mathcal{M} \tag{4.14}$$

$$\sum_{r \in \mathcal{R}} x_{mr} \leq m, \qquad m \in \mathcal{M} \tag{4.15}$$

$$x_{mr} - x_{m+1,r} \leq \alpha_{mr}, \qquad r \in \mathcal{R}, \quad m = 1, \ldots, M-1 \tag{4.16}$$

$$\sum_{r=1}^{R} \alpha_{mr} \leq 1, \qquad m = 1, \ldots, M-1 \tag{4.17}$$

$$\alpha_{mr} \geq 0, \qquad r \in \mathcal{R}, \quad m = 1, \ldots, M-1 \tag{4.18}$$

$$x_{mr} \in \{0, 1, \ldots, m\}, \qquad r \in \mathcal{R}, \quad m \in \mathcal{M} \tag{4.19}$$

$$y_{mki} \in \{0, 1\}, \qquad k \in \mathcal{K}, \quad m \in \mathcal{M} \tag{4.20}$$

4.5.3 DMEXCLP Heuristics

When compliance table relocation policy is used, the state that is achieved after relocation does not depend on the current state of the system, only on the number of idle service engineers. In contrast, the DMEXCLP heuristic uses all the information about the current state to make a decision. This approach is more flexible than the compliance tables. We consider the DMEXCLP heuristic relocation policy introduced in [47] and adjust it for our model.

There are two types of *decision moments*:

1. When a service is completed and there are no jobs assigned to the service engineer that just became idle. In this case, that service engineer must be allocated to one of the base stations.

2. When an idle service engineer is dispatched to an incident, it should be decided whether one of the other idle service engineers should be relocated or not.

According to the DMEXCLP relocation policy, the action that maximizes the expected covered demand is always chosen. In the first case, one service engineer is added sequentially to every base station, the expected covered demand is calculated, and the base station that leads to the best result is chosen. In the second case, all pairs of base stations (r_1, r_2), with at least one service engineer at the base station r_1, are considered. We calculate the improvement in the expected covered demand after the relocation of a service engineer from station r_1 to station r_2. Suppose that (r_1', r_2') is the pair with the maximum improvement. If this improvement is positive, then we decide to relocate a service engineer from station r_1' to station r_2'. If the maximum improvement is not positive, then no relocation happens. Note that for both situations the expected covered demand is computed only for the working machines.

The problem of large relocation times leading to the possibly poor performance can appear. In [111] the authors impose restrictions on the relocations as a solution to this problem. There are three possible parameters that can be used to describe these restrictions. In the first type of decision moment the maximum relocation distance can be set. In this case, the best station is chosen among all stations within this distance from the machine where the service engineer is located. If there are no such base stations then the choice is made between all base stations. In the second type of decision moment the restriction can be imposed not only on the maximum relocation distance, but also on the minimum improvement in the expected covered demand. If the maximum relocation distance is set, than only the pairs (r_1, r_2) with smaller distance are considered. If there are no such pairs, the relocation is forbidden. Note that this distance can differ from the

distance for the first type of decision. If the minimum improvement threshold is set, then the relocation happens only if the improvement in expected covered demand exceeds this threshold. Setting this threshold to 0 means no restrictions are imposed. Choosing a threshold larger than the number of machines leads to no relocation.

The optimal restriction parameters depend on the type of the system. However, there are no known results on how to find the optimal parameters for a given system. In the computational study in Section 4.6.3 below we consider this policy with different parameters and study the improvement that can be gained by parameter tuning.

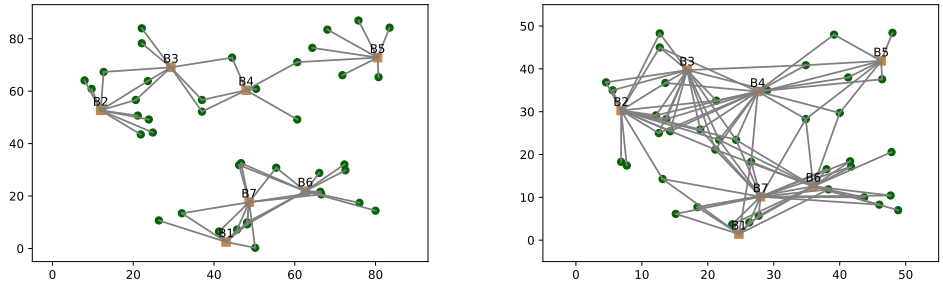
4.6 Numerical Experiments

In this section we present the setup and the results of our numerical experiments. To compare different policies we use simulation. In Section 4.6.1 we describe the type of the systems we use in our simulations, and the parameters defining the properties of the systems that affect the policies' performance. Section 4.6.2 presents the results comparing the dispatching policies introduced in Section 4.4 to each other with no relocation allowed. Next, in Section 4.6.3 the relocation policies from Section 4.5 are compared to each other, given a fixed dispatching policy. Finally, in Section 4.6.4 a heuristic policy based on the combination of the best dispatching and relocation policy from the previous sections is compared against the optimal policy for a small problem instance. We present the results of numerical experiments in tables for a limited set of parameter values. In the Appendix 4.8 we provide extended tables for a wider range of systems.

4.6.1 Setup of the Numerical Experiments

In our computational experiments, the networks are generated randomly as follows. First, we define a square on a Euclidean plane and generate the coordinates of the base stations within that square uniformly at random. After that, the coordinates of the machines are sampled at random such that each machine is within the distance of t^* units from at least one station. We only consider networks where each base station covers at least one machine. After the locations of the machines and the base stations are determined, the service engineers are allocated to the base stations such that the expected covered demand is maximized. This is done by solving an integer program that we formulate in Appendix 4.8.

The size of the square is determined by the combination of the time limit t^* and the density parameter d. The higher the density for a given value of t^*, the smaller the square, meaning that the map is more dense with each node covered by more stations on average. An example of how d affects the network structure can be seen in Figure 4.2.



(a) $d = 0.3$ (b) $d = 2$

Figure 4.2: Networks of different density d, given $t^* = 20$, $K = 40$, $R = 10$

The two maps are randomly generated for two values of d and the same values of t^*, R and K. The base stations are connected with an edge to the machines they cover. The map with low value of d is more sparse. In sparse networks, the machines are covered by fewer stations, and the distances are larger compared to t^*. On the one hand, if the distances are large, relocation might be not desirable as it takes more time. On the other hand, in sparse networks the same station may cover multiple machines if there are more machines than stations. In that case, relocation from a full station to an empty one upon a incident may be beneficial for the system performance. The map with high value of d has smaller distances and more connecting edges. If the map is dense, then the machines are located closer to each other, meaning that a busy service engineer may potentially respond quicker to a nearby incident than the closest idle one. In that case, the optimal policy might be putting the new incident in the queue instead of dispatching an engineer immediately.

The other two parameters affecting the policy performance are the breakdown rate λ and the service rate μ. These two parameters together define the load of the system; that is, the fraction of time the service engineers are busy responding to incidents. In our experiments, we fix the parameter λ and control the load via μ. The larger μ the faster the service engineers fix the breakdowns, and hence, the lower the load.

4.6.2 Comparison of Dispatching Heuristics

In this section we show the performance of the policies described in Section 4.4. To this end, we generate systems with different parameters and compare the fraction of calls answered in time under each of the five policies DP1-DP5. Relocation is not allowed,

meaning that each service engineer is allocated to a fixed base station according to the model in Appendix 4.8, and returns there after repairing a machine. The starting state for all policies is the same, chosen to maximize the expected covered demand.

We test the policies in a simulation. The number of machines, the number of bases and the failure rate are fixed at $K = 20$, $R = 12$ and $\lambda = 0.01$, respectively. We then change the number of engineers M, the map density d, the time limit t^*, and the the repair rate μ. Those parameters in combination control the geographical structure of the service region and the load. For each combination of parameters we randomly generate 10 different maps, and run a simulation over a time horizon of 1000 time units. For our first experiment we compare the policies DP1, DP2 and DP3 used on the same maps. For each simulation run, we measure the fraction of calls responded to within the time limit t^*. Table 4.1 contains the obtained results for a range of parameter values. One can see that the three policies perform very close to each other for all systems, and there is no policy that performs the best across all instances. The performance decreases with load (the load increases for larger M and lower μ). We also observe that the performance of all three policies decreases with the increase of t^* for a given density d. This effect is particularly pronounced when d and the load are small. Increasing the time limit t^* for a fixed density d results in the maps with larger distances compared to the average repair time.

M	d	t^*	$\mu = 0.2$			$\mu = 0.05$		
			DP1	**DP2**	**DP3**	**DP1**	**DP2**	**DP3**
10	**0.3**	**5**	0.92	0.92	0.93	0.78	0.79	0.78
		20	0.38	0.36	0.37	0.29	0.28	0.28
	2	**5**	0.99	1.00	1.00	0.94	0.95	0.95
		20	0.87	0.86	0.86	0.73	0.72	0.73
13	**0.3**	**5**	0.98	0.98	0.98	0.92	0.93	0.94
		20	0.84	0.82	0.82	0.64	0.60	0.62
	2	**5**	1.00	1.00	1.00	0.99	0.99	0.99
		20	0.98	0.98	0.98	0.93	0.94	0.94

Table 4.1: Fraction of calls answered in time for the policies DP1, DP2 and DP3

As the difference in performance between the first three policies is negligible, we then compare the simplest policy DP1 against the policies DP4 and DP5 in a separate experiment. Again, the three policies DP1, DP4 and DP5 are used in a simulation run on 10 randomly generated maps for each combination of parameter values. The obtained

simulation results can be found in Table 4.2. For most of the systems, both policies with waiting outperform (or at least perform as well as) the traditional closest-first policy. The maximum relative improvement is over 50%. It increases with the decrease in map density and load, and increase in traveling times (recall that the traveling times increase with t^* for a given d). In dense maps it is more likely that the machines are close to each other, and waiting can be beneficial only if the service times are relatively low compared to the traveling times. Comparing the performance of policies DP4 and DP5, one can see that including the remaining repair time leads to an improved performance, although marginal. Policy DP4, however, is probably more realistic, as it does not assume that the repair times are known up front, just their distributions.

M	d	t^*	$\mu = 0.2$			$\mu = 0.05$		
			DP1	**DP4**	**DP5**	**DP1**	**DP4**	**DP5**
10	**0.3**	**5**	0.92	0.95	0.96	0.79	0.81	0.83
		20	0.37	0.83	0.85	0.31	0.54	0.64
	2	**5**	0.99	0.99	0.99	0.94	0.95	0.95
		20	0.86	0.97	0.97	0.72	0.79	0.88
13	**0.3**	**5**	0.98	0.98	0.98	0.92	0.93	0.92
		20	0.77	0.92	0.94	0.67	0.79	0.84
	2	**5**	1.00	0.99	0.99	0.99	0.98	0.99
		20	0.98	0.95	0.98	0.94	0.93	0.96

Table 4.2: Fraction of calls answered in time for the policies DP1, DP4 and DP5

4.6.3 Comparison of Relocation Heuristics

In this section we present simulation results comparing different relocation policies. The dispatching policy is fixed to the policy DP4 as it is the best performing policy from Section 4.6.2.

As mentioned in Section 4.5.3 above, there are three parameters that define restrictions for the policy RP5. Those are the maximum relocation distance upon redeployment after repair is finished, the maximum relocation distance upon dispatching, and the minimum performance improvement for relocation upon dispatching. To fit these parameters for the RP5 policy, we simulated the system for the first and the second parameters equal to $0.5t^*$, t^*, $2t^*$ and $100t^*$, and the third parameter equal to 0, 1, 5 and 100. The best result for each type of the system was chosen and used as an input for the policy RP5.

d	t^*	$\mu = 0.2$					$\mu = 0.05$				
		RP1	**RP2**	**RP3**	**RP4**	**RP5**	**RP1**	**RP2**	**RP3**	**RP4**	**RP5**
0.3	**5**	0.93	0.69	0.71	0.89	0.98	0.76	0.70	0.70	0.85	0.95
	20	0.77	0.43	0.43	0.48	0.92	0.52	0.25	0.25	0.43	0.82
2	**5**	0.97	0.87	0.89	0.99	0.99	0.87	0.92	0.96	0.98	0.99
	20	0.92	0.74	0.79	0.79	0.97	0.64	0.49	0.55	0.75	0.96

Table 4.3: Fraction of calls answered in time for the policies RP1-RP5 combined with DP4

The relocation policies are compared using simulation. We set $K = 20$, $R = 10$ and $\lambda = 0.01$. This time we also fix the number of service engineers equal to $M = 13$. For each type of the system we generate 10 random maps, run simulation for each of the maps, and measure the fraction of calls responded to within the time limit for each of the five policies. The results can be found in Table 4.3. One can see that compliance tables (RP2 and RP3) demonstrate poor performance for most systems. The only type of the systems for which RP2 and RP3 perform better than the static policy RP1 is those with high density and repair times much larger than t^*. The reason is that both the MCRP and MEXCRP algorithms ignore distances, so when these are large, those policies lead to inefficient relocations and poor performance.

Policy RP4 outperforms compliance tables for all systems because it uses more information about the state of the system to make relocation decisions. However, it also ignores the distances, so for maps with large distances and small density we observe that it performs worse than the static policy. Finally, policy RP5 leads to good results for all of the systems. The fraction of calls answered in time stays above 80%, even for the systems with high load, where all other policies result in less than 60% of calls answered in time. The difference in performance between RP5 and RP4 shows the importance of relocation restrictions and accurate tuning of the parameters of these restrictions.

4.6.4 Optimal Policy Performance

In this section, we benchmark the best performing heuristic against the optimal policy. Heuristic policy is combined of the dispatching policy DP4 and the relocation policy RP5 (i.e., minimal response time + DMEXCLP). To obtain the optimal policy, we use the discrete-time model described in Appendix 4.8. Due to computational complexity of finding the optimal policy, we do this only for one small system depicted on Figure 4.3. There are two base stations, two service engineers, and four machines. The failure rate is $\lambda = 0.01$ and the repair rate is 1. The time limit is set to $t^* = 3$.

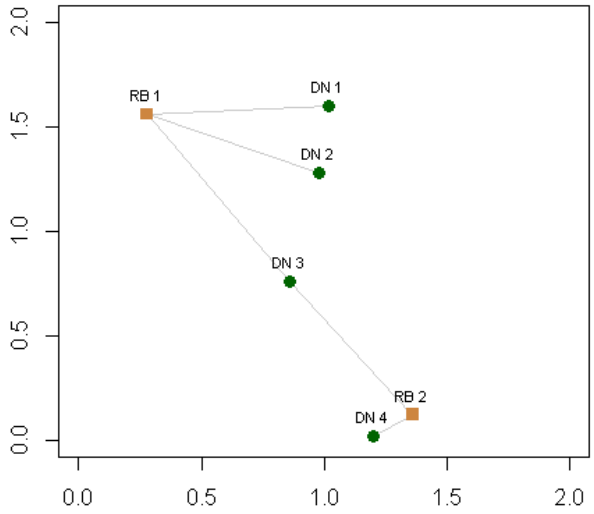


Figure 4.3: Sample map with two base stations (RB) and four machines (DN)

We use policy iteration to derive the optimal policy. Then we run 10 iterations of simulation under the optimal policy and under the heuristic. The obtained average fraction of calls answered in time is 0.87 for the optimal policy and 0.82 under the DP4+RP5 heuristic policy, which is 5.7% less than the optimal performance. We see that the heuristic performs close to the optimal at least for this instance. Unlike the optimal policy, however, it can be easily derived for real-life systems with significantly larger state spaces.

4.7 Conclusion

In this chapter, we studied the problem of real-time management of service engineers. We drew inspiration from the vast research in EMS domain to develop a number of scalable heuristics that lead to a low number of late arrivals to the emergency calls. We compared the performance of multiple heuristics, examining if there is any that works well irrespective of the network structure. We conducted extensive computational experiments where a range of dispatching and relocation policies were compared against each other for various types of systems. One of the best combined policies was then

tested in a simulation against the optimal policy showing close to the optimal performance.

The dispatching policies were compared with the relocation policy fixed to a static one, where idle service engineers reside at preassigned base stations. Five dispatching policies were compared in a simulation for systems with different parameters. Parameters defining the system include the number of service engineers, the repair rate, the time limit and the map density. The policy that assigns the service engineers with the smallest response time outperformed other policies for most of the systems we considered. We showed that it may be beneficial not to dispatch an idle service engineer immediately upon a break down of a machine, but wait for another service engineer to finish repair. We also demonstrated that accurate estimation of repair time can lead to improvement in performance, although only for the systems with high load.

The performance of five relocation policies was measured with the fixed dispatching policy that always chooses the service engineer with the minimal response time. For most of the systems, compliance tables performed worse than the closest-first policy, except for those where the distances are small compared to the average repair time. The numerical results favor the DMEXCLP relocation policy. However, its performance depends on the choice of the restriction parameters, such as the maximum distance of relocation and the minimum improvement in the expected covered demand. Without these restrictions it performs worse than the closest-first policy for systems with large distances. However, when the restriction parameters are carefully chosen, it outperforms other policies by up to 60% in terms of fraction of calls answered in time.

4.8 Appendices

In this section we provide supplementary materials to the main body of the chapter.

4.8.1 Discrete-Time Model Formulation

In this section, we construct a discrete-time approximation of the original process formulated in Section 4.3. To that end, we discretize time and the service region. The service region is approximated by rounding all distances to the integers. Recall that for the continuous time model, we look at the state of the process at each moment an event happens. In the discrete-time model we look at the state of the process at each time point, where several events can happen between subsequent time points. The resulting process has a finite state space that allows us to perform the policy iteration algorithm. In practice, the length of the time unit is an important decision to make. Small duration of the time unit provides good approximation of the real process, however it may also lead to a large state space that is computationally intractable.

We then define the state space based on the continuous-time model, ensuring it is finite. For the continuous-time process the state is described by a tuple $s = (t, e, \eta, \kappa)$, where t is the time, e is the event, η is the state of the service engineers and κ is the state of the machines. For the discrete-time process, we look at the state of the process at each time unit. So the time in state s_n is deterministic $t(s_n) = t_n = n$, and the optimal action in state s should not depend on time $t(s)$. We then omit the time component of state in the discrete-time version of the process. The vector η contains the pairs of destinations and distances to those destinations for each service engineer. For the discrete-time model, however, distances take only integer values, so there is only a finite number of possible locations of service engineers. Vector κ contains the state of each of the machines: the waiting time if the machine is broken, 0 if it is working, and -1 if it is in repair. To have a finite number of possible states for each machine, the waiting time is bounded by the time limit t^*. If the waiting time of a broken machine k reaches t^*, its state remains unchanged ($\kappa_k = t^*$) until the repair starts.

In the discrete-time process more than one event can happen during the transition from state s_n to state s_{n+1} (for example, two machines may break down). Hence, we define the event e as a triplet of sets $e = (\mathcal{K}_1, \mathcal{K}_2, \mathcal{M}_a)$, where $\mathcal{K}_1$ is the set of machines that got broken during the last time unit, $\mathcal{K}_2$ is the set of machines that got repaired during the last time unit, and $\mathcal{M}_a$ is the set of service engineers that arrived to their destinations during the last time unit. It is possible that no events happened during the transition, and all three sets are empty.

Transitions

To determine transition probabilities, we need to derive the probability of a certain event $e = (\mathcal{K}_1, \mathcal{K}_2, \mathcal{M}_a)$ happening in a given state s. This probability depends only on the first two components, as the last component $\mathcal{M}_a$ is deterministic and depends only on the previous state of the system. The probability that a working machine will break down during one time unit is $p = 1 - e^{-\lambda}$, and the probability that a repair in progress will end during one time unit is $q = 1 - e^{-\mu}$. Denote by $W(s)$ the number of all working machines in state s, and by $H(s)$ the number of machines in repair. Then the probability of event $e = (\mathcal{K}_1, \mathcal{K}_2, \mathcal{M}_a)$ happening in state s is equal to

$$P(s, e) = P(s, \mathcal{K}_1, \mathcal{K}_2) = p^{|\mathcal{K}_1|}(1-p)^{W(s)-|\mathcal{K}_1|} q^{|\mathcal{K}_2|}(1-q)^{H(s)-|\mathcal{K}_2|}.$$

The next state s_{n+1} of the process depends only on the current state s_n of the process, the action a_n taken, and the random components of the event ($\mathcal{K}_1$ and $\mathcal{K}_2$):

$$s_{n+1} = \Phi\left(s_n, a_n, \mathcal{K}_1, \mathcal{K}_2\right).$$

Actions

Let $\mathcal{F}(s) = \{m \in 1, \ldots, M \mid l_m(s) \in \mathcal{R}\}$ denote the set of all idle service engineers, and $Q(s)$ the set of all machines in the queue in state s. Note that $\mathcal{M}_a \subseteq \mathcal{F}(s)$ and $\mathcal{K}_1 \subseteq Q(s)$. An action a in state s consists of three binary matrices $\boldsymbol{X}, \boldsymbol{Y}$ and $\boldsymbol{Z}$. Matrix $\boldsymbol{X}$ describes the dispatching decision for all machines in the set $Q(s)$. Let $X_{mk} = 1$ if service engineer m is dispatched to machine k, and $X_{mk} = 0$ otherwise. Matrix $\boldsymbol{Y}$ describes the redeployment decision for the service engineers that finished repairing machines in the set $\mathcal{K}_2$. Let $Y_{ml} = 1$ if service engineers m is redeployed to location l, and $Y_{ml} = 0$ otherwise. Matrix $\boldsymbol{Z}$ describes the relocation decision. Let $Z_{mr} = 1$ if service engineer m is relocated to base station r, and $Z_{mr} = 0$ otherwise. The action space in state s is given by

$$\begin{aligned}\mathcal{A}(s) = \Bigg\{(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}) \mid & \sum_{m \in \mathcal{F}(s)} X_{mk} + \sum_{m \in \mathcal{M}: l_m(s) \in \mathcal{K}_2} Y_{mk} \leq 1 \ \forall k \in Q(s); \\ & \sum_{l \in Q(s) \cup \mathcal{R}} Y_{ml} = 1 \ \forall m \in \mathcal{M} : l_m(s) \in \mathcal{K}_2; \\ & \sum_{\substack{m \in \mathcal{F}(s), \\ r \in \mathcal{R}}} Z_{mr} \leq 1; \quad \sum_{\substack{m \in \mathcal{F}(s), \\ r \in \mathcal{R}, \\ k \in Q(s)}} X_{mk} Z_{mr} = 0; \\ & \mathbb{I}\left(\textstyle\sum^{m \in \mathcal{F}(s),}_{k \in \mathcal{K}_1} X_{mk} = 0\right) \mathbb{I}(\mathcal{K}_2 = \emptyset) \sum_{\substack{m \in \mathcal{F}(s), \\ r \in \mathcal{R}}} Z_{mr} = 0\Bigg\},\end{aligned}$$

where the constraints ensure that not more than one service engineer is dispatched to each broken machine, each service engineer that finished a job is redeployed to exactly one location, at most one relocation is made and the relocated service engineer differs from the dispatched ones, relocation is done only if at least one service engineer was dispatched to one of the newly broken machines or at least one service engineer finished a repair.

Costs

We apply the following cost structure. If a call arrives from machine k and a service engineer does not reach this machine within the time limit t^*, a cost 1 is incurred. Moreover, a small cost $0 < \epsilon \ll 1$ is incurred per time unit of waiting for service over t^*. This second penalty is used to ensure dispatching is done by the optimal policy. Our goal is to find actions that minimize the long-run discounted penalty. All travel costs and other operational costs are ignored, but could be readily added to the model.

Denote by $c(s_n, a_n, s_{n+1})$ the costs incurred during the transition period from state s_n to state s_{n+1} when action a_n is taken. The first component is a unit penalty for each machine who's waiting time exceeds t^* during transition. The extra costs are incurred for the total waiting time over t^*. Then in total

$$\begin{aligned} c(s_n, a_n, s_{n+1}) =& \left| \{k \in \mathcal{K} \mid \kappa_k(s_n) < t^* \text{ and } \kappa_k(s_{n+1}) = t^*\} \right| \\ &+ \epsilon \left| \{k \in \mathcal{K} \mid \kappa_k(s_n) = t^*\} \right|. \end{aligned}$$

It is important to note that our goal is to maximize the fraction of calls answered in time. The penalty ϵ is introduced only to prevent the situation of leaving some machines broken forever, which would otherwise be optimal. So ϵ is set to be small. In the computational experiments, we set $\epsilon = 0.001$, so it does not affect the optimal policy.

Optimality equations

Consider a discounted version of the process $\{s_n,\ n = 0, 1, \dots\}$ with discount factor γ [89]. Denote by $V_\pi(s)$ the expected total discounted costs under policy π when starting in state s:

$$V_\pi(s) = \mathbb{E}\left[\sum_{n=0}^{\infty} \gamma^n c(s_n, \pi(s_n), s_{n+1}) \mid s_0 = s\right].$$

If policy π^* is the optimal policy, then $V_{\pi^*}(s)$ satisfies the Bellman optimality equation:

$$V_{\pi^*}(s) = \min_{a \in \mathcal{A}(s)} \left\{\mathbb{E}_a \left[c(s, a, s') + \gamma V_{\pi^*}(s')\right]\right\}, \quad \forall s \in S,$$

where $s' = \Phi(s, \pi(s), \omega(s, a))$ is the next state of the process when action a is taken in state s, and

$$\pi^*(s) = \underset{a \in \mathcal{A}(s)}{\arg\min} \{\mathbb{E}_a [c(s, a, s') + \gamma V_\pi(s')]\}, \quad \forall s \in S.$$

Due to the curse of dimensionality, finding the optimal policy π^* is computationally intractable for realistic-sized systems. Hence, in Sections 4.4 and 4.5 we focus on various scalable heuristic approaches to the above problem.

The state of the discrete-time process is represented by a triplet $s = (e, \eta, \kappa)$. As the set of possible states of service engineers and machines, and the set of possible events are all finite, the state space of the process is finite. As the transitions and the costs depend only on the current state and the random component, the resulting process is a finite-state Markov decision process.

4.8.2 Expected Covered Demand Approximation

In this section we derive an approximation of the expected covered demand, where we follow the procedure introduced by Larson [63, 64], and apply it to the model described in Section 4.3. Given the locations of the service engineers, the expected covered demand estimates the long-term fraction of calls that will be answered in time. We use this metric for several dispatching and relocation policies (see Sections 4.4 and 4.5).

First, we consider the process $C = \{C_n,\ n = 1, 2, \dots\}$, that approximates the number of broken machines in the n^{th} state of the original process s_n, $n = 1, 2, \dots$, where $C_n = |\{k \in \mathcal{K} \mid \kappa_k(s_n) \neq 0 \text{ or } e = \text{'machine } k \text{ breaks down'}\}|$. We compute the steady-state distribution of this process.

The time until breakdown of a machine is exponentially distributed with rate λ. The time a machine stays broken is also exponentially distributed with rate $\hat{\mu}$. This time includes the traveling time of a service engineers and the duration of the repair. If all service engineers are busy, and the machine is put in the queue first, then the waiting time in the queue is not included.

The process C is a Markov process. The state space of the process is $\{0, 1, \dots, K\}$ (see Figure 4.4), so it is a finite-state process. There are two possible transitions from a state C_n with k broken machines:

- The event in state s_{n+1} is of type *"a machine breaks down"*. Then $C_{n+1} = k + 1$. The rate of this transition is equal to $\lambda(K - k)$. (This transition is not possible if $k = K$.)

- The event in state s_{n+1} is of type *"a repair ends"*. Then $C_{n+1} = k - 1$. The rate of this transition $\hat{\mu} \cdot \#$ machines in repair $= \hat{\mu} \cdot \min\{k, M\}$. (This transition is not possible if $k = 0$.)

Let $P(k)$ denote the stationary probability of being in state k. Then the balance equations for C_n can be formulated as follows:

$$\begin{aligned}
&\lambda K P(0) = \hat{\mu} P(1), \\
&(\lambda(K-k) + \hat{\mu}k)\, P(k) = \lambda(K-k+1)P(k-1) + \hat{\mu}(k+1)P(k+1), && k = 2, \dots, M-1, \\
&(\lambda(K-k) + \hat{\mu}M)\, P(k) = \lambda(K-k+1)P(k-1) + \hat{\mu}MP(k+1), && k = M, \dots, K-1, \\
&\hat{\mu}MP(K) = \lambda P(K-1).
\end{aligned}$$

One can check that

$$P(k) = \begin{cases} \binom{K}{k} \left(\frac{\lambda}{\hat{\mu}}\right)^k P(0), & k = 0, \dots, M-1, \\ \frac{k!}{M! M^{k-M}} \binom{K}{k} \left(\frac{\lambda}{\hat{\mu}}\right)^k P(0), & k = M, \dots, K \end{cases} \tag{4.21}$$

is the solution of the balance equations. Adding the normalization equation $\sum_{k=0}^{K} P(k) = 1$ to the system we get

$$P(0) = \left[\sum_{k=0}^{M-1} \binom{K}{k} \left(\frac{\lambda}{\hat{\mu}}\right)^k + \sum_{k=M}^{K} \frac{k!}{M! M^{k-M}} \binom{K}{k} \left(\frac{\lambda}{\hat{\mu}}\right)^k \right]^{-1}. \tag{4.22}$$

Using these formulas one can calculate $P(k)$ for $k = 0, 1, \dots, K$. Now, let S_m be the event of having m busy service engineers, then

$$\mathbb{P}(S_m) = \begin{cases} P(m), & m = 1, \dots, M-1, \\ \sum_{k=M}^{K} P(k), & m = M. \end{cases} \tag{4.23}$$

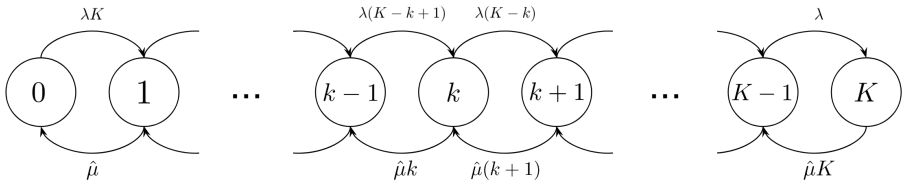


Figure 4.4: State diagram of the discrete-time process C

We consider a system where relocation is not allowed. It means that each service engineer is assigned to a base station, and returns to that base station after each repair. Suppose that the system is in state s where all service engineers are at their base stations. Assume that the dispatching policy is fixed per machine, such that if machine k breaks down, we first send the service engineer $m_1^{(k)}$. Then, if that service engineer is busy, $m_2^{(k)}$ is dispatched, and so on. Assume also that first we try to send the service engineers that can reach the machine within the time limit, and, if all of them are busy, the service engineers that cannot arrive on time. An example of such dispatching policy is the closest-first policy that always dispatches the closest service engineer.

Let us compute the probability that service engineer $m_i^{(k)}$ is dispatched to machine k, meaning that all service engineers $m_1^{(k)}, \ldots, m_{i-1}^{(k)}$ are busy. If B_i is the event that service engineer $m_i^{(k)}$ is busy, F_i is the event that service engineer $m_i^{(k)}$ is idle, and S_m is the event that there are m busy service engineers in the system, then

$$\begin{aligned}\mathbb{P}\left(B_1 \ldots B_{i-1} F_i\right) &= \sum_{m=i}^{M} \mathbb{P}\left(B_1 \ldots B_{i-1} F_i \middle| S_m\right) \mathbb{P}(S_m) \\ &= \sum_{m=i}^{M} \mathbb{P}(S_m) \mathbb{P}\left(F_i \middle| S_m B_1 \ldots B_{i-1}\right) \mathbb{P}\left(B_{i-1} \middle| S_m B_1 \ldots B_{i-2}\right) \ldots \mathbb{P}\left(B_1 | S_m\right).\end{aligned} \tag{4.24}$$

The probabilities $P(S_m)$ can be computed using equations (4.23). Other terms can be approximated by assuming that all service engineers have the same load and are independent of each other. Under these assumptions we get

$$\begin{aligned}\mathbb{P}\left(B_1 | S_m\right) &= \frac{m}{M}, \\ \mathbb{P}\left(B_2 | S_m B_1\right) &= \frac{m-1}{M-1}, \\ &\vdots \\ \mathbb{P}\left(F_i \middle| S_m B_1 \ldots B_{i-1}\right) &= 1 - \frac{m-i+1}{M-i+1} = \frac{M-m}{M-i+1}.\end{aligned} \tag{4.25}$$

Finally, we can approximate

$$\begin{aligned}
&\mathbb{P}\left(B_1 \dots B_{i-1} F_i\right) = \\
&= \sum_{m=i-1}^{M} \mathbb{P}(S_m)\mathbb{P}\left(F_i \middle| S_m B_1 \dots B_{i-1}\right)\mathbb{P}\left(B_{i-1} \middle| S_m B_1 \dots B_{i-2}\right) \dots \mathbb{P}\left(B_1 | S_m\right) \\
&\approx \sum_{m=i-1}^{M} \mathbb{P}(S_m) \cdot \frac{M-m}{M-i+1} \cdot \dots \cdot \frac{m}{M} \\
&= \sum_{m=i-1}^{M} (M-m)\,\mathbb{P}(S_m) \cdot \frac{m!(M-i)!}{(m-i+1)!M!}.
\end{aligned} \tag{4.26}$$

Let us denote

$$P_i = \sum_{m=i-1}^{M} (M-m)\,\mathbb{P}(S_m) \cdot \frac{m!(M-i)!}{(m-i+1)!M!}, \quad i = 1, \dots, M. \tag{4.27}$$

Let the binary variable z_{ki} equal 1 if the i^{th} closest service engineer to machine k can reach it in time. The probability that a call from machine k will be answered in time is $\sum_{i=1}^{M} P_i z_{ki}$, and the expected covered demand can be approximated by

$$\frac{1}{K} \sum_{k=1}^{K} \sum_{i=1}^{M} P_i z_{ki}. \tag{4.28}$$

Note that for a given system all parameters needed to calculate expression (5.1) are known, except for the parameter $\hat{\mu}$. This parameter is hard to calculate in practice as it depends on the policy. For computational study, we first assume $\hat{\mu} = 1/(t^* + 1/\mu)$ and then run several iterations of simulation to find a better approximation for $\hat{\mu}$.

4.8.3 Optimal Allocation of Service Engineers

In this section we formulate an ILP problem that finds the optimal allocation of the service engineers to the base stations maximizing the expected covered demand.

Let the decision variables x_r, $r = 1, \dots, R$, represent the number of service engineers at base station r, and z_{ki} indicate if service engineer $m_i^{(k)}$ can reach machine k in time. The objective function is the expected covered demand that we approximate with equation (5.1). The total number of service engineers is M, so $\sum_{r=1}^{R} x_r = M$. The variables z_{ki}, $k = 1, \dots, K$, $i = 1, \dots, M$, and the variables x_r, $r = 1, \dots, R$, are connected by the equation $\sum_{i=1}^{M} z_{ki} = \sum_{r \in N_k} x_r$, $k = 1, \dots, K$, where N_k is the set of

all bases from which machine k can be reached in time. The problem can be formulated as follows:

$$\begin{aligned}
\max \quad & \sum_{k=1}^{K} \sum_{i=1}^{M} P_i z_{ki} \\
\text{s.t.} \quad & \sum_{i=1}^{M} z_{ki} \leq \sum_{r \in N_k} x_r, \quad k = 1, \ldots, K, \\
& \sum_{r=1}^{R} x_r \leq M, \\
& x_r = 0, 1, 2, \ldots, \qquad r = 1, \ldots R, \\
& z_{ki} \in \{0, 1\}, \qquad k = 1, \ldots, K, \quad r = 1, \ldots, R,
\end{aligned} \tag{4.29}$$

where constraints are relaxed with inequalities. The total number of decision variables is $R + KM$ and the total number of constraints equals $K + 1$.

4.8.4 Extended Computational Results

In this section, we provide computational results for a wider range of parameter values. Tables 4.4 and 4.5 compare the different dispatching policies. The performance of the relocation policies is presented in Table 4.6.

M	d	t^*	$\mu = 0.2$			$\mu = 0.1$			$\mu = 0.05$			$\mu = 0.02$		
			DP1	**DP2**	**DP3**	**DP1**	**DP2**	**DP3**	**DP1**	**DP2**	**DP3**	**DP1**	**DP2**	**DP3**
10	**0.3**	**5**	0.92	0.92	0.93	0.88	0.88	0.89	0.78	0.79	0.78	0.44	0.44	0.42
		10	0.81	0.79	0.79	0.76	0.73	0.74	0.60	0.57	0.57	0.33	0.33	0.31
		20	0.38	0.36	0.37	0.32	0.32	0.32	0.29	0.28	0.28	0.25	0.25	0.24
		50	0.23	0.24	0.24	0.26	0.26	0.26	0.22	0.23	0.23	0.22	0.22	0.21
	1	**5**	0.96	0.97	0.97	0.94	0.95	0.95	0.89	0.90	0.89	0.61	0.62	0.60
		10	0.91	0.92	0.91	0.86	0.87	0.87	0.76	0.76	0.76	0.44	0.45	0.42
		20	0.59	0.56	0.57	0.54	0.54	0.53	0.45	0.45	0.45	0.29	0.30	0.29
		50	0.34	0.34	0.35	0.34	0.34	0.34	0.37	0.37	0.37	0.28	0.29	0.28
	2	**5**	0.99	1.00	1.00	0.98	0.98	0.98	0.94	0.95	0.95	0.75	0.75	0.73
		10	0.97	0.97	0.98	0.95	0.95	0.96	0.90	0.90	0.90	0.62	0.63	0.61
		20	0.87	0.86	0.86	0.81	0.82	0.81	0.73	0.72	0.73	0.47	0.47	0.46
		50	0.53	0.52	0.52	0.49	0.49	0.50	0.50	0.50	0.50	0.45	0.45	0.44
13	**0.3**	**5**	0.98	0.98	0.98	0.96	0.97	0.97	0.92	0.93	0.94	0.80	0.81	0.79
		10	0.94	0.94	0.95	0.92	0.93	0.93	0.88	0.89	0.89	0.68	0.68	0.67
		20	0.84	0.82	0.82	0.73	0.68	0.69	0.64	0.60	0.62	0.44	0.43	0.42
		50	0.32	0.33	0.33	0.32	0.32	0.32	0.29	0.30	0.30	0.28	0.29	0.28
	1	**5**	0.99	1.00	1.00	0.98	0.99	0.99	0.97	0.97	0.97	0.86	0.88	0.87
		10	0.98	0.98	0.99	0.96	0.96	0.97	0.94	0.95	0.95	0.80	0.82	0.80
		20	0.90	0.89	0.91	0.91	0.91	0.91	0.84	0.83	0.83	0.62	0.61	0.60
		50	0.44	0.45	0.45	0.43	0.43	0.43	0.44	0.45	0.45	0.40	0.40	0.40
	2	**5**	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.99	0.99	0.93	0.94	0.94
		10	1.00	1.00	1.00	0.99	0.99	0.99	0.98	0.98	0.98	0.90	0.91	0.91
		20	0.98	0.98	0.98	0.96	0.96	0.97	0.93	0.94	0.94	0.81	0.79	0.79
		50	0.65	0.63	0.63	0.60	0.60	0.60	0.58	0.56	0.56	0.58	0.58	0.58
16	**0.3**	**5**	0.99	0.99	0.99	0.99	0.99	0.99	0.97	0.98	0.98	0.93	0.94	0.94
		10	0.98	0.99	0.99	0.96	0.97	0.97	0.95	0.96	0.96	0.90	0.91	0.91
		20	0.94	0.95	0.96	0.92	0.92	0.93	0.91	0.90	0.91	0.81	0.81	0.80
		50	0.51	0.51	0.51	0.52	0.50	0.53	0.48	0.46	0.48	0.41	0.41	0.41
	1	**5**	1.00	1.00	1.00	0.99	1.00	1.00	0.99	0.99	1.00	0.96	0.97	0.97
		10	0.99	0.99	0.99	0.99	0.99	0.99	0.98	0.98	0.98	0.94	0.95	0.95
		20	0.97	0.98	0.98	0.97	0.97	0.97	0.95	0.96	0.97	0.89	0.90	0.90
		50	0.68	0.67	0.68	0.63	0.61	0.63	0.61	0.60	0.60	0.53	0.53	0.53
	2	**5**	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00	1.00	0.98	0.99	0.99
		10	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.98	0.98	0.99
		20	0.99	0.99	1.00	0.99	0.99	1.00	0.98	0.98	0.99	0.96	0.96	0.97
		50	0.90	0.89	0.90	0.88	0.86	0.87	0.82	0.80	0.80	0.78	0.77	0.75

Table 4.4: Fraction of calls answered in time for policies DP1, DP2 and DP3 ($K = 20$, $R = 12$, $\lambda = 0.01$)

M	d	t^*	$\mu = 0.2$			$\mu = 0.1$			$\mu = 0.05$			$\mu = 0.02$		
			DP1	**DP4**	**DP5**	**DP1**	**DP4**	**DP5**	**DP1**	**DP4**	**DP5**	**DP1**	**DP4**	**DP5**
10	**0.3**	**5**	0.92	0.95	0.96	0.88	0.90	0.92	0.79	0.81	0.83	0.48	0.49	0.48
		10	0.80	0.91	0.90	0.74	0.85	0.87	0.61	0.73	0.77	0.31	0.41	0.42
		20	0.37	0.83	0.85	0.33	0.80	0.78	0.31	0.54	0.64	0.21	0.23	0.31
		50	0.24	0.64	0.54	0.23	0.52	0.52	0.22	0.43	0.41	0.21	0.22	0.21
	1	**5**	0.97	0.97	0.98	0.94	0.94	0.96	0.86	0.87	0.88	0.63	0.59	0.59
		10	0.92	0.96	0.97	0.88	0.90	0.92	0.74	0.81	0.83	0.48	0.50	0.55
		20	0.60	0.93	0.91	0.59	0.83	0.86	0.44	0.63	0.78	0.31	0.30	0.43
		50	0.35	0.76	0.77	0.32	0.71	0.70	0.33	0.57	0.60	0.30	0.28	0.30
	2	**5**	0.99	0.99	0.99	0.98	0.98	0.99	0.94	0.95	0.95	0.72	0.73	0.73
		10	0.97	0.98	0.98	0.94	0.96	0.97	0.89	0.91	0.92	0.65	0.64	0.66
		20	0.86	0.97	0.97	0.83	0.94	0.94	0.72	0.79	0.88	0.49	0.53	0.63
		50	0.54	0.88	0.87	0.50	0.84	0.87	0.45	0.71	0.72	0.43	0.34	0.43
13	**0.3**	**5**	0.98	0.98	0.98	0.96	0.95	0.97	0.92	0.93	0.92	0.79	0.79	0.79
		10	0.94	0.97	0.97	0.93	0.94	0.94	0.88	0.90	0.91	0.69	0.73	0.73
		20	0.77	0.92	0.94	0.78	0.90	0.91	0.67	0.79	0.84	0.43	0.56	0.69
		50	0.31	0.83	0.77	0.31	0.77	0.77	0.30	0.69	0.70	0.28	0.41	0.56
	1	**5**	0.99	0.99	0.99	0.98	0.99	0.99	0.95	0.96	0.97	0.86	0.86	0.86
		10	0.97	0.99	0.98	0.97	0.96	0.97	0.94	0.93	0.95	0.81	0.78	0.81
		20	0.91	0.96	0.96	0.88	0.95	0.96	0.80	0.89	0.92	0.62	0.67	0.80
		50	0.44	0.89	0.90	0.42	0.83	0.84	0.42	0.79	0.81	0.40	0.47	0.66
	2	**5**	1.00	0.99	0.99	0.98	0.99	0.99	0.99	0.98	0.99	0.94	0.87	0.90
		10	0.99	0.98	0.99	0.99	0.98	0.99	0.97	0.96	0.97	0.91	0.85	0.91
		20	0.98	0.95	0.98	0.97	0.95	0.98	0.94	0.93	0.96	0.82	0.83	0.90
		50	0.64	0.93	0.91	0.64	0.93	0.91	0.58	0.90	0.93	0.53	0.61	0.82
16	**0.3**	**5**	0.99	0.99	0.99	0.98	0.98	0.99	0.97	0.98	0.97	0.94	0.92	0.91
		10	0.98	0.98	0.99	0.97	0.97	0.98	0.96	0.95	0.96	0.90	0.90	0.90
		20	0.94	0.97	0.97	0.92	0.94	0.96	0.90	0.93	0.95	0.81	0.81	0.89
		50	0.52	0.90	0.88	0.47	0.89	0.89	0.48	0.83	0.85	0.40	0.64	0.79
	1	**5**	1.00	1.00	1.00	0.99	0.99	0.99	0.98	0.99	0.99	0.96	0.94	0.94
		10	0.98	0.99	0.99	0.99	0.98	0.99	0.98	0.98	0.98	0.94	0.93	0.90
		20	0.98	0.98	0.98	0.97	0.96	0.97	0.94	0.94	0.97	0.89	0.87	0.91
		50	0.70	0.94	0.93	0.65	0.92	0.91	0.59	0.89	0.91	0.56	0.76	0.84
	2	**5**	1.00	1.00	0.99	0.99	0.99	0.99	1.00	0.98	0.98	0.98	0.97	0.95
		10	1.00	0.99	0.99	1.00	0.99	0.99	1.00	0.98	0.98	0.97	0.93	0.94
		20	0.99	0.98	0.98	0.99	0.96	0.99	0.99	0.95	0.97	0.95	0.90	0.94
		50	0.90	0.96	0.96	0.86	0.96	0.96	0.86	0.92	0.94	0.77	0.83	0.93

Table 4.5: Fraction of calls answered in time for policies DP1, DP4 and DP5 ($K = 20$, $R = 12$, $\lambda = 0.01$)

μ	t^*	d	RP1	RP2	RP3	RP4	RP5
0.2	**5**	0.3	0.93	0.69	0.71	0.89	0.98
		1	0.94	0.79	0.81	0.95	0.99
		2	0.97	0.87	0.89	0.99	0.99
	10	0.3	0.86	0.58	0.57	0.72	0.95
		1	0.91	0.74	0.75	0.85	0.97
		2	0.94	0.82	0.87	0.94	0.99
	20	0.3	0.77	0.43	0.43	0.48	0.92
		1	0.83	0.61	0.62	0.64	0.96
		2	0.92	0.74	0.79	0.79	0.97
0.1	**5**	0.3	0.86	0.70	0.73	0.91	0.96
		1	0.87	0.84	0.85	0.96	0.98
		2	0.94	0.92	0.95	0.98	0.99
	10	0.3	0.83	0.57	0.56	0.69	0.94
		1	0.84	0.74	0.74	0.81	0.96
		2	0.88	0.81	0.85	0.94	0.99
	20	0.3	0.70	0.32	0.31	0.47	0.91
		1	0.75	0.58	0.58	0.60	0.95
		2	0.83	0.69	0.73	0.75	0.97
0.05	**5**	0.3	0.76	0.70	0.70	0.85	0.95
		1	0.80	0.83	0.86	0.95	0.97
		2	0.87	0.92	0.96	0.98	0.99
	10	0.3	0.66	0.46	0.47	0.68	0.91
		1	0.71	0.64	0.64	0.81	0.95
		2	0.77	0.82	0.86	0.93	0.98
	20	0.3	0.52	0.25	0.25	0.43	0.82
		1	0.56	0.37	0.38	0.58	0.91
		2	0.64	0.49	0.55	0.75	0.96

Table 4.6: Fraction of calls answered in time for policies RP1-RP5 ($K = 20$, $R = 10$, $M = 13$, $\lambda = 0.01$)

5

Approximate Dynamic Programming for Real-time Dispatching and Relocation of Service Engineers

In this chapter, we continue addressing the questions of how to dispatch service engineers to breakdowns, and how to relocate idle engineers between base stations. We develop an Approximate Dynamic Programming (ADP) approach to produce dispatching and relocation policies, and propose two new algorithms to tune the ADP policy. We conduct extensive computational experiments to compare the ADP policy against two benchmark policies by means of simulation. These demonstrate that the ADP approach can generate high-quality solutions that outperform both benchmarks across a wide range of networks and parameters. We observe significant improvements in terms of fraction of late arrivals over the two benchmarks, without increasing the average response time.

The work in this chapter is based on [104]: D. Usanov, A. Pechina, P.M. van de Ven, and R.D. van der Mei. Approximate Dynamic Programming for Real-time Dispatching and Relocation of Emergency Service Engineers (2019). Manuscript submitted for publication.

5.1 Introduction

As in Chapter 4, we are interested in the question of how to manage service engineers in the most efficient way. This includes decisions on which engineer to dispatch to a new maintenance job, whether to relocate engineers to different base stations when gaps in the coverage arise, and whether to hold off on certain demand in order to wait for nearby service engineers to complete their existing job. In this chapter we develop another heuristic that is based on Approximate Dynamic Programming (ADP).

In Chapter 4 we considered a general network structure and compared a wide range of heuristics taken from the research literature on a closely related problem of dispatching and relocation of EMS. We identified the heuristic with the best performance in almost all cases. In our work we use that algorithm as one of the benchmarks.

ADP relies on approximating the value function found in MDPs by a linear combination of easily computable basis functions. When the basis functions and their coefficients are selected carefully, such an approximation may yield sub-optimal but excellent policies that do not suffer from the scalability issues encountered with MDPs. For an excellent discussion of various ADP techniques, we refer to [88]. ADP was successfully used in many application areas. Some important examples include fleet management [97], dynamic container allocation [62], spare parts and supply chain management [96, 27], capacity allocation in service industry [94] and healthcare [3]. The ADP approach was also applied to the problem of ambulance dispatching and relocation. In [71], ADP was used for ambulance redeployment upon completion of service, assuming a fixed closest-first dispatching policy and no other relocations. In [93], the authors used ADP to optimize both dispatching and redeployment of ambulances that finish their jobs. More recently, in [79] the authors formulated a general model where the new incidents can be put in a queue, and there is a possibility to relocate idle ambulances.

The contribution of this chapter is threefold:

1. We develop a new relocation and dispatching policy based on ADP for the model studied in Chapter 4, with a number of basis functions introduced for the ADP approach.

2. We introduce two algorithms for fine-tuning ADP in our setting. The two algorithms, genetic algorithm and tabu search, tune the coefficients of the basis functions based on the actual performance of the corresponding policy, rather than on making a close approximation of the value function.

3. We conduct extensive computational experiments, where the ADP policies obtained

with both tuning algorithms are compared against two benchmark policies for various types of systems of realistic size. The benchmark policies are the heuristic algorithm used in Section 4.6.4 and the closest-first dispatching policy commonly used in practice. We show that the ADP policies outperform both benchmarks on various types of networks. By tuning the ADP cost function appropriately it is possible to significantly reduce the fraction of late arrivals while maintaining the same level of average response time observed under the benchmark policies.

The remainder of the chapter is organized as follows. The benchmark heuristic is described in Section 5.2. Section 5.4 introduces the ADP approach together with the tuning algorithms. Numerical experiments are presented in Section 5.5. Finally, Section 5.6 contains concluding remarks and discussion.

5.2 Benchmark Heuristic

In this section we briefly describe the combined dispatching and relocation heuristic policy used in Section 4.6.4, that consists of the best combination of dispatching and relocation heuristics studied in Chapter 4. Below we describe the two parts the heuristic consists of. One is responsible for dispatching service engineers to broken machines, and the other part for relocation of idle service engineers between stations as well as allocation to base stations of service engineers that just became idle.

Dispatching. The dispatching decision is made based on the notion of response time. For an idle service engineer m response time $rt(k,m)$ to machine k is the distance in time units from the current destination l_m to the machine location l_k plus the remaining distance to the current destination d_m. That is, $rt(k,m) = d_{l_m l_k} + d_m$. For a busy service engineer m the response time is a random variable $RT(k,m) = d_{l_m l_k} + d_m + T^{rep}$, where T^{rep} is an exponentially distributed repair time. The estimation is done using an α^{th} percentile of the repair time distribution. Throughout the computational study, we take $\alpha = 80\%$. Thus, for a busy service engineer m the response time to machine k is estimated as $rt(k,m) = d_{l_m l_k} + d_m + T^{rep}_{80\%}$. This ensures with probability 0.8 that the real response time is smaller than the estimation.

The dispatching policy works as follows. If the service engineer $m = \arg\min_n rt(k,n)$, that minimizes response time, is idle, then that service engineer is dispatched immediately. If the service engineer m is busy, then the call is placed in the queue. When a busy service engineer finishes a repair, he/she is dispatched to the closest machine from the queue, unless the queue is empty.

Relocation. The relocation of service engineers is done using the DMEXCLP heuristic introduced in [47] and adjusted for our model. This heuristic is used either when an idle service engineer is dispatched to a new breakdown, or when a service engineer finishes a repair and the queue is empty. In the first case, the heuristic considers relocating one of the rest idle stationary service engineers to another base station. In the second case, the heuristic allocates the service engineer that just finished the job to one of the base stations.

The DMEXCLP algorithm uses the notion of the *expected covered demand*. To estimate the expected covered demand, we use the procedure introduced by Larson [63, 64], and apply it to the model described in Section 4.3. Given the locations of the service engineers, the expected covered demand estimates the fraction of new requests that will be answered in time, and is approximated by

$$\frac{1}{|\mathcal{W}(s)|} \sum_{k \in \mathcal{W}(\int)} \sum_{i \in \mathcal{M}} P_i z_{ki}(s), \tag{5.1}$$

where $\mathcal{W}(s)$ is the set of working machines in state s, $z_{ki}(s)$ is a binary variable indicating if the i^{th} closest service engineer covers machine k in state s, and P_i is the probability that the first $i-1$ closest engineers are busy and the i^{th} closest service engineer is idle. Note that machine k is considered covered if there is at least one service engineer m, such that $rt(k,m) \leq t^*$. The exact procedure for estimating P_i is described in Section 4.8.

According to the DMEXCLP algorithm, the action is selected that maximizes the expected covered demand (5.1), given that it satisfies certain constraints. In the case when a service engineer becomes idle, that service engineer is allocated sequentially to each base station that is reachable within a certain time threshold T_1, then the resulting expected covered demand is computed using equation (5.1) for the obtained configuration, and the base station that leads to the best result is chosen. If there are no base stations within the given time threshold T_1, then all base stations are considered. In the case when a service engineer is dispatched to a new breakdown, we consider all pairs of base stations (r_1, r_2) that are within T_2 time units from each other, and with at least one service engineer at the base station r_1. The improvement in the expected covered demand is computed upon relocation of a service engineer from station r_1 to station r_2. If this improvement is larger than a given parameter Δ, the relocation is made. If the gain in the expected covered demand is smaller than Δ, or if there are no pairs of base stations within T_2 traveling time from each other, no relocation is made. In the computational study in Section 5.5.2 we tune the parameters T_1, T_2 and Δ separately for each system using grid search.

5.3 Model Cost Structure

In Section 4.3 we described the problem as an MDP. In this section we complete the MDP formulation with the cost structure of the problem.

Our main objective is to maximize the fraction of failures responded to within the time limit t^*. We define the cost structure accordingly. If a machine breaks down, and a service engineer does not reach this machine within the time limit t^*, then a penalty 1 is incurred. Apart from that, a small penalty $0 < \epsilon \ll 1$ is paid per time unit of service delay over t^*. The penalty ϵ is introduced only to prevent broken machines being ignored once they have experienced a service delay greater than t^*. So ϵ should not be too small to avoid unreasonably large waiting times.

Let $c\left(s_n, a_n, s_{n+1}\right)$ denote the costs incurred during the transition period from state s_n to state s_{n+1} when action a_n is taken. The transition costs are computed as follows. A unit cost is incurred for each broken machine whose waiting time exceeds the time limit t^* during the transition. An additional penalty ϵ is incurred per unit of waiting time over t^* for each broken machine. Then in total

$$c(s_n, a_n, s_{n+1}) = \sum_{k=1,\ldots,K} \mathbb{I}\{\kappa_k(s_{n+1}) \geq t^*\}\mathbb{I}\{\kappa_k(s_n) < t^*\} + \epsilon D, \tag{5.2}$$

where D is the total time the machines where waiting for service in the period $(t_n, t_{n+1}]$ that is over the time limit t^* and is equal to

$$\begin{aligned} D = &\sum_{k=1,\ldots,K} \mathbb{I}\left\{\kappa_k(s_{n+1}) > t^*\right\} \min\left(t(s_{n+1}) - t(s_n), \kappa_k(s_{n+1}) - t^*\right) \\ &+ \sum_{k=1,\ldots,K} \mathbb{I}\left\{\kappa_k(s_n) \geq 0\right\} \mathbb{I}\left\{\kappa_k(s_{n+1}) = -1\right\} \\ &\qquad \times \min\left(t(s_{n+1}) - t(s_n), \kappa_k(s_n) + t(s_{n+1}) - t(s_n) - t^*\right). \end{aligned}$$

The first component of D is the total waiting time over t^* during the transition period for all machines k such that $\kappa_k(s_{n+1}) > t^*$. The second component accounts for the event of type $e_t(s_{n+1}) = 4$ (a service engineer arrives at a machine), and adds the waiting time over t^* of the corresponding machine.

5.4 Approximate Dynamic Programming

With the ADP approach, the goal is to approximate the value function as a linear combination of several so-called *basis functions*, and choose actions using by substituting this

approximation into the optimality equations. The idea is that the resulting policy will be close to optimal if the value function approximation is.

Let us first formulate the optimality equations. Consider the process $\{s_n,\ n = 0, 1, \dots\}$ introduced in Section 4.3 with a discount factor $0 < \gamma < 1$. Let $V_\pi(s)$ denote the expected total discounted costs under policy π when starting in state s:

$$V_\pi(s) = \mathbb{E}\left[\sum_{n=0}^{\infty} \gamma^{t(s_n)} c(s_n, \pi(s_n), s_{n+1}) \mid s_0 = s\right].$$

If policy π^* is the optimal policy, then $V_{\pi^*}(s)$ satisfies the Bellman optimality equation

$$V_{\pi^*}(s) = \min_{a \in \mathcal{A}(s)} \left\{\mathbb{E}_a\left[c(s, a, s') + \gamma^{t(s')-t(s)} V_{\pi^*}(s')\right]\right\}, \quad \forall s \in S,$$

where $s' = \Phi\left(s, a, \omega(s, a)\right)$ is the next state of the process when action a is taken. We denote

$$\pi^*(s) \in \arg\min_{a \in \mathcal{A}(s)} \left\{\mathbb{E}_a\left[c(s, a, s') + \gamma^{t(s')-t(s)} V_{\pi^*}(s')\right]\right\}, \quad \forall s \in S.$$

Note that there might be multiple minimizing actions in the above expression, leading to multiple optimal policies. In that case, π^* refers to an arbitrary optimal policy. In the remainder of this chapter, we denote $V(s) := V_{\pi^*}(s)$ for ease of presentation.

To overcome the problem of a large (infinite in our case) state space, the ADP approach suggests to use an approximation $\hat{V}(s)$ of $V(s)$ that can be computed explicitly for any state s. We approximate V by $\hat{V}$, a linear combination of several basis functions $\varphi_i(\cdot),\ i = 1, \dots, I$, i.e.,

$$\hat{V}(\boldsymbol{\alpha}, s) = \alpha_0 + \sum_{i=1}^{I} \alpha_i \varphi_i(s),$$

where $\boldsymbol{\alpha} = (\alpha_0, \dots, \alpha_I)$ is a vector of coefficients, and the approximate optimal policy is defined by

$$\hat{\pi}(\alpha, s) = \arg\min_{a \in \mathcal{A}(s)} \left\{\mathbb{E}_a\left[c(s, a, s') + \gamma^{t(s')-t(s)} \hat{V}(\alpha, s')\right]\right\}, \quad \forall s \in S. \tag{5.3}$$

As the state space of the process $\{s_n,\ n = 0, 1, \dots\}$ is countably infinite, the optimal action can not be computed in advance for each state. But if for each action $a \in \mathcal{A}(s)$ we could compute $\mathbb{E}_a[c(s, a, s') + \gamma^{t(s')-t(s)} V(s')]$ offline, then being in state s we can find the optimal action a, as the action space is finite. In practice, the expected costs $\mathbb{E}_a[c(s, a, s') + \gamma^{t(s')-t(s)} V(s')]$ can be still hard to compute, even though the distribution

of s' given s and a is known (see Section 4.3). This is due to the fact that the state space is infinite. We estimate the value of $\mathbb{E}_a(\cdot)$ using Monte Carlo simulation, where the next state s' is sampled G times. Then for each realization $s'_g,\ g = 1, ..., G$ we compute the future costs $c(s, a, s'_g) + \gamma^{t(s'_g)-t(s)} V(s'_g)$ and use the average of these costs as an approximation of the expected future costs. In our computational experiments we use $G = 30$.

The choice of basis functions is very important for the performance of the approach. First, they should be straightforward to compute for any state s. Second, they should jointly capture characteristics of the optimal value function, in order to obtain an accurate approximation. We discuss our choice of basis functions in Section 5.4.1. Given a set of basis functions, the approximation is finalized by tuning the vector of coefficients $\boldsymbol{\alpha}$. We propose two metaheuristics to do this: a genetic algorithm and tabu search. We discuss these approaches in Section 5.4.2.

5.4.1 Basis Functions

In this section we discuss the basis functions we consider for our model. Some of them capture the ability of the system to respond to future breakdowns (e.g., the number of uncovered machines and expected covered demand), while others capture future penalties for decisions made in the past (e.g., the number of unassigned calls and the number of unreachable calls).

Number of unreachable machines. Consider a machine for which a service engineer was already dispatched but he/she is still on the way and is not going to reach that machine in time. If its downtime did not yet exceed the time limit t^*, we did not yet incur a penalty for this breakdown, but will in the future. The first basis function $\varphi_1(\cdot)$ represents the number of such machines:

$$\varphi_1(s) = \left| \left\{ k \in \mathcal{K} \mid 0 < \kappa_k(s) < t^* \right\} \cap \left\{ l_m(s),\ m \in \mathcal{M} \right\} \right|.$$

Number of unassigned requests. Each unassigned repair request in the current state may result in future costs. First, this request may be responded to late leading to a penalty. Second, when a service engineer is going to be dispatched to the machine, the remaining coverage will decrease, which may in turn lead to costs for future simultaneous breakdowns. The second basis function counts the number of unassigned repair requests in state s:

$$\varphi_2(s) = \left| \left\{ k \in \mathcal{K} \setminus \{ l_m(s),\ m \in \mathcal{M} \} \mid (\kappa_k(s) > 0)\ \vee\ (e_t(s) = 1) \wedge (e_l(s) = l_k) \right\} \right|.$$

Number of missed unassigned requests. The missed requests are those with waiting time larger than the time limit t^*. The missed unassigned requests are already incurring costs. Such requests also still require dispatching of a service engineer, which results in a decrease in coverage. At the same time as part of the costs is already incurred, such requests cannot be considered equal to other unassigned requests. The number of unassigned requests in state s that already passed the time limit is defined as follows:

$$\varphi_3(s) = \left| \left\{ k \in \mathcal{K} \setminus \{l_m(s),\ m \in \mathcal{M}\} \mid \kappa_k(s) \geq t^* \right\} \right|.$$

Number of uncovered machines. A machine is considered *covered* if there is at least one service engineer with the estimated response time less then t^* time units. For each pair of machine k and service engineer m the response time $rt(k, m)$ is computed according to the procedure described in Section 5.2. If a machine is not covered in state s and a failure occurs, then this will result in costs, so the number of uncovered machines is an important metric. We consider only working machines as only those can break down:

$$\varphi_4(s) = \left| \left\{ k \in \mathcal{K} \mid (\kappa_k = 0) \wedge (l_k \neq e_l(s)) \wedge (rt(k, m) > t^*\ \forall m \in \mathcal{M}) \right\} \right|.$$

Expected covered demand. In Section 5.2, we describe the relocation policy based on maximizing the expected covered demand that proved to perform well in Chapter 4. Therefore, we include the expected covered demand in the set of basis functions:

$$\varphi_5(s) = \frac{1}{|W(s)|} \sum_{k \in \mathcal{W}(s)} \sum_{i \in \mathcal{M}} P_i z_{ki}(s).$$

Average response time. The last function is the average response time to the working machines. First, the response time $rt(k, m)$ is estimated for each pair of service engineer m and machine k. Then for each demand node we choose the smallest response time over all service engineers and calculate the average over all machines. So, if $\mathcal{W}(s)$ is the set of working machines in state s, then

$$\varphi_6(s) = \frac{1}{|W(s)|} \sum_{k \in \mathcal{W}(s)} \min_{m \in \mathcal{M}} rt(k, m).$$

Future basis functions. All basis functions described above characterize the *current* state of the system. However, when we make a dispatching or relocation decision, we are not interested only in the state right after the decision is made, but also in the state upon arrival of the relocated repairman. Following [79], for each basis function $\varphi_i(s)$ we introduce two additional basis functions $\varphi_i^{(1)}(s)$ and $\varphi_i^{(2)}(s)$ which characterize the

state of the system after the arrival of the *closest* and the *second-closest* (respectively) traveling service engineer to their destinations.

Let $s^{(1)}$ denote the state of the system after the arrival of the closest traveling service engineer to his/her destination, and $s^{(2)}$ the state of the system after the arrival of the first two closest traveling service engineers to their destinations, assuming that the system is initially in state s and no other events occur. The future basis functions are computed as $\varphi_i^{(1)}(s) = \varphi_i(s^{(1)})$ and $\varphi_i^{(2)}(s) = \varphi_i(s^{(2)})$. Note that if there are no traveling repairmen, then $\varphi_i^{(1)}(s) = \varphi_i^{(2)}(s) = \varphi_i(s)$.

5.4.2 Tuning the ADP Policy

Standard methods for tuning the ADP policy aim at fitting the coefficients $\boldsymbol{\alpha}$ so that $\hat{V}(\boldsymbol{\alpha}, s) \approx V_{\pi^*}(s)$ for each state s, in hope that the obtained policy shows close to optimal performance. In this case, the coefficients $\boldsymbol{\alpha}$ in (5.3) are chosen such that the distance between the optimal value function $\boldsymbol{V}_{\pi^*}$ and $\hat{\boldsymbol{V}}_{\boldsymbol{\alpha}}$, the total discounted costs under the ADP policy with coefficients $\boldsymbol{\alpha}$, is minimized:

$$\min_{\boldsymbol{\alpha}} ||\boldsymbol{V}_{\pi^*} - \hat{\boldsymbol{V}}_{\boldsymbol{\alpha}}||_p. \tag{5.4}$$

One of such algorithms is called *approximate policy iteration*, where the current policy value function is evaluated in a simulation, and the coefficients are iteratively updated using linear regression, that is, the value $p = 2$ is used in (5.4). As noted in [70], despite some of the advantages of approximate policy iteration (e.g., ease of understanding and implementation), it also has drawbacks that might lead to low quality solutions in terms of actual performance in terms of fraction of late arrivals. We implemented approximate policy iteration, but observed consistently poor performance despite convergence to low values of the mean squared error in the linear regression. Note that the choice of an action in a given state under the ADP policy (5.3) depends on the relative difference in value function for different states, rather than on the actual values. As our goal is to obtain a high performance policy rather than to fit the value function in (5.4), we resort to heuristics that tune the coefficients $\boldsymbol{\alpha}$ based on the corresponding performance in a simulation.

Genetic algorithm

The first approach we propose is a genetic algorithm. Genetic algorithms draw inspiration in evolution and natural selection, and are widely used for optimization problems [123]. The key concepts of genetic algorithms are:

- *Population*: each individual/solution is a part of a pool;

- *Selection*: the fittest individuals survive;
- *Crossover*: the fit individuals reproduce, propagating their fit genes;
- *Mutation*: sometimes new characteristics appear by accident.

In our case an individual is a vector of coefficients $\boldsymbol{\alpha}$, and a population is a set of vectors $\boldsymbol{\alpha}^{(n)},\ n = 1, \ldots, N^{GA}$, where N^{GA} is the size of the population and one of the parameters of the algorithm. In each iteration of the algorithm, the population is updated using *mutation*, *crossover* and *selection*. The population is initialized by adding random vectors to the same vector $\alpha^{(0)}$. In our experiments we set all elements of vector $\alpha^{(0)}$ equal to 1, except those corresponding to the expected covered demand, for which the coefficient is set to -1. To produce an individual of an initial population, we then add a random value to each element of $\alpha^{(0)}$ drawn from a uniform distribution $\mathcal{U}(-1, 1)$.

During each iteration of the algorithm, a new population is constructed as follows. First, the *crossover* operator is used, where N^{GA} pairs of candidate solutions are randomly selected from the current population and the average of each pair is chosen as a candidate for the next population. Second, the *mutation* operator is applied to each candidate solution from the current population. Given a solution $\boldsymbol{\alpha}^{(n)}$, the *mutation* operator adds a vector, the ith component of which is normally distributed with mean 0 and standard deviation $A^{GA}|\boldsymbol{\alpha}_i^{(n)}|$. Here, A^{GA} is the parameter of the algorithm, and is referred to as *mutation amplitude*. This way, together with the current population, after mutation and crossover is done, we obtain $3N$ candidate solutions for the next population.

Finally, the selection is done to produce the next population, where N^{GA} solutions are chosen out of the $3N^{GA}$ candidates. To measure fitness of a candidate solution, we run simulation with the corresponding ADP policy. Simulation starts from the same state of each candidate, and the fitness of a given candidate solution is measured as the costs per machine failure observed in a simulation (see the cost structure in Section 5.3). Note that the larger the time horizon of simulation, the better is an estimation of the solution fitness, but the larger the computational time of the algorithm. In our experiments, we set the time horizon equal to $500/(\lambda K)$.

All $3N$ solutions of the new population are evaluated by means of simulation: the system is simulated under corresponding policies from the same initial state and the fraction of calls answered in time is observed. Then only the $q^{GA}N^{GA}$ best performing and $(1 - q^{GA})N^{GA}$ randomly chosen candidate solutions survive and constitute the next population. Here, the fraction of fittest candidate solutions q^{GA} is the parameter of he algorithm. The algorithm stops after a certain number of iterations. The scheme of the

algorithm is depicted in Figure 5.1.

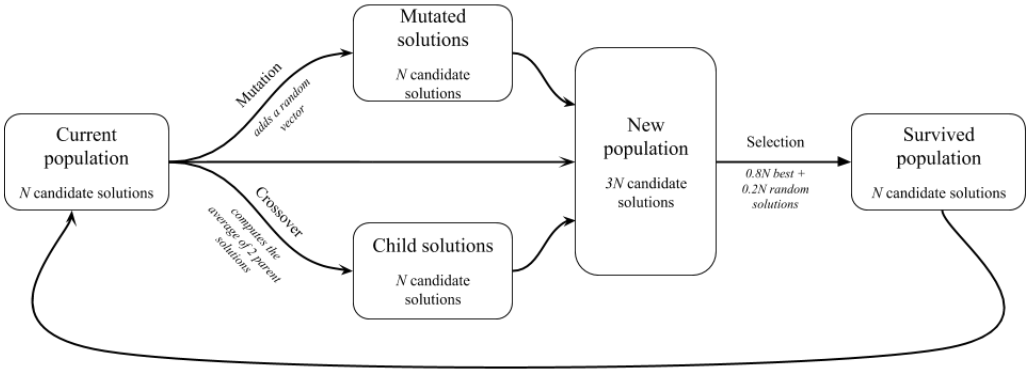


Figure 5.1: Genetic algorithm

Tabu search

Tabu search is a high-level metaheuristic technique that guides other local search heuristic methods and is constructed in such a way that allows to escape local optima [32]. The main idea is to allow moves to worse solutions and prevent cycling back to the local optima with the help of the so-called *tabu lists*. Here, we introduce the main concepts of the tabu search metaheuristic:

- *Incumbent solution*: the current solution representing the current state of the algorithm. Tabu search performs a walk in the search space through a sequence of incumbent solutions. The best found solution is the outcome of the algorithm.
- *Move*: a procedure of obtaining a new feasible solution from the incumbent solution.
- *Neighborhood*: determined by the move and represents all the feasible solutions that can be reached by moving from the incumbent one.
- *Tabu list*: a list of restrictions that impose limitations based on certain attributes of the recently performed moves. The tabu list has a limited size, it is based on a short term memory and prevents the search to return to the recently visited solutions.

- *Tenure:* a number of iterations of the algorithm a tabu restriction stays in the tabu list.

The algorithm has two phases: (1) the *diversification phase* and (2) the *intensification phase*. First, it performs a certain number of iterations in the diversification phase, followed by a certain number of iterations in the intensification phase that starts from the best found solution of the diversification phase. In the diversification phase, the focus is on exploring the neighborhood and escaping the local optima. In each iteration, the algorithm moves to the best found candidate solution from the neighborhood, even if it is worse than the current incumbent solution. In the intensification phase, the goal is to find the local optimum around the best found solution of the diversification phase. The movement to a new incumbent solution is allowed only if it is better than the current one.

The *move* operator works as follows. Given an incumbent solution $\boldsymbol{\alpha}$, the *move* operator adds a random vector to it, ith component of which is normally distributed with mean 0 and standard deviation $A^{TS}|\alpha_i|$, unless i is in the tabu list. The tabu list contains the indices of the coefficients in the incumbent solution that are not allowed to change, as well as the remaining number of iterations those indices will stay in the tabu list. The size of the tabu list TLS is one of the parameters of the algorithm. The tabu list is initialized as an empty set. In the end of each iteration, the remaining number of iterations is decreased by one. If it becomes zero, a new set of TLS indices is chosen as tabu with the remaining number of iterations equal to tenure TLT, another parameter of the algorithm. In the diversification phase, the tabu indices are chosen as non-tabu indices that changed the most in the last iteration, and in the intensification phase those that changed the least.

We initialize the algorithm with the incumbent solution $\alpha^{(0)}$ defined the same way as in the genetic algorithm. At every iteration N^{TS} candidates are obtained from the incumbent solution using the *move* operator. The candidate solutions are estimated first by the surrogate fitness function, and then a subset of $f^{TS}N^{TS}$ good candidate solutions is estimated by the primary fitness function. Both fitness functions are measured as costs per machine failure observed in a simulation. The only difference is that the surrogate fitness is computed over a shorter time horizon to quickly identify potentially good candidates. Those are further evaluated in a larger simulation. Using surrogate fitness allows us to better explore the neighborhood at a lower computational cost. In our experiments, we use the time horizon of $30/(\lambda K)$ time unites for the surrogate fitness, and $500/(\lambda K)$ for the primary fitness. If more than $f^{TS}N^{TS}$ candidates result in zero surrogate fitness (this may happen when the quality of the incumbent solution improves), the corresponding time horizon is doubled.

At the end of each iteration, the algorithm updates the incumbent solution and the

tabu list. If in the intensification phase the incumbent solution does not change, then the *move* amplitude is decreased by 10% before the next iteration. The algorithm stops after a certain number of iterations in the diversification and intensification phases. If necessary, the cycle can repeat starting from the incumbent solution. In the numerical experiments in Section 5.5 we perform only one such cycle.

5.5 Numerical Experiments

In this section, we present the results of our numerical experiments. The setup of the experiments is as described in Section 4.6.1. In Section 5.5.1, we study the parameters of the two ADP tuning algorithms presented in Section 5.4.2. In Section 5.5.2, we use simulation to compare the ADP policies obtained by both tuning algorithms against the heuristic described in Section 5.2 and the closest-first policy.

5.5.1 Parameters of the ADP Tuning Algorithms

In this section, we discuss the important parameters of the two ADP tuning algorithms, genetic algorithm and tabu search, introduced in Section 5.4.2. There are two criteria that affect the choice of the values for each of the parameters. The first one is the convergence rate, that is, how quickly in terms of the number of iterations the algorithm finds good quality solutions. The second criterion is the computational time, that is, how much time the algorithm spends per iteration. The right balance should be found so that the good solution are found within a reasonable amount of time. In all experiments in the rest of this section we use the randomly generated network shown in Figure 5.2 with $K = 8$, $R = 3$, $M = 5$, $t^* = 10$, $\lambda = 0.01$ and $\mu = 0.1$. The fitness of each solution is measured in terms of cost per incident incurred in a simulation.

Number of basis functions

First, we determine if there are basis functions that do not contribute to the quality of the solutions found. The more basis functions are included, the more complex the problem becomes, as each extra basis function adds an extra dimension to the search space. This leads to large computational times. In our experiments, we discovered that when an ADP tuning algorithm is ran with the six main basis functions $\varphi_1(\cdot), \ldots, \varphi_6(\cdot)$, the coefficients corresponding to the functions $\varphi_4(\cdot)$ and $\varphi_6(\cdot)$ are driven to zero, meaning that these two basis functions have insignificant effect on the quality of the ADP policy. Since the number of uncovered machines and the average response time seem to contain redundant information about the state of the system, we consider omitting the corresponding main

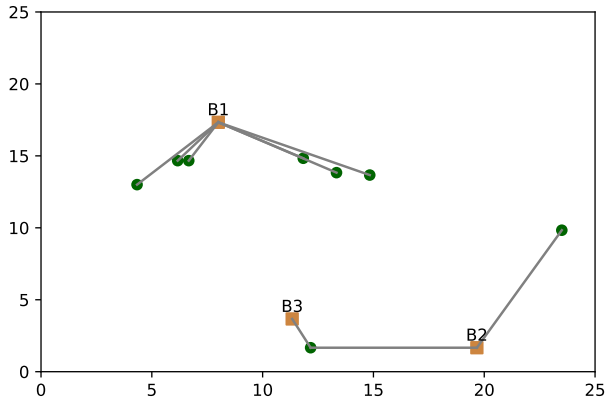


Figure 5.2: Network used to test the parameters of the ADP tuning algorithms

and future basis functions. Figure 5.3 demonstrates the convergence of the genetic algorithm depending on the number of basis functions used. Here, '6' corresponds to the six main basis functions, '4' to the main basis functions excluding $\varphi_4(\cdot)$ and $\varphi_6(\cdot)$, '8' to the previous four plus the corresponding one-step-ahead future basis functions, '12' to the previous eight plus the corresponding two-steps-ahead future basis functions. The fitness of the best solution in a population is plotted against the number of iteration.

Note that the initial populations are different for different sets of basis functions, and the solutions are sampled at random. So, the quality of the initial population does not depend on the set of basis functions used. It can be seen from Figure 5.3 that using six main basis functions instead of four does not contribute to the quality of the obtained policies. Adding the one-step-ahead future basis functions, however, boosts the performance of the policies. Including the two-steps-ahead future basis functions further improves the quality of the best found policy, although the effect is marginal. A similar effect is observed when running the tabu search algorithm. In Section 5.5.2 we use twelve basis functions with both algorithms when tuning the ADP policy.

Parameters of genetic algorithm N^{GA}, A^{GA}, q^{GA}

Figure 5.4 presents the effects of the genetic algorithm parameters on the fitness convergence, where the best population fitness is plotted per iteration. It can be seen that

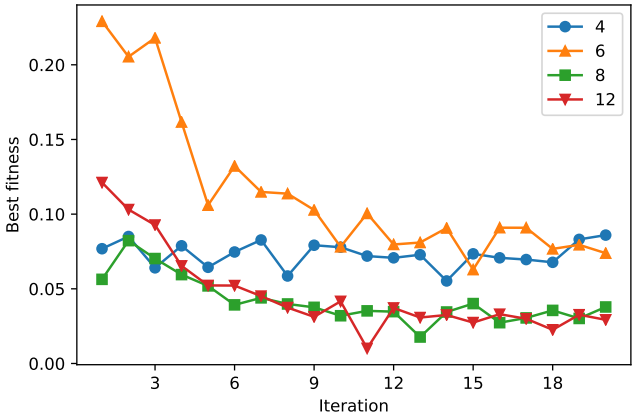
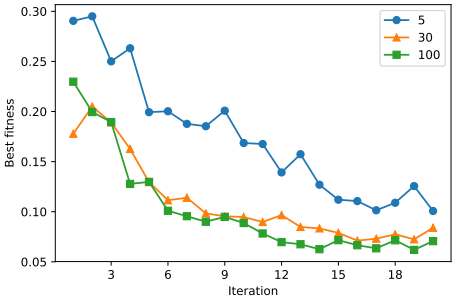


Figure 5.3: Genetic algorithm convergence for different sets of basis functions. $N^{GA} = 50$, $A^{GA} = 1$, $q^{GA} = 0.8$

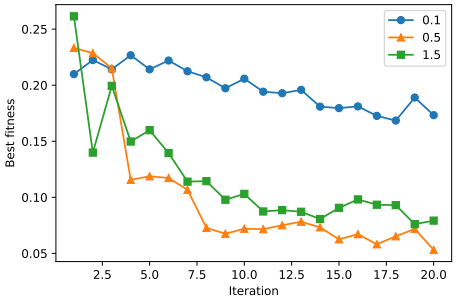
increasing the population size N^{GA} has a positive effect. It, however, directly affects the computation time, as in each iteration a simulation is run for every solution in the population. Hence, the choice should be made depending on the computational resources to ensure that the policy can be obtained within a reasonable amount of time. In Section 5.5.2, we use N^{GA} between 50 and 100 depending on the system. The mutation amplitude A^{GA} and the fraction of fittest q^{GA} do not affect computation time, but do influence the convergence of the algorithm. Both parameters should not be too high or too low. When tuning the ADP policy with genetic algorithm in Section 5.5.2, we set $A^{GA} = 0.7$ and $q^{GA} = 0.8$.

Parameters of tabu search f^{TS}, TLS, TLT

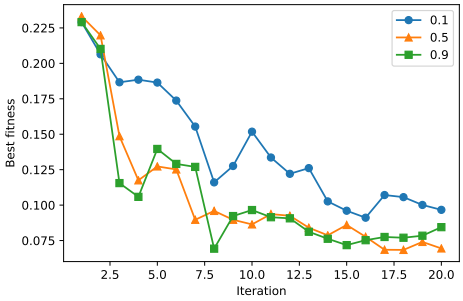
The parameters N^{TS} and A^{TS} of the tabu search algorithm demonstrate similar influence as the N^{GA} and A^{GA} parameters of the genetic algorithm. Therefore, we do not discuss them here. In our computational experiments in Section 5.5.2, we use the values between 200 and 400 for N^{TS}, and we set $A^{TS} = 0.7$. The first tabu search specific parameter we consider is f^{TS}. It is intuitive that the larger its value, the better the convergence in fitness, as for larger values of f^{TS} a bigger part of the neighborhood is explored in each iteration. It is also intuitive that this leads to larger computation time, as in each iteration more solutions from the neighborhood are evaluated with the primary fitness



(a) $\boldsymbol{N^{GA}}$ ($A^{GA} = 1$, $q^{GA} = 0.8$)



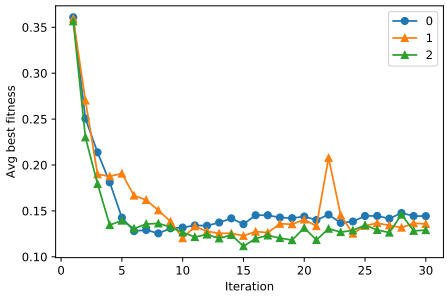
(b) $\boldsymbol{A^{GA}}$ ($N^{GA} = 50$, $q^{GA} = 0.8$)



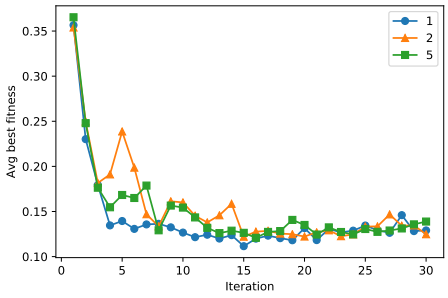
(c) $\boldsymbol{q^{GA}}$ ($N^{GA} = 50$, $A^{GA} = 1$)

Figure 5.4: Genetic algorithm convergence with six main basis functions

function. When fitting ADP policies in Section 5.5.2, we set $f^{TS} = 0.3$. Figure 5.5 shows how the parameters TLS and TLT affect the algorithm convergence, where the average primary fitness of the best candidates in the neighborhood is plotted per iteration of a diversification phase. These parameters have no effect on computation time, but do influence the convergence of the algorithm and should be chosen carefully. In the example shown in Figure 5.5 increasing TLS helped to find a better neighborhood, but at a cost of making more iterations. Increasing TLT, however, did not help converging to a good neighborhood. Note that the right choice of these parameters depends on the number of basis functions used. In Section 5.5.2, we set $TLS = 3$ and $TLT = 2$ in combination with twelve basis functions.



(a) ***TLS***
($N^{TS} = 400$, $A^{TS} = 1$, $f^{TS} = 0.3$, $TLT = 1$)



(b) ***TLT***
($N^{TS} = 400$, $A^{TS} = 1$, $f^{TS} = 0.3$, $TLS = 2$)

Figure 5.5: Tabu search convergence with six main basis functions

5.5.2 ADP Performance

In this section, we compare the ADP policy against the heuristic policy (Heuristic) described in Section 5.2 and the closest-first (CF) policy that always dispatches the closest engineer and does not perform relocations. We consider various types of systems by changing the parameters μ, d and t^*. We fix λ at 0.01 and the size of the networks with $K = 40$, $R = 7$ and $M = 10$, which is realistic for many real life maintenance networks. For each combination of the parameters 10 random networks are generated as described in Section 4.6.1. For each of the networks the ADP policy is tuned with both genetic algorithm (GA) and tabu search (TS). The two ADP policies are then compared against the Heuristic and the CF policies using simulation over the time horizon of $1000/\lambda = 100000$ time units. The performance of the policies is measured with the cost per incident (computed according to (5.2) with $\epsilon = 0.01$ and referred to as Cost), fraction of late arrivals (FLAR), and average response time (ART). Both ADP tuning algorithms were parallelized and ran on the 16-core nodes of a cluster computer.

Table 5.1 presents the obtained results. Note that our main objective was to minimize FLAR. Therefore, we chose the value of ϵ such that there is no notable decrease in ART compared to the other two policies. The ADP policy obtained with both tuning algorithms significantly outperforms both CF and Heuristic in terms of FLAR for all considered types of the systems. There is almost no increase in ART, and in some cases ART is also decreased by a large margin. The ADP policy performs especially good for the systems with larger distances and higher load, those where relocations contribute most to reducing response time to future incidents.

Table 5.1: ADP performance on various types of systems, $\epsilon = 0.01$

μ	d	t^*	Metric	CF	Heuristic	ADP GA	ADP TS
0.1	**0.3**	**20**	Cost	1.68	1.26	1.35	1.40
			FLAR	97.0%	79.4%	74.4%	73.8%
			ART	90.7	65.1	78.5	84.6
		5	Cost	0.49	0.39	0.38	0.35
			FLAR	46.1%	36.7%	34.3%	32.7%
			ART	6.9	5.7	5.8	6.1
	2	**20**	Cost	1.07	0.85	0.56	0.63
			FLAR	87.3%	60.2%	40.7%	46.3%
			ART	39.4	42.4	32.2	33.2
		5	Cost	0.15	0.16	0.14	0.15
			FLAR	15.0%	15.7%	13.1%	14.3%
			ART	3.5	3.3	4.5	4.6
0.5	**0.3**	**20**	Cost	1.46	1.58	0.85	0.96
			FLAR	95.6%	80.8%	49.2%	58.6%
			ART	69.7	96.2	52.1	54.6
		5	Cost	0.11	0.13	0.09	0.10
			FLAR	10.4%	10.1%	9.1%	9.3%
			ART	3.5	5.4	3.2	3.7
	2	**20**	Cost	0.80	0.88	0.30	0.31
			FLAR	68.2%	70.9%	23.3%	24.2%
			ART	29.2	27.7	21.9	22.2
		5	Cost	0.02	0.02	0.02	0.02
			FLAR	2.0%	1.9%	1.7%	1.8%
			ART	2.7	3.0	3.4	3.2

Table 5.2: ADP performance with $\epsilon = 0.001$

μ	d	t^*	Metric	CF	Heuristic	ADP GA
0.1	**0.3**	**20**	Cost	1.04	0.84	0.49
			FLAR	97.0%	80.1%	28.5%
			ART	90.7	61.1	225.1
		5	Cost	0.46	0.34	0.38
			FLAR	46.1%	34.3%	33.8%
			ART	6.9	5.3	44.8
	2	**20**	Cost	0.89	0.63	0.20
			FLAR	87.3%	59.9%	9.3%
			ART	39.4	44.2	124.1
		5	Cost	0.15	0.15	0.12
			FLAR	15.0%	15.2%	12.2%
			ART	3.5	3.3	4.4

As already mentioned before, the ADP cost structure (5.2) allows flexibility in terms of prioritizing FLAR against ART. If, for example, it is not important how large the waiting time of a broken machine is, given that it is over t^* time units, the ϵ parameter can be reduced to reflect that. Table 5.2 shows the performance of the ADP policy for a subset of systems used in Table 5.1, with $\epsilon = 0.001$.

For all the considered systems, both the genetic algorithm and the tabu search were able to find good solutions within a couple of days, and in some cases within a few hours. Note that the parameters N^{GA} and N^{TS} of the ADP tuning algorithms that affect both solution quality and computation time, as well as the number of performed iterations, were chosen such that the good policies are obtained for all systems within a reasonable amount of time. For any given system, the ADP policy can be further improved by increasing N^{GA} and N^{TS} and/or the number of iterations. The choice between the genetic algorithm and the tabu search depends on the computational resources available. In our experiments, when parallelized and ran on the 16-core nodes of a cluster computer, the genetic algorithm was able to find good quality solution about twice faster than the parallelized tabu search. However, when run sequentially, the tabu search was a few times faster.

5.6 Conclusion

In this chapter, we studied the problem of dynamic dispatching and relocation of service engineers. We considered the model introduced in Chapter 4 and developed an

ADP approach to the problem. To that end, we proposed a number of basis functions and demonstrated in the computational experiments which of them are most important for finding a high performance policy. We also introduced two algorithms for tuning the coefficients of the basis functions in the ADP approach. We conducted extensive computational experiments, where we studied the parameters of the proposed ADP tuning algorithms, and compared the ADP approach against the two benchmark policies, closest-first policy and a heuristic algorithm that proved to perform well in Chapter 4.

We showed that the ADP based policy outperforms both benchmarks for various types of systems, with most significant improvements for those with larger distances and higher loads. For the types of systems used in the study, we observed significant improvement over the best benchmark in terms of FLAR, without loss in ART. As it is computed offline and only once for each given type of system, the ADP policy is computationally tractable for real-life applications. Additionally, by modifying the cost structure with a single parameter ϵ, it is possible to strike the desirable balance between the fraction of late arrivals and the average response time.

6

Integrating Condition-Based Maintenance into Dynamic Spare Parts Management

In this chapter we introduce a new model where the concept of condition-based maintenance is combined in a network setting with dynamic spare parts management. The model facilitates both preventive and corrective maintenance of geographically distributed capital goods as well as relocation of spare parts between different warehouses based on the availability of stock and the condition of all capital good installations. We formulate the problem as a Markov decision process, with the degradation process explicitly incorporated into the model. Numerical experiments show that that significant cost savings can be achieved when condition monitoring is used for preventive maintenance in a service network for capital goods.

The work in this chapter is based on [105]: D. Usanov, P.M. van de Ven, and R.D. van der Mei. Integrating Condition-Based Maintenance into Dynamic Spare Parts Management (2019). Manuscript submitted for publication.

6.1 Introduction

Capital goods such as MRI scanners, lithography machines, aircraft, or wind turbines are subject to deterioration and require maintenance over their lifetime. Continuous operation of such assets is crucial, as failures can have significant negative effects. Advancements in condition monitoring techniques facilitate tracking the degradation process of capital goods in real time. This creates a tremendous potential for implementing preventive maintenance policies that use sensor data to indicate which spare parts should be replaced before a breakdown happens. This type of preventive maintenance is called *condition-based maintenance* (CBM), and it can be extremely useful to mitigate the risks related to downtime of capital goods.

The existing research on CBM is focused on optimizing control limits and/or maintenance intervals for one single- or multi-component machine. However, these works typically ignore the fact that these machines are parts of a network comprising many machines distributed across different geographical locations, and the spare parts supply distributed over a number of stock points to ensure short response times to failures. The research literature on dynamic spare parts management in a service network is primarily focused on optimal corrective maintenance and relocation of spare parts, where the failures are typically assumed to follow a Poisson process and cannot be predicted.

In this chapter we integrate the CBM concept into a network setting with dynamic spare parts management. This allows us to relocate spare parts between stock points and perform proactive maintenance of machines based on stock levels and the condition of all machines. Figure 6.1 illustrates how incorporating CBM changes the complexity and the dynamics of the service network. The state space increases, as each of the machines has more than just two possible condition states (perfect and failure), as it is typically assumed in the research literature on dynamic spare parts management. However, this provides more information about the overall state of the network, and therefore, enables more educated decision making. For instance, instead of relocating a spare part upon failure of one of the machines, it might be better to preventively repair another machine that is close to failure. Using the information obtained through condition monitoring can improve both maintenance and relocation activities, and boost the performance of the service network.

We consider a single-component machine and assume a Markovian degradation process, where a machine moves through a sequence of intermediate states before it reaches the failure state. This is a common assumption in research literature (see for example [56, 53]). A number of such machines are spread across a service region, and we optimize corrective and preventive maintenance actions, as well as proactive relocation of spare parts between

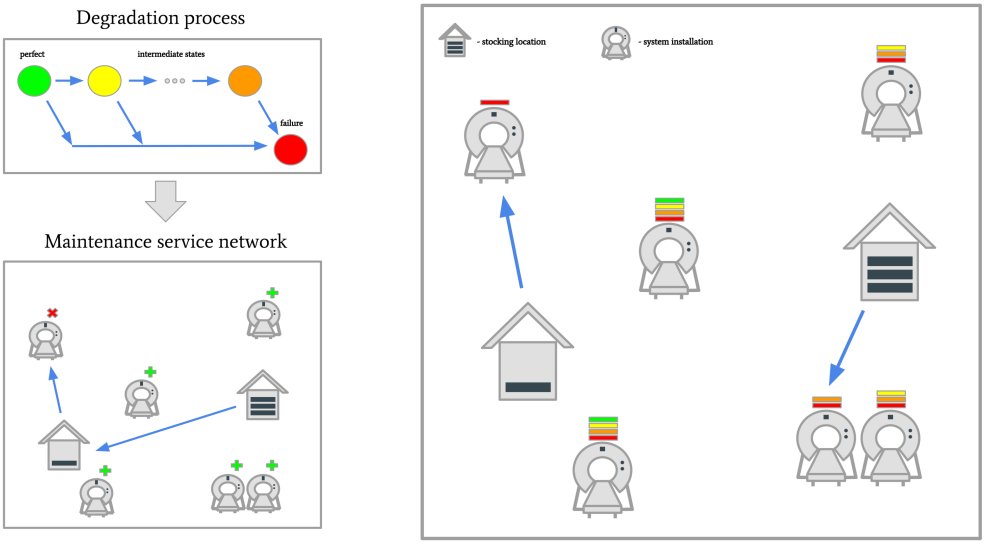


Figure 6.1: Integrating CBM into dynamic spare parts management

stock points. We show that, by introducing condition monitoring into a network setting, significant improvements can be achieved in reducing total expected costs, independent of the cost structure.

To summarize, in this chapter we make the following contributions:

1. We propose a new model, where condition-based maintenance is integrated into the dynamic spare-parts management;
2. We formulate the problem as a Markov Decision Process (MDP), and conduct numerical experiments showing that incorporating CBM can significantly reduce the maintenance costs.

The remainder of this chapter is organized as follows. In Section 6.2, we provide brief review of the relevant literature. Section 6.3, presents the model and the MDP formulation. In Section 6.4 we numerically evaluate the impact of CBM the optimal policy performance. Conclusions and suggestions for further research are made in Section 6.5.

6.2 Literature Review

The research most related to our work comes from the two streams of literature: dynamic dispatching and relocation of spare parts, and condition-based maintenance. Although attention to both topics seems to increase in recent years, to the best of our knowledge, there has been no research on combining these two concepts. Below we make a brief overview of the latest work in the two domains.

Dispatching and relocation of spare parts considers operational level decision making in maintenance service networks. Proactive and reactive allocation of stock in spare parts networks is commonly referred to in research literature as *lateral transshipment*. For a comprehensive overview of the research done on lateral transshipments we refer to [121, 83]. In [118] the authors consider an inventory model with fixed inventory level and two warehouses, both facing Poisson demand for spare parts. They provide an exact analysis of the model that derives an optimal policy for allocation of demand to warehouses. In [101] a dynamic demand allocation rule is developed that is scalable for spare parts networks of realistic size. The authors show that significant cost savings can be achieved with their approach compared to the static allocation rule commonly used in practice, where the closest warehouse is used to fulfill the demand. An interesting contribution is made in [84], where the authors consider relocating additional stock when satisfying real-time demand. Recent developments include [28, 74], where proactive relocation of stock is studied along with the reactive policies.

The recent works on condition-based maintenance include [126, 86, 125, 22]. In [126] the authors study a multi-component system where each component follows a stochastic degradation process according to the so-called random coefficient model. A joint maintenance of components whose condition falls below a control limit is performed at scheduled downs. Those control limits together with the maintenance interval are subject to optimization. In [125] the authors consider condition-based maintenance of a single component that is a part of a complex system. This component follows a stochastic degradation process for which the authors use the random coefficient model and Gamma process. A control limit policy is analyzed, with maintenance actions taken at scheduled or unscheduled downs related to other components of the system. A single component model with stochastic degradation process is also studied in [86]. The authors consider an application for a manufacturing system and look into joint optimization of control limit and production quantity of the lot-sizing policy. Another way to model the degradation process is using Markov process with discrete states. In [22] the authors consider a single-component system that follows Markovian degradation process with two intermediate states and study the optimal control limit policy of such system.

Our work is different from these streams of research literature, as we consider a generic model where *both reactive and proactive allocation of stock* is allowed in real-time, along with condition-based maintenance.

6.3 Model

We consider a network of identical single-component machines supported by a set of local warehouses and a central warehouse. The state of each machine is completely and continuously observable. Replacement of components happens either *preventively* or *correctively* upon a failure. Let $\mathcal{I} = \{0, 1, ..., I\}$ be a set of warehouses, with $i = 0$ - central warehouse with ample capacity. Let $\mathcal{J} = \{1, ..., J\}$ be a set of machines.

We assume that the lifetime of a machine is Cox distributed with $N > 0$ phases [60]. We choose Cox distribution because it allows to approximate any random variable with positive support. Denote the condition of a machine $j \in \mathcal{J}$ by $C_j \in \{0, 1, ..., N\}$, where 0 corresponds to failure and N to the perfect condition. A machine stays in each state $n \in \{1, ..., N\}$ for an exponential amount of time with parameter μ_n, then it moves either to the 'failure' state 0 with probability $\alpha_n < 1$ or to the state $n-1$ with probability $1-\alpha_n$. Note that from state 1 the machine moves to state 0 with probability 1, so $\alpha_1 = 1$. Upon breakdown, a spare part is dispatched to that machine from either a local or the central warehouse. The downtime of a machine due to corrective maintenance includes traveling time and repair time, and we assume it is exponentially distributed with parameter μ_0. A machine can be preventively repaired at any point in time. We assume that there is no downtime of a machine in this case, so a spare part is replaced instantly. This is a common assumption in literature, as preventive maintenance is typically easier than corrective, and the corresponding downtime does not include delivery of a spare part. After a spare part is replaced, either correctively or preventively, a machine moves back into the 'perfect' state N.

If a failure occurs, a spare part is to be dispatched immediately either from a local warehouse, or from the central warehouse. Once a spare part is dispatched from a local warehouse, a replenishment order is placed. The replenishment lead time is exponentially distributed with parameter γ. If a spare part was dispatched from a local warehouse i, a relocation of a single spare part from one of the other warehouses to the warehouse i is allowed. We assume that such relocations happen instantaneously. This is a reasonable assumption, as the traveling times between warehouses are typically low compared to the average time between consecutive failures of capital goods.

If the condition of a machine degrades but it is still not in a failure state, we consider

two types of actions. We may decide to repair this machine preventively, and do up to one relocation the local warehouse from which a spare part is dispatched. Alternatively, we may decide to not repair the machine, and do up to one relocation between any two local warehouses. These relocations are intended to better distribute stock across the network, and reduce future downtime.

Let $\boldsymbol{C} = (C_1, ..., C_J)$, $\boldsymbol{F} = (F_1, ..., F_I)$, $\boldsymbol{P} = (P_1, ..., P_I)$, where $F_i \geq 0$ and $P_i \geq 0$ denote the stock level and the pipeline stock (replenishment orders) at warehouse i, respectively. Denote by $K = \sum_{i \in \mathcal{I}} (F_i(t) + P_i(t))$ the aggregate inventory level. Note that K always remains constant, as each time a spare part is dispatched, a new one is ordered immediately and added to the corresponding pipeline stock. Let $\boldsymbol{X} = (\boldsymbol{F}, \boldsymbol{P}, \boldsymbol{C}, j)$ with $j \in \{0, 1 \ldots, J\}$, denote the state of the system immediately after the condition of machine $j \in \mathcal{J}$ changes or a replenishment order arrives ($j = 0$). Let also $\boldsymbol{a}(\boldsymbol{X}) = (x, y, z)$ represent the action in state $\boldsymbol{X}$. Here, $x \in \{-1, 0, 1, ..., I\}$ indicates the warehouse from which a spare part is to be dispatched, $y \in \{-1, 1, ..., I\}$ indicates the warehouse from which a spare part is to be relocated, and $z \in \{-1, 1, ..., I\}$ the warehouse to which a relocated spare part should be placed. The value $x = -1$ corresponds to the case when no dispatching is made, $y = z = -1$ to the decision not to relocate a spare part.

6.3.1 Actions

We consider two types of actions. The first type includes both dispatching a spare part and relocating another spare part to the warehouse from which the dispatching was made. Relocating a spare part is not necessary and is only considered if a part was dispatched from a local warehouse. This type of action can be made either correctively upon a failure, in which case repair is required, or preventively when a machine's condition degrades but the machine is still functioning. Let $\mathcal{W}(\boldsymbol{X}) \subseteq \mathcal{I}$ denote the set of local warehouses that have at least one spare part in stock in state $\boldsymbol{X}$. For a state $\boldsymbol{X} = (\boldsymbol{F}, \boldsymbol{P}, \boldsymbol{C}, j)$, $C_j < N$, the type-1 action space is defined as

$$\begin{aligned} \mathcal{A}_1(\boldsymbol{X}) = &\{(x, y, z) | x \in \mathcal{W}(\boldsymbol{X}),\ y \in \mathcal{W}(\boldsymbol{X}) \setminus \{x\},\ z = x\} \\ &\cup \{(x, y, z) | x \in \mathcal{W}(\boldsymbol{X}) \cup \{0\},\ y = -1,\ z = -1\}. \end{aligned} \tag{6.1}$$

The second type of action includes only relocations. These are allowed upon a change of a machine state that does not result in a failure. In this case, a single relocation is allowed between any pair of local warehouses as long as the origin warehouse has a spare part available. For a state $\boldsymbol{X} = (\boldsymbol{F}, \boldsymbol{P}, \boldsymbol{C}, j)$, $C_j > 0$, the type-2 action space is defined as

$$\mathcal{A}_2(\boldsymbol{X}) = \{(x, y, z) | x = -1,\ y \in \mathcal{W}(\boldsymbol{X}),\ z \in \mathcal{I} \setminus \{y\}\} \cup \{(-1, -1, -1)\}. \tag{6.2}$$

Thus, the total action space is $\mathcal{A}(\boldsymbol{X}) = \mathcal{A}_1(\boldsymbol{X}) \cup \mathcal{A}_2(\boldsymbol{X})$.

We denote by $\boldsymbol{X}^{\boldsymbol{a}}(t)$ the state of the system at time t under decision rule $\boldsymbol{a}$. The process $\{\boldsymbol{X}^{\boldsymbol{a}}(t)\}_{t\geq 0}$ is a continuous-time Markov process, with the following state space:

$$\mathcal{S} = \big\{(\boldsymbol{F},\boldsymbol{P},\boldsymbol{C},j)|F_i \geq 0, P_i \geq 0\ \forall i \in \mathcal{I}; \sum_{i\in\mathcal{I}}(F_i+P_i) = K; \\ C_j \in \{0,1,...,N\}\ \forall j \in \mathcal{J}; j \in \{0,1,...,J\}\big\}.$$

6.3.2 Transitions

In this subsection we define transition rates between states corresponding to decision epochs. We consider the following four types of events when the state of the system changes:

1. arrival of a replenishment order at local warehouse i;
2. failure of machine j;
3. degradation of machine j that is not a failure;
4. repair of machine j.

Given the machines' condition vector $\boldsymbol{C}$, let $\boldsymbol{C}^{j,n}$ denote a vector that is obtained from $\boldsymbol{C}$ by setting its j-th component to n, so $C_k^{j,n} = C_k\ \forall k \neq j$ and $C_j^{j,n} = n$. We denote as $\boldsymbol{e}_k$ the vector of length I with the k-th element equal to 1 and all other elements equal to 0 for $k \in \{1,...I\}$. Finally, $\boldsymbol{e}_0$ and $\boldsymbol{e}_{-1}$ both denote a zero vector of length I.

Assume that the system is in state $\boldsymbol{X} = (\boldsymbol{F},\boldsymbol{P},\boldsymbol{C},j)$ immediately after an event occurred at machine j and before an action is taken. Remember that we set $j = 0$ if the last even is an arrival of a replenishment order. We want to define the transition rate $q^{\boldsymbol{a}}\big((\boldsymbol{F},\boldsymbol{P},\boldsymbol{C},j),(\boldsymbol{F}'+,\boldsymbol{P}',\boldsymbol{C}',j')\big)$ from each state $(\boldsymbol{F},\boldsymbol{P},\boldsymbol{C},j)$ to each possible next state $(\boldsymbol{F}'+,\boldsymbol{P}',\boldsymbol{C}',j')$ that is determined by an action $\boldsymbol{a}$ and the next event j'. The following types of transitions are possible in our model.

Type 1. <u>*Last event:*</u> replenishment at warehouse $i \in \{1,..,I\}$; <u>*action:*</u> $\boldsymbol{a} = (-1,-1,-1)$. The last event is a replenishment at a local warehouse i. In that case only one action is possible, and that is to do nothing. The state of the system immediately after the action is taken is $\boldsymbol{X} = (\boldsymbol{F},\boldsymbol{P},\boldsymbol{C},0)$. The transition rates then depend on the next state, and are defined as follows.

1. <u>*Next event:*</u> replenishment at k, $P_k > 0$. The corresponding transition rate is

$$q^{\boldsymbol{a}}\big((\boldsymbol{F},\boldsymbol{P},\boldsymbol{C},0),(\boldsymbol{F}+\boldsymbol{e}_k,\boldsymbol{P}-\boldsymbol{e}_k,\boldsymbol{C},0)\big) = P_k\gamma.$$

2. *Next event:* failure of machine j, $C_j > 0$. The corresponding transition rate is

$$q^{\boldsymbol{a}}\big((\boldsymbol{F},\boldsymbol{P},\boldsymbol{C},0),(\boldsymbol{F},\boldsymbol{P},\boldsymbol{C}^{j,0},j)\big) = \alpha_{C_j}\mu_{C_j}.$$

3. *Next event:* degradation of machine j, $C_j > 1$. The corresponding transition rate is

$$q^{\boldsymbol{a}}\big((\boldsymbol{F},\boldsymbol{P},\boldsymbol{C},0),(\boldsymbol{F},\boldsymbol{P},\boldsymbol{C}^{j,C_j-1},j)\big) = (1-\alpha_{C_j})\mu_{C_j}.$$

4. *Next event:* repair of machine j, $C_j = 0$. The corresponding transition rate is

$$q^{\boldsymbol{a}}\big((\boldsymbol{F},\boldsymbol{P},\boldsymbol{C},0),(\boldsymbol{F}+\boldsymbol{e}_k,\boldsymbol{P}-\boldsymbol{e}_k,\boldsymbol{C}^{j,N},j)\big) = \mu_0.$$

Type 2.1 *Last event:* failure of machine $j \in \mathcal{J}$; *action:* $\boldsymbol{a} = (x,y,z) \in \mathcal{A}_1(\boldsymbol{F},\boldsymbol{P},\boldsymbol{C},j)$. The last event is a failure of machine j, and the action is to dispatch a spare part from a warehouse x. The action may also include a relocation of one spare part from warehouse y to warehouse $z = x$. The state of the system immediately after the action is taken is $\boldsymbol{X} = (\boldsymbol{F}-\boldsymbol{e}_x\mathbb{1}\{y=-1\}-\boldsymbol{e}_y,\boldsymbol{P}+\boldsymbol{e}_x,\boldsymbol{C},j)$. The transition rates are defined as follows.

1. *Next event:* replenishment at k, $(\boldsymbol{P}+\boldsymbol{e}_x)_k > 0$. The corresponding transition rate is

$$q^{\boldsymbol{a}}\big((\boldsymbol{F},\boldsymbol{P},\boldsymbol{C},j),(\boldsymbol{F}-\boldsymbol{e}_x\mathbb{1}\{y=-1\}-\boldsymbol{e}_y+\boldsymbol{e}_k,\boldsymbol{P}+\boldsymbol{e}_x-\boldsymbol{e}_k,\boldsymbol{C},0)\big) = (\boldsymbol{P}+\boldsymbol{e}_x)_k\gamma.$$

2. *Next event:* failure of machine l, $C_l > 0$. The corresponding transition rate is

$$q^{\boldsymbol{a}}\big((\boldsymbol{F},\boldsymbol{P},\boldsymbol{C},j),(\boldsymbol{F}-\boldsymbol{e}_x\mathbb{1}\{y=-1\}-\boldsymbol{e}_y,\boldsymbol{P}+\boldsymbol{e}_x,\boldsymbol{C}^{l,0},l)\big) = \alpha_{C_l}\mu_{C_l}.$$

3. *Next event:* degradation of machine l, $C_l > 1$. The corresponding transition rate is

$$q^{\boldsymbol{a}}\big((\boldsymbol{F},\boldsymbol{P},\boldsymbol{C},j),(\boldsymbol{F}-\boldsymbol{e}_x\mathbb{1}\{y=-1\}-\boldsymbol{e}_y,\boldsymbol{P}+\boldsymbol{e}_x,\boldsymbol{C}^{l,C_l-1},l)\big) = (1-\alpha_{C_l})\mu_{C_l}.$$

4. *Next event:* repair of machine l, $C_l = 0$. The corresponding transition rate is

$$q^{\boldsymbol{a}}\big((\boldsymbol{F},\boldsymbol{P},\boldsymbol{C},j),(\boldsymbol{F}-\boldsymbol{e}_x\mathbb{1}\{y=-1\}-\boldsymbol{e}_y,\boldsymbol{P}+\boldsymbol{e}_x,\boldsymbol{C}^{l,N},l)\big) = \mu_0.$$

Type 2.2 *Last event:* degradation of machine $j \in \mathcal{J}$; *action:* $\boldsymbol{a} = (x,y,z) \in \mathcal{A}_1(\boldsymbol{F},\boldsymbol{P},\boldsymbol{C},j)$. The last event is a degradation of machine j that is not a failure. A spare part from a warehouse x is dispatched for preventive maintenance of the machine. The action may also include a relocation of one spare part from warehouse y to warehouse $z = x$. As preventive maintenance assumed to be done instantaneously, the state of the system immediately after the action is taken is $\boldsymbol{X} = (\boldsymbol{F}-\boldsymbol{e}_x\mathbb{1}\{y=-1\}-\boldsymbol{e}_y,\boldsymbol{P}+\boldsymbol{e}_x,\boldsymbol{C}^{j,N})$. The transition rates are defined as follows.

1. *Next event:* replenishment at k, $(\boldsymbol{P}+\boldsymbol{e}_x)_k > 0$. The corresponding transition rate is

$$q^{\boldsymbol{a}}\big((\boldsymbol{F},\boldsymbol{P},\boldsymbol{C},j),(\boldsymbol{F}-\boldsymbol{e}_x\mathbb{1}\{y=-1\}-\boldsymbol{e}_y+\boldsymbol{e}_k,\boldsymbol{P}+\boldsymbol{e}_x-\boldsymbol{e}_k,\boldsymbol{C}^{j,N},0)\big)=(\boldsymbol{P}+\boldsymbol{e}_x)_k\gamma.$$

2. *Next event:* failure of machine l, $C_l > 0$. The corresponding transition rate is

$$q^{\boldsymbol{a}}\big((\boldsymbol{F},\boldsymbol{P},\boldsymbol{C},j),(\boldsymbol{F}-\boldsymbol{e}_x\mathbb{1}\{y=-1\}-\boldsymbol{e}_y,\boldsymbol{P}+\boldsymbol{e}_x,\boldsymbol{C}^{j,N^{l,0}},l)\big)=\alpha_{C_l}\mu_{C_l}.$$

3. *Next event:* degradation of machine l, $C_l > 1$. The corresponding transition rate is

$$q^{\boldsymbol{a}}\big((\boldsymbol{F},\boldsymbol{P},\boldsymbol{C},j),(\boldsymbol{F}-\boldsymbol{e}_x\mathbb{1}\{y=-1\}-\boldsymbol{e}_y,\boldsymbol{P}+\boldsymbol{e}_x,\boldsymbol{C}^{j,N^{l,C_l-1}},l)\big)=(1-\alpha_{C_l})\mu_{C_l}.$$

4. *Next event:* degradation of machine j. The corresponding transition rate is

$$q^{\boldsymbol{a}}\big((\boldsymbol{F},\boldsymbol{P},\boldsymbol{C},j),(\boldsymbol{F}-\boldsymbol{e}_x\mathbb{1}\{y=-1\}-\boldsymbol{e}_y,\boldsymbol{P}+\boldsymbol{e}_x,\boldsymbol{C}^{j,N-1},j)\big)=(1-\alpha_N)\mu_N.$$

5. *Next event:* repair of machine l, $C_l = 0$. The corresponding transition rate is

$$q^{\boldsymbol{a}}\big((\boldsymbol{F},\boldsymbol{P},\boldsymbol{C},j),(\boldsymbol{F}-\boldsymbol{e}_x\mathbb{1}\{y=-1\}-\boldsymbol{e}_y,\boldsymbol{P}+\boldsymbol{e}_x,\boldsymbol{C}^{j,N^{l,N}},l)\big)=\mu_0.$$

Type 3 *Last event:* degradation or repair of machine $j \in \mathcal{J}$; *action:* $\boldsymbol{a} = (x,y,z) \in \mathcal{A}_2(\boldsymbol{F},\boldsymbol{P},\boldsymbol{C},j)$.
The last event is either a repair of machine j or a degradation that is not a failure. No preventive maintenance is done, so $x = -1$. However, a relocation of one spare part between any two warehouses y and z is possible. The state of the system immediately after the action is taken is $\boldsymbol{X} = (\boldsymbol{F}-\boldsymbol{e}_y+\boldsymbol{e}_z,\boldsymbol{P},\boldsymbol{C},j)$. The transition rates are defined as follows.

1. *Next event:* replenishment at k, $P_k > 0$. The corresponding transition rate is

$$q^{\boldsymbol{a}}\big((\boldsymbol{F},\boldsymbol{P},\boldsymbol{C},j),(\boldsymbol{F}-\boldsymbol{e}_y+\boldsymbol{e}_z+\boldsymbol{e}_k,\boldsymbol{P}-\boldsymbol{e}_k,\boldsymbol{C},0)\big)=P_k\gamma.$$

2. *Next event:* failure of machine l, $C_l > 0$. The corresponding transition rate is

$$q^{\boldsymbol{a}}\big((\boldsymbol{F},\boldsymbol{P},\boldsymbol{C},j),(\boldsymbol{F}-\boldsymbol{e}_y+\boldsymbol{e}_z,\boldsymbol{P},\boldsymbol{C}^{l,0},l)\big)=\alpha_{C_l}\mu_{C_l}.$$

3. *Next event:* degradation of machine l, $C_l > 1$. The corresponding transition rate is

$$q^{\boldsymbol{a}}\big((\boldsymbol{F},\boldsymbol{P},\boldsymbol{C},j),(\boldsymbol{F}-\boldsymbol{e}_y+\boldsymbol{e}_z,\boldsymbol{P},\boldsymbol{C}^{l,C_l-1},l)\big)=(1-\alpha_{C_l})\mu_{C_l}.$$

4. *Next event:* repair of machine l, $C_l = 0$. The corresponding transition rate is

$$q^{\boldsymbol{a}}\big((\boldsymbol{F},\boldsymbol{P},\boldsymbol{C},j),(\boldsymbol{F}-\boldsymbol{e}_y+\boldsymbol{e}_z,\boldsymbol{P},\boldsymbol{C}^{l,N},l)\big)=\mu_0.$$

Uniformization

To be able to compute the optimal policy, we uniformize our Markov process $\boldsymbol{X}^{\boldsymbol{a}}(t)$. To do so, we introduce the constant $\tau = \gamma K + J\max_{n\in\{0,\dots,N\}}\mu_n$ that is larger than the total transition rate from any state, and add the following dummy transitions that make the total transition rate from any state equal to τ. We add the following dummy transitions for each type of transition described above with $\boldsymbol{a}=\{-1,-1,-1\}$.

Type 1:

$$\begin{aligned}q^{\boldsymbol{a}}\big((\boldsymbol{F},\boldsymbol{P},\boldsymbol{C},0),(\boldsymbol{F},\boldsymbol{P},\boldsymbol{C},0)\big)&=\tau-\sum_{k=1}^{I}P_k\gamma-\sum_{l\in\mathcal{J}_1}\alpha_{C_l}\mu_{C_l}-\sum_{l\in\mathcal{J}_2}(1-\alpha_{C_l})\mu_{C_l}-\mu_0|\mathcal{J}_0|\\&=\tau-\sum_{k=1}^{I}P_k\gamma-\sum_{l\in\mathcal{J}_1}\mu_{C_l}-\mu_0|\mathcal{J}_0|\\&=\tau-\sum_{k=1}^{I}P_k\gamma-\sum_{l\in\mathcal{J}}\mu_{C_l},\end{aligned}$$

where $\mathcal{J}_0=\{l\in\mathcal{J}:C_l=0\}$, $\mathcal{J}_1=\{l\in\mathcal{J}:C_l>0\}$ and $\mathcal{J}_2=\{l\in\mathcal{J}:C_l>1\}$.

Type 2.1:

$$q^{\boldsymbol{a}}\big((\boldsymbol{F},\boldsymbol{P},\boldsymbol{C},j),(\boldsymbol{F}-\boldsymbol{e}_x\mathbb{1}\{y=-1\}-\boldsymbol{e}_y,\boldsymbol{P}+\boldsymbol{e}_x,\boldsymbol{C},0)\big)=\tau-\sum_{k=1}^{I}(\boldsymbol{P}+\boldsymbol{e}_x)_k\gamma-\sum_{l\in\mathcal{J}}\mu_{C_l}.$$

Type 2.2:

$$\begin{aligned}q^{\boldsymbol{a}}\big((\boldsymbol{F},\boldsymbol{P},\boldsymbol{C},j),(\boldsymbol{F}-\boldsymbol{e}_x\mathbb{1}\{y=-1\}-\boldsymbol{e}_y,\boldsymbol{P}+\boldsymbol{e}_x,\boldsymbol{C}^{j,N},0)\big)=&\tau-\sum_{k=1}^{I}(\boldsymbol{P}+\boldsymbol{e}_x)_k\gamma\\&-\sum_{l\in\mathcal{J}}\mu_{C_l}+\mu_{C_j}-\mu_N.\end{aligned}$$

Type 3:

$$q^{\boldsymbol{a}}\big((\boldsymbol{F},\boldsymbol{P},\boldsymbol{C},j),(\boldsymbol{F}-\boldsymbol{e}_y+\boldsymbol{e}_z,\boldsymbol{P},\boldsymbol{C},0)\big)=\tau-\sum_{k=1}^{I}P_k\gamma-\sum_{l\in\mathcal{J}}\mu_{C_l}.$$

6.3.3 Costs

We consider a generic cost structure that would allow to study various types of settings and examine the effects of different actions on the optimal policy. We incorporate the following cost components that are common in research literature on lateral transshipments and CMB. Let c_{cs} and c_{ps} be fixed setup costs for corrective and preventive maintenance, respectively, given that a spare part is dispatched from a local warehouse. In case a spare part is dispatched from the central warehouse, the costs c_e are incurred, independent of the type of maintenance. Let c_{rs} denote the setup costs incurred per relocation, and c_r - the replenishment setup costs. Assume that for each pair of local warehouse i and machine j the corresponding response time R_{ij} is deterministic and known. A fixed penalty c_{cl} is incurred if response time to a failed machine is larger than a given time threshold t^*, and an extra penalty of c_{cp} per time unit of delay over t^*. We assume that, in case a spare part is dispatched from the central warehouse, response time is always smaller than the time threshold t^*, independent of the machine. The immediate costs of action $\boldsymbol{a}(\boldsymbol{X}) = (x, y, z)$ in state $\boldsymbol{X} = (\boldsymbol{F}, \boldsymbol{P}, \boldsymbol{C}, j)$ can be computed as follows:

$$
\begin{aligned}
&c(\boldsymbol{X}, \boldsymbol{a}(\boldsymbol{X})) = \\
&= \begin{cases}
c_e & \text{if } x = 0, \\
c_{cs} + c_r + (c_{cl} + c_{cp}(R_{xj} - t^*))\mathbb{1}\{R_{xj} > t^*\} + c_{rs}\mathbb{1}\{y > 0\} & \text{if } x > 0 \text{ and } C_j = 0, \\
c_{ps} + c_r + c_{rs}\mathbb{1}\{y > 0\} & \text{if } x > 0 \text{ and } C_j > 0, \\
c_{rs}\mathbb{1}\{y > 0\} & \text{if } x = -1.
\end{cases}
\end{aligned}
$$

6.3.4 Optimality Equations

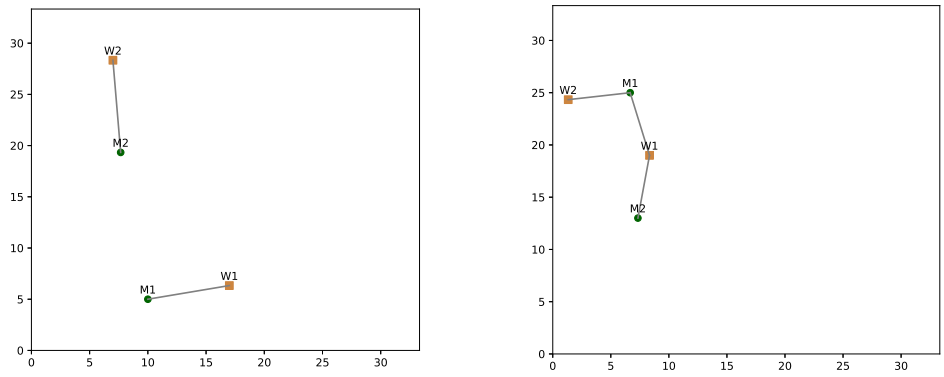
We formulate the problem as an infinite-horizon discounted MDP. Let $V(\boldsymbol{X})$ denote the expected total discounted costs under the optimal policy, when starting in state $\boldsymbol{X}$. Then $V(\boldsymbol{X})$ satisfies the Bellman equations:

$$
V(\boldsymbol{X}) = \sup_{\boldsymbol{a} \in \mathcal{A}(\boldsymbol{X})} \left\{ c(\boldsymbol{X}, \boldsymbol{a}) + \sum_{\boldsymbol{X'} \in \mathcal{S}} \lambda p(\boldsymbol{X'} | \boldsymbol{X}, \boldsymbol{a}) V(\boldsymbol{X'}) \right\}, \tag{6.3}
$$

where $\lambda < 1$ is a discount factor.

6.4 Numerical Experiments

In this section we conduct a number of experiments to study the performance and the structure of the optimal policy. To compute the optimal policy, we use the policy iteration algorithm with the maximum number of iterations set to 1000. All experiments are run in Python 3.7 on a computer with 8 GB RAM, Intel Core i5-5250U 1.6 GHz processor, running Linux Fedora 30.



(a) (b)

Figure 6.2: Examples of problem instances. The warehouses are connected to the machines that are reachable within t^* time units

6.4.1 Experimental Setup

Parameters. The following parameters are fixed throughout all experiments: the time threshold $t^* = 10$, the discount factor $\lambda = 0.95$, the number of warehouses $I = 2$, the number of machines $J = 2$, and the inventory level $K = 2$. Note that we only consider small problem instances, as due to the curse of dimensionality, it would be infeasible to derive optimal policy for multiple instances and for a wide range of parameter settings. We also assume $\mu_i = 1$ $(i = 0, 1, ..., N)$, and $\alpha_i = 0$ $(i = 2, ..., N)$. To study the system performance under the different levels of workload, we introduce the load parameter $\rho = \frac{J}{N\gamma K}$. For given values of ρ and N, we adjust γ accordingly.

Response times. An important component of a problem instance is a matrix $\boldsymbol{R}$ of fixed response times R_{ij} between each pair of warehouse i and machine j. The matrix $\boldsymbol{R}$ is used to compute the immediate costs in Section 6.3.3. For a given random seed, we construct it as follows. Machines and warehouses are allocated at random within a square of size 33×33 in terms of time units, such that each warehouse is within $t^* = 10$ time units from at least one machine, and each machine is within $t^* = 10$ time units from at least one warehouse. The response times R_{ij} are then computed as the corresponding Euclidean distances. Figure 6.2 presents two examples of problem instances used in this study.

Policy types. To study the effects of different types of actions on the policy performance,

and in particular, the effects of condition-based maintenance, we introduce the following five types of policies that are defined by limiting the original action spaces (6.1) and (6.2) as follows:

1. **Closest-First corrective maintenance (CF).**
   With this policy type only corrective maintenance is done, using the closest available spare part in terms of response time. The central warehouse stock is used only if all local warehouses are empty. The corresponding action space is defined as follows:

$$\mathcal{A}_{CF}(\boldsymbol{X}) = \begin{cases} \{(x,y,z)|x = \underset{i\in\mathcal{W}(\boldsymbol{X})}{\arg\min} R_{ij},\ y=-1,\ z=-1\}, & C_j = 0,\ \mathcal{W}(\boldsymbol{X}) \neq \emptyset, \\ \{(0,-1,-1)\}, & C_j = 0,\ \mathcal{W}(\boldsymbol{X}) = \emptyset, \\ \{(-1,-1,-1)\}, & \text{otherwise.} \end{cases}$$

   Note that with this policy type there is exactly one action per state. So, there is no need to use policy iteration, and the value function can be obtained by solving a set of linear equations:

$$V(\boldsymbol{X}) = c(\boldsymbol{X}, \boldsymbol{a}_{CF}(\boldsymbol{X})) + \sum_{\boldsymbol{X'}\in\mathcal{S}} \lambda p(\boldsymbol{X'}|\boldsymbol{X}, \boldsymbol{a}_{CF}(\boldsymbol{X}))V(\boldsymbol{X'}),$$

   where $\boldsymbol{a}_{CF}(\boldsymbol{X})$ is the action taken in state $\boldsymbol{X}$ under the CF policy.

2. **Optimal Corrective maintenance (OC).**
   With this policy type only corrective maintenance is done that is subject to optimization as in equations (6.3). The corresponding action space is defined as follows:

$$\mathcal{A}_{OC}(\boldsymbol{X}) = \begin{cases} \{(x,y,z) \in \mathcal{A}_1(\boldsymbol{X})|y=-1\}, & C_j = 0, \\ \{(-1,-1,-1)\}, & \text{otherwise.} \end{cases}$$

3. **Optimal Corrective maintenance with Relocation (OCR).**
   With this policy both corrective maintenance and relocation actions are optimized, given that no preventive maintenance is done. Relocation is also allowed upon a change of a machine state that is not a failure. The corresponding action space is defined as follows:

$$\mathcal{A}_{OCR}(\boldsymbol{X}) = \begin{cases} \mathcal{A}_1(\boldsymbol{X}), & C_j = 0, \\ \mathcal{A}_2(\boldsymbol{X}), & \text{otherwise.} \end{cases}$$

4. **Optimal Corrective & Preventive maintenance (OCP).**
   With this policy type both corrective and preventive maintenance are subject to

optimization, and no relocations are allowed. The corresponding action space is defined as follows:

$$\mathcal{A}_{OC}(\boldsymbol{X}) = \{(x, y, z) \in \mathcal{A}_1(\boldsymbol{X}) | y = -1\}$$

5. **Optimal Corrective & Preventive maintenance with Relocation (OCPR).** The last policy type corresponds to the full action space defined in Section 6.3.1:

$$\mathcal{A}(\boldsymbol{X}) = \mathcal{A}_1(\boldsymbol{X}, j) \cup \mathcal{A}_2(\boldsymbol{X}, j).$$

Performance measures. Solving the Bellman equations gives a value function $\boldsymbol{V}$ with the total expected discounted costs per state. To measure the policy performance we use the weighted average of the components of $\boldsymbol{V}$, where the steady state probabilities under the optimal policy are used as the weights. The steady state probabilities vector $\boldsymbol{\pi}$ is computed by solving the system of linear equations:

$$\begin{cases} \boldsymbol{\pi P} = \boldsymbol{\pi}, \\ \sum_{i=1}^{|\mathcal{S}|} \boldsymbol{\pi} = 1, \end{cases}$$

where $\boldsymbol{P}$ is the matrix of transition probabilities under the optimal policy. The weighted average of the value function is denoted by $\upsilon = \boldsymbol{\pi}'\boldsymbol{V}$.

For the policy types 2 to 5 we also report the relative improvement over the CF policy denoted by Δ. For example, for the optimal OCPR policy we define

$$\Delta_{OCPR} = \frac{\upsilon_{CF} - \upsilon_{OCPR}}{\upsilon_{CF}} \times 100\%.$$

6.4.2 Different Cost Settings

In this section we study the performance of the different policies depending on the cost setting and the load. The general assumption we make when choosing the values for cost parameters are aligned with the research literature on spare parts management and are as follows. The setup costs for relocation and dispatching preventively are lower than the setup costs for dispatching correctively from a local warehouse. Dispatching from the central warehouse has higher setup costs than corrective dispatching from a local warehouse. as it is supposed to be done only in emergency situations.

We study the policies' performance under different values of the load parameter ρ. For each combination of the cost parameters and the load ρ, 30 random instances are generated as described in Section 6.4.1, and the average performance is computed.

Table 6.1: Average performance of the policies per cost setting and load over 30 problem instances

		Average v					**Average Δ**			
Cost setting	ρ	**CF**	**OC**	**OCR**	**OCP**	**OCPR**	**OC**	**OCR**	**OCP**	**OCPR**
1	**1**	7.19	7.12	6.57	6.98	6.57	1.0%	8.6%	2.9%	8.7%
	0.7	5.37	5.29	4.79	4.89	4.54	1.5%	10.8%	9.0%	15.6%
	0.5	4.02	3.94	3.50	3.48	3.16	2.1%	13.0%	13.6%	21.5%
	0.3	2.54	2.46	2.13	2.09	1.85	3.3%	16.3%	17.6%	27.3%
2	**1**	63.67	63.62	63.19	62.18	61.89	0.1%	0.8%	2.3%	2.8%
	0.7	46.03	45.97	45.58	41.49	41.22	0.1%	1.0%	9.9%	10.5%
	0.5	33.30	33.24	32.90	28.10	27.86	0.2%	1.2%	15.6%	16.3%
	0.3	19.91	19.85	19.60	15.74	15.57	0.3%	1.6%	21.0%	21.8%
3	**1**	5.24	5.15	4.48	5.15	4.48	1.7%	14.4%	1.7%	14.4%
	0.7	3.55	3.45	2.84	3.28	2.84	2.8%	20.0%	7.6%	19.8%
	0.5	2.38	2.28	1.74	2.01	1.62	4.5%	26.8%	15.7%	32.0%
	0.3	1.25	1.15	0.74	0.93	0.63	8.3%	40.6%	25.7%	49.8%

Table 6.1 presents the obtained results.

The immediate costs in Section 6.3.3 depend on seven parameters, and it is infeasible to cover all possible cases. Hence, we choose the following three different cost settings:

1. $c_p = 0.05$, $c_r = 0$, $c_{cs} = 1$, $c_{cl} = 1$, $c_{ps} = 0.2$, $c_{rs} = 0.2$, $c_e = 10$;
2. $c_p = 0.1$, $c_r = 0$, $c_{cs} = 10$, $c_{cl} = 1$, $c_{ps} = 0.2$, $c_{rs} = 0.2$, $c_e = 100$;
3. $c_p = 0$, $c_r = 0$, $c_{cs} = 0$, $c_{cl} = 1$, $c_{ps} = 0$, $c_{rs} = 0$, $c_e = 10$.

Setting 1 corresponds to machinery of *moderate criticality*, where it is important to address breakdowns within the given time limit. The delay in response time is also penalized, although not significantly. Setting 2 corresponds to *critical machinery*, where breakdowns are very costly independent of response time. There is also a larger penalty for the delay in response time. Setting 3 corresponds to the case where breakdowns are *not critical* as long as they are taken care of within the time limit. For all three cost settings we choose relocation and preventive maintenance setup costs to be noticeably lower than the corrective setup costs and equal to each other.

For each cost setup and each value of the load parameter ρ we generate 30 different problem instances. For instances and compute the average performance of the optimal

policy for each of the policy types. The obtained results are reported in Table 6.1. We observe that the largest improvement over the CF policy is obtained under the cost setting 3 for all of the other policy types. Note that for the cost setting 2, where breakdowns are critical, optimal corrective maintenance and relocation have only marginal effect, while preventive maintenance results in a significant reduction in costs. For cost settings 1 and 3 doing relocations (OCR) has a bigger effect than doing preventive maintenance (OCP). Independently of the cost settings, all four types of policies with decision optimization improve over CF, with OCRP showing the best performance. The relative improvement over CF increases as the load ρ decreases, which is intuitive, as there are more opportunities to deviate from the CF policy.

6.4.3 Importance of Better Condition Diagnostics

With better diagnostics we can more accurately identify at which point of a degradation process a machine is. We model improvement in diagnostics by decomposing the degradation process in a larger number of intermediate steps, that is, by increasing N while keeping the load ρ fixed. We consider the cost setting 1, and as before, use 30 instances per parameter setting. Table 6.2 shows that the average Δ of OCP and OCRP policies increases significantly with N for different loads. This means the contribution of preventive maintenance grows with N, demonstrating the importance of accurate diagnostics.

6.4.4 Balancing Relocation and Preventive Maintenance

In this section we show an example of how relocation actions are balanced with preventive maintenance actions in the optimal OCPR policy. We consider the problem instance from Figure 6.2a and fix the parameters $\rho = 0.5$ and $N = 2$. We vary the cost components c_{ps} and c_{rs} in range $[0, 1.5]$ each, with other components fixed as in the cost setting 1. For each combination we compute the total number of states where relocation (preventive maintenance) is done in the optimal policy, divided by the total number of states where relocation (preventive maintenance) is possible. In Figure 6.3 this metric is plotted against c_{ps} and c_{rs} for both prevention and relocation actions. We observe that both types of actions take place in the optimal policy while both c_{ps} and c_{rs} are relatively low. When one of the two cost components increases, the optimal policy leans towards either of the two with lower setup costs, and when both c_{ps} and c_{rs} are large, the optimal policy does not include either relocation or preventive maintenance actions.

Table 6.2: Average performance of the policies for different N over 30 problem instances

		Average v					Average Δ			
ρ	N	CF	OC	OCR	OCP	OCPR	OC	OCR	OCP	OCPR
1	2	7.19	7.12	6.57	6.98	6.57	1.0%	8.6%	2.9%	8.7%
	3	6.69	6.64	6.20	5.71	5.44	0.8%	7.4%	14.7%	18.7%
	4	6.03	5.99	5.62	4.70	4.55	0.7%	6.8%	22.1%	24.5%
	5	5.42	5.39	5.07	3.73	3.65	0.6%	6.4%	31.2%	32.7%
	6	4.89	4.86	4.59	3.10	3.23	0.6%	6.2%	36.6%	33.9%
0.5	2	4.02	3.94	3.50	3.48	3.16	2.1%	13.0%	13.6%	21.5%
	3	4.06	4.00	3.95	2.98	2.93	1.6%	2.8%	26.6%	28.0%
	4	3.88	3.83	3.84	2.27	2.14	1.4%	0.9%	41.6%	45.0%
	5	3.64	3.59	3.39	1.74	1.75	1.2%	6.8%	52.0%	51.7%
	6	3.39	3.35	3.09	1.35	1.36	1.1%	8.9%	60.1%	59.9%

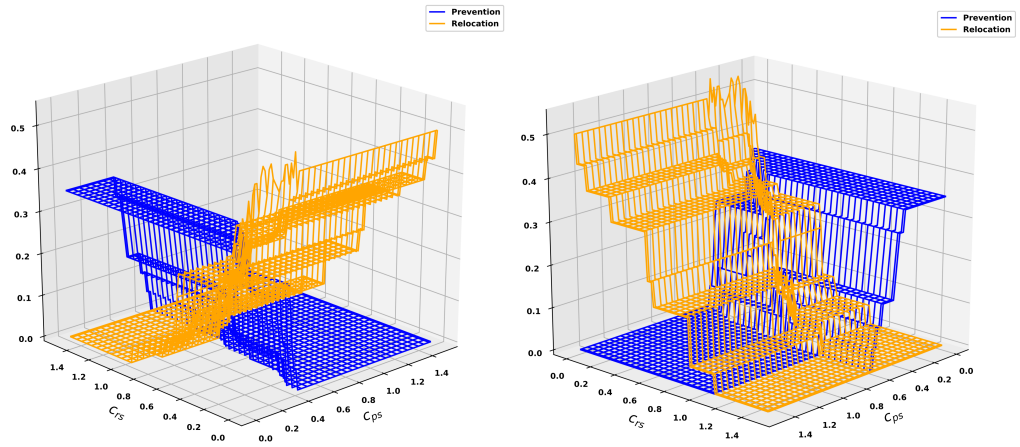


Figure 6.3: Relative number of states with relocation / preventive maintenance actions in the optimal OCPR policy as a function of c_{ps} and c_{rs}

6.5 Conclusion

The work in this chapter is a pioneering contribution to the field of dynamic spare parts management. We introduce the concept of condition-based maintenance into the problem of dynamic dispatching and relocation of spare parts in a service network, and study the effects of this on the optimal policy. With the degradation process explicitly incorporated into the model, preventive maintenance of the machines and proactive relocation of spare parts become possible based on the current condition of all machines in the network as well as the availability and spatial distribution of resources.

We formulate the problem as an MDP, and study the optimal performance of various types of policies to evaluate the relative contribution of introducing CBM in a spare parts network. To that end, we conduct numerical experiments with a different cost settings, and show that the policies that use the information about the condition of the machines outperform those not doing so. We also demonstrate that better condition diagnostics can further improve the CBM based policy performance.

Due to the curse of dimentionality, solving MDP is computationally infeasible for large networks. Hence, in this work we only consider small problem instances. Given the benefits of introducing CBM on a network that we show in this chapter, further research should focus on developing scalable heuristic approaches to the problem that would work for the problem instances of realistic sizes. Another interesting direction for further research is the parametric study of the degradation process. One could consider the effects of the corresponding parameters on the policy structure and its performance.

Bibliography

[1] R. Alanis, A. Ingolfsson, and B. Kolfal. A Markov chain model for an EMS system with repositioning. *Production and Operations Management*, 22(1):216–231, 2013.

[2] T. Andersson and P. Värbrand. Decision support tools for ambulance dispatch and relocation. *Journal of the Operational Research Society*, 58(2):195–201, 2007.

[3] D. Astaraky and J. Patrick. A simulation based approximate dynamic programming approach to multi-class, multi-resource surgical scheduling. *European Journal of Operational Research*, 245(1):309–319, 2015.

[4] M. A. Badri, A. K. Mortagy, and C. A. Alsayed. A multi-objective model for locating fire stations. *European Journal of Operational Research*, 110(2):243–260, 1998.

[5] D. Bandara, M. E. Mayorga, and L. A. McLay. Optimal dispatching strategies for emergency vehicles to increase patient survivability. *International Journal of Operational Research*, 15(2):195–214, 2012.

[6] D. Bandara, M. E. Mayorga, and L. A. McLay. Priority dispatching strategies for EMS systems. *Journal of the Operational Research Society*, 65(4):572–587, 2014.

[7] V. Bélanger, A. Ruiz, and P. Soriano. Recent optimization models and trends in location, relocation, and dispatching of emergency medical vehicles. *European Journal of Operational Research*, 272(1):1–23, 2019.

[8] A. Benmerzouga and S. Harous. Optimal (m-failure p-repairmen) policies with random repair time. *Stochastic Analysis and Applications*, 17(3):327–338, 1999.

[9] O. Berman. Repositioning of distinguishable urban service units on networks. *Computers & Operations Research*, 8(2):105–118, 1981.

[10] S. Bhulai. Dynamic routing policies for multiskill call centers. *Probability in the Engineering and Informational Sciences*, 23(1):101–119, 2009.

[11] S. Bhulai and G. M. Koole. On the structure of value functions for threshold policies in queueing models. *Journal of Applied Probability*, 40(3):613–622, 2003.

[12] L. Brotcorne, G. Laporte, and F. Semet. Ambulance location and relocation models. *European Journal of Operational Research*, 147(3):451–463, 2003.

[13] R. E. Burkard, M. Dell'Amico, and S. Martello. *Assignment Problems*, chapter 6. Society for Industrial and Applied Mathematics, 2009.

[14] M. F. Candas and E. Kutanoglu. Benefits of considering inventory in service parts logistics network design problems with time-based service constraints. *IIE Transactions*, 39(2):159–176, 2007.

[15] G. M. Carter and E. J. Ignall. A simulation model of fire department operations: design and preliminary results. *IEEE Transactions on Systems Science and Cybernetics*, 6(4):282–293, 1970.

[16] P. Chevalier, I. Thomas, D. Geraets, E. Goetghebeur, O. Janssens, D. Peeters, and F. Plastria. Locating fire stations: an integrated approach for Belgium. *Socio-Economic Planning Sciences*, 46(2):173–182, 2012.

[17] R. Church and C. ReVelle. The maximal covering location problem. *Papers in Regional Science*, 32(1):101–118, 1974.

[18] M. A. Cohen and H. L. Lee. Out of touch with customer needs? Spare parts and after sales service. *MIT Sloan Management Review*, 31(2):55, 1990.

[19] M. S. Daskin. A maximum expected covering location model: formulation, properties and heuristic solution. *Transportation Science*, 17(1):48–70, 1983.

[20] M. S. Daskin and E. H. Stern. A hierarchical objective set covering model for emergency medical service vehicle deployment. *Transportation Science*, 15(2):137–152, 1981.

[21] D. Degel, L. Wiesche, S. Rachuba, and B. Werners. Reorganizing an existing volunteer fire station network in Germany. *Socio-Economic Planning Sciences*, 48(2):149–157, 2014.

[22] C. Drent, S. Kapodistria, and J. Resing. Condition-based maintenance policies under imperfect maintenance at scheduled and unscheduled opportunities. *Queueing Systems*, 93(3-4):269–308, 2019.

[23] C. Drent, M. O. Keizer, and G.-J. van Houtum. Dynamic dispatching and repositioning policies for service engineers. *European Journal of Operational Research*, 2020. To appear.

[24] S. Enayati, M. E. Mayorga, H. K. Rajagopalan, and C. Saydam. Real-time ambulance redeployment approach to improve service coverage with fair and restricted workload for EMS providers. *Omega*, 79:67–80, 2018.

[25] S. Enayati, O. Y. Özaltın, M. E. Mayorga, and C. Saydam. Ambulance redeployment and dispatching under uncertainty with personnel workload limitations. *IISE Transactions*, 50(9):777–788, 2018.

[26] E. Erkut, A. Ingolfsson, and G. Erdoğan. Ambulance location for maximum survival. *Naval Research Logistics*, 55(1):42–58, 2008.

[27] J. Fang, L. Zhao, J. C. Fransoo, and T. Van Woensel. Sourcing strategies in supply risk management: An approximate dynamic programming approach. *Computers & Operations Research*, 40(5):1371–1382, 2013.

[28] P. Feng, R. Y. Fung, and F. Wu. Preventive transshipment decisions in a multi-location inventory system with dynamic approach. *Computers & Industrial Engineering*, 104:1–8, 2017.

[29] M. Gendreau, G. Laporte, and F. Semet. Solving an ambulance location model by tabu search. *Location Science*, 5(2):75–88, 1997.

[30] M. Gendreau, G. Laporte, and F. Semet. A dynamic model and parallel tabu search heuristic for real-time ambulance relocation. *Parallel Computing*, 27(12):1641–1653, 2001.

[31] M. Gendreau, G. Laporte, and F. Semet. The maximal expected coverage relocation problem for emergency vehicles. *Journal of the Operational Research Society*, 57(1):22–28, 2006.

[32] F. Glover and M. Laguna. Tabu search. In *Handbook of Combinatorial Optimization*, pages 2093–2229. Springer, 1998.

[33] S. C. Graves. A multi-echelon inventory model for a repairable item with one-for-one replenishment. *Management Science*, 31(10):1247–1256, 1985.

[34] L. V. Green and P. J. Kolesar. Improving emergency responsiveness with management science. *Management Science*, 50(8):1001–1014, 2004.

[35] Gurobi Optimization, Inc. Gurobi Optimizer Reference Manual. *https://www.gurobi.com/documentation/7.0/refman/index.html*, 2017.

[36] R. Haijema and J. van der Wal. An MDP decomposition approach for traffic control at isolated signalized intersections. *Probability in the Engineering and Informational Sciences*, 22(4):587–602, 2008.

[37] K. Hogan and C. ReVelle. Concepts and applications of backup coverage. *Management Science*, 32(11):1434–1444, 1986.

[38] J. M. Hogg. The siting of fire stations. *Journal of the Operational Research Society*, 19(3):275–287, 1968.

[39] J. Huiskonen. Maintenance spare parts logistics: special characteristics and strategic choices. *International Journal of Production Economics*, 71(1-3):125–133, 2001.

[40] R.-H. Hwang, J. F. Kurose, and D. Towsley. MDP routing for multi-rate loss networks. *Computer Networks*, 34(2):241–261, 2000.

[41] A. P. Iannoni and R. Morabito. A multiple dispatch and partial backup hypercube queuing model to analyze emergency medical systems on highways. *Transportation Research Part E: Logistics and Transportation Review*, 43(6):755–771, 2007.

[42] A. P. Iannoni, R. Morabito, and C. Saydam. A hypercube queueing model embedded into a genetic algorithm for ambulance deployment on highways. *Annals of Operations Research*, 157(1):207–224, 2008.

[43] E. Ignall, G. Carter, and K. Rider. An algorithm for the initial dispatch of fire companies. *Management Science*, 28(4):366–378, 1982.

[44] K. Ines, H. Chabchoub, and B. Aouni. Goal programming model for fire and emergency service facilities site selection. *Information Systems and Operational Research*, 48(3):143–153, 2010.

[45] A. Ingolfsson. EMS planning and management. In *Operations Research and Health Care Policy*, pages 105–128. Springer, 2013.

[46] A. Ingolfsson, S. Budge, and E. Erkut. Optimal ambulance location with random delays and travel times. *Health Care Management Science*, 11(3):262–274, 2008.

[47] C. J. Jagtenberg, S. Bhulai, and R. D. van der Mei. An efficient heuristic for real-time ambulance redeployment. *Operations Research for Health Care*, 4:27–35, 2015.

[48] C. J. Jagtenberg, S. Bhulai, and R. D. van der Mei. Dynamic ambulance dispatching: is the closest-idle policy always optimal? *Health Care Management Science*, 20(4):517–531, 2017.

[49] C. J. Jagtenberg, P. L. van den Berg, and R. D. van der Mei. Benchmarking online dispatch algorithms for emergency medical services. *European Journal of Operational Research*, 258(2):715–725, 2017.

[50] J. P. Jarvis. Optimal assignments in a Markovian queueing system. *Computers & Operations Research*, 8(1):17–23, 1981.

[51] J. Jasper. Quick response solutions, fedex critical inventory logistics revitalized. FedEx White Paper, 2006.

[52] V. Jeet, E. Kutanoglu, and A. Partani. Logistics network design with inventory stocking for low-demand parts: Modeling and optimization. *IIE Transactions*, 41(5):389–407, 2009.

[53] R. Jiang, M. J. Kim, and V. Makis. Availability maximization under partial observations. *OR Spectrum*, 35(3):691–710, 2013.

[54] O. Karasakal and E. K. Karasakal. A maximal covering location model in the presence of partial coverage. *Computers & Operations Research*, 31(9):1515–1526, 2004.

[55] M. N. Katehakis and C. Derman. Optimal repair allocation in a series system. *Mathematics of Operations Research*, 9(4):615–623, 1984.

[56] J. P. Kharoufeh, C. J. Solo, and M. Y. Ulukus. Semi-markov models for degradation-based reliability. *IIE Transactions*, 42(8):599–612, 2010.

[57] V. A. Knight, P. R. Harper, and L. Smith. Ambulance allocation for maximal survival with heterogeneous outcome measures. *Omega*, 40(6):918–926, 2012.

[58] P. Kolesar and E. H. Blum. Square root laws for fire engine response distances. *Management Science*, 19(12):1368–1378, 1973.

[59] P. Kolesar and W. E. Walker. An algorithm for the dynamic relocation of fire companies. *Operations Research*, 22(2):249–274, 1974.

[60] G. Koole. A formula for tail probabilities of cox distributions. *Journal of Applied Probability*, 41(3):935–938, 2004.

[61] P. Koudal. The service revolution in global manufacturing industries. *Deloitte Research*, 2:1–22, 2006.

[62] S.-W. Lam, L.-H. Lee, and L.-C. Tang. An approximate dynamic programming approach for the empty container allocation problem. *Transportation Research Part C: Emerging Technologies*, 15(4):265–277, 2007.

[63] R. C. Larson. A hypercube queuing model for facility location and redistricting in urban emergency services. *Computers & Operations Research*, 1(1):67–95, 1974.

[64] R. C. Larson. Approximating the performance of urban emergency service systems. *Operations Research*, 23(5):845–868, 1975.

[65] S. Lee. The role of preparedness in ambulance dispatching. *Journal of the Operational Research Society*, 62(10):1888–1897, 2011.

[66] X. Li, Z. Zhao, X. Zhu, and T. Wyatt. Covering models and optimization techniques for emergency response facility location and planning: a review. *Mathematical Methods of Operations Research*, 74(3):281–310, 2011.

[67] C. S. Lim, R. Mamat, and T. Braunl. Impact of ambulance dispatch policies on performance of emergency medical services. *IEEE Transactions on Intelligent Transportation Systems*, 12(2):624–632, 2011.

[68] M. L. Littman, T. L. Dean, and L. P. Kaelbling. On the complexity of solving Markov decision problems. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 394–402, 1995.

[69] V. Marianov and C. ReVelle. The capacitated standard response fire protection siting problem: deterministic and probabilistic models. *Annals of Operations Research*, 40(1):303–322, 1992.

[70] M. S. Maxwell, S. G. Henderson, and H. Topaloglu. Tuning approximate dynamic programming policies for ambulance redeployment via direct search. *Stochastic Systems*, 3(2):322–361, 2013.

[71] M. S. Maxwell, M. Restrepo, S. G. Henderson, and H. Topaloglu. Approximate dynamic programming for ambulance redeployment. *INFORMS Journal on Computing*, 22(2):266–281, 2010.

[72] L. A. McLay and M. E. Mayorga. A dispatching model for server-to-customer systems that balances efficiency and equity. *Manufacturing & Service Operations Management*, 15(2):205–220, 2013.

[73] L. A. McLay and M. E. Mayorga. A model for optimally dispatching ambulances to emergency calls with classification errors in patient priorities. *IIE Transactions*, 45(1):1–24, 2013.

[74] J. Meissner and O. V. Senicheva. Approximate dynamic programming for lateral transshipment problems in multi-location inventory systems. *European Journal of Operational Research*, 265(1):49–64, 2018.

[75] J. A. Muckstadt. A model for a multi-item, multi-echelon, multi-indenture inventory system. *Management Science*, 20(4-part-i):472–481, 1973.

[76] D. Murthy, O. Solem, and T. Roren. Product warranty logistics: Issues and challenges. *European Journal of Operational Research*, 156(1):110–126, 2004.

[77] D. P. Murthy and N. Jack. *Extended warranties, maintenance service and lease contracts: modeling and analysis for decision-making*. Springer Science & Business, 2014.

[78] J. Naoum-Sawaya and S. Elhedhli. A stochastic optimization model for real-time ambulance redeployment. *Computers & Operations Research*, 40(8):1972–1978, 2013.

[79] A. A. Nasrollahzadeh, A. Khademi, and M. E. Mayorga. Real-time ambulance dispatching and relocation. *Manufacturing & Service Operations Management*, 20(3):467–480, 2018.

[80] J. M. Norman. *Heuristic Procedures in Dynamic Programming*. Manchester University Press, 1972.

[81] K. Öner, R. Franssen, G. Kiesmüller, and G. Van Houtum. Life cycle costs measurement of complex systems manufactured by an engineer-to-order company. In *The 17th International Conference on Flexible Automation and Intelligent Manufacturing*, pages 569–589, 2007.

[82] T. J. Ott and K. Krishnan. Separable routing: A scheme for state-dependent routing of circuit switched telephone traffic. *Annals of Operations Research*, 35(1):43–68, 1992.

[83] C. Paterson, G. Kiesmüller, R. Teunter, and K. Glazebrook. Inventory models with lateral transshipments: A review. *European Journal of Operational Research*, 210(2):125–136, 2011.

[84] C. Paterson, R. Teunter, and K. Glazebrook. Enhanced lateral transshipments in a multi-location inventory system. *European Journal of Operational Research*, 221(2):317–327, 2012.

[85] A. Pechina, D. Usanov, P. M. van de Ven, and R. D. van der Mei. Real-time dispatching and relocation of emergency service engineers. *European Journal of Operational Research*, 2019. In revision.

[86] H. Peng and G.-J. van Houtum. Joint optimization of condition-based maintenance and production lot-sizing. *European Journal of Operational Research*, 253(1):94–107, 2016.

[87] D. R. Plane and T. E. Hendrick. Mathematical programming and the location of fire companies for the denver fire department. *Operations Research*, 25(4):563–578, 1977.

[88] W. B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, volume 703. John Wiley & Sons, 2007.

[89] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

[90] J. A. Rappold and B. D. van Roo. Designing multi-echelon service parts networks with finite repair capacity. *European Journal of Operational Research*, 199(3):781–792, 2009.

[91] C. ReVelle and K. Hogan. The maximum availability location problem. *Transportation Science*, 23(3):192–200, 1989.

[92] D. A. Schilling, C. Revelle, J. Cohon, and D. J. Elzinga. Some models for fire protection locational decisions. *European Journal of Operational Research*, 5(1):1–7, 1980.

[93] V. Schmid. Solving the dynamic ambulance relocation and dispatching problem using approximate dynamic programming. *European Journal of Operational Research*, 219(3):611–621, 2012.

[94] H.-J. Schuetz and R. Kolisch. Approximate dynamic programming for capacity allocation in the service industry. *European Journal of Operational Research*, 218(1):239–250, 2012.

[95] C. C. Sherbrooke. Metric: A multi-echelon technique for recoverable item control. *Operations Research*, 16(1):122–141, 1968.

[96] H. Simao and W. Powell. Approximate dynamic programming for management of high-value spare parts. *Journal of Manufacturing Technology Management*, 20(2):147–160, 2009.

[97] H. P. Simao, J. Day, A. P. George, T. Gifford, J. Nienow, and W. B. Powell. An approximate dynamic programming algorithm for large-scale fleet management: A case application. *Transportation Science*, 43(2):178–197, 2009.

[98] K. Sudtachat, M. E. Mayorga, and L. A. McLay. Recommendations for dispatching emergency vehicles under multitiered response via simulation. *International Transactions in Operational Research*, 21(4):581–617, 2014.

[99] K. Sudtachat, M. E. Mayorga, and L. A. Mclay. A nested-compliance table policy for emergency medical service systems under relocation. *Omega*, 58:154–168, 2016.

[100] A. J. Swersey. A Markovian decision model for deciding how many fire companies to dispatch. *Management Science*, 28(4):352–365, 1982.

[101] H. G. Tiemessen, M. Fleischmann, G.-J. van Houtum, J. A. van Nunen, and E. Pratsini. Dynamic demand fulfillment in spare parts networks with multiple customer classes. *European Journal of Operational Research*, 228(2):367–380, 2013.

[102] C. Toregas, R. Swain, C. ReVelle, and L. Bergman. The location of emergency service facilities. *Operations Research*, 19(6):1363–1373, 1971.

[103] D. Usanov, G. A. G. Legemaate, P. M. van de Ven, and R. D. van der Mei. Fire truck relocation during major incidents. *Naval Research Logistics*, 66(2):105–122, 2019.

[104] D. Usanov, A. Pechina, P. M. van de Ven, and R. D. van der Mei. Approximate dynamic programming for real-time dispatching and relocation of emergency service engineers. Manuscript submitted for publication, 2019.

[105] D. Usanov, P. M. van de Ven, and R. D. van der Mei. Integrating condition-based maintenance into dynamic spare parts management. Manuscript submitted for publication, 2019.

[106] D. Usanov, P. M. van de Ven, and R. D. van der Mei. Dispatching fire trucks under stochastic driving times. *Computers & Operations Research*, 114, 2020.

[107] C. Valdez-Flores and R. M. Feldman. A survey of preventive maintenance models for stochastically deteriorating single-unit systems. *Naval Research Logistics*, 36(4):419–446, 1989.

[108] T. van Barneveld. The minimum expected penalty relocation problem for the computation of compliance tables for ambulance vehicles. *INFORMS Journal on Computing*, 28(2):370–384, 2016.

[109] T. van Barneveld, S. Bhulai, and R. D. van der Mei. The effect of ambulance relocations on the performance of ambulance service providers. *European Journal of Operational Research*, 252(1):257–269, 2016.

[110] T. van Barneveld, S. Bhulai, and R. D. van der Mei. A dynamic ambulance management model for rural areas. *Health Care Management Science*, 20(2):165–186, 2017.

[111] T. van Barneveld, C. J. Jagtenberg, S. Bhulai, and R. D. van der Mei. Real-time ambulance relocation: Assessing real-time redeployment strategies for ambulance relocation. *Socio-Economic Planning Sciences*, 62:129–142, 2018.

[112] T. van Barneveld, R. D. van der Mei, and S. Bhulai. Compliance tables for an EMS system with two types of medical response units. *Computers & Operations Research*, 80:68–81, 2017.

[113] P. L. van den Berg and K. Aardal. Time-dependent MEXCLP with start-up and relocation cost. *European Journal of Operational Research*, 242(2):383–389, 2015.

[114] P. L. van den Berg, G. J. Kommer, and B. Zuzáková. Linear formulation for the maximum expected coverage location model with fractional coverage. *Operations Research for Health Care*, 8:33–41, 2016.

[115] P. L. van den Berg, G. A. Legemaate, and R. D. Van Der Mei. Increasing the responsiveness of firefighter services by relocating base stations in amsterdam. *Interfaces*, 47(4):352–361, 2017.

[116] L. A. M. van Dongen. Maintenance engineering: Instand-houding van verbindingen. Oratie, UTwente, 2011.

[117] G.-J. van Houtum and B. Kranenburg. *Spare Parts Inventory Control under System Availability Constraints*, volume 227 of *International Series in Operations Research & Management Science*. Springer, 2015.

[118] S. van Wijk, I. Adan, and G.-J. van Houtum. Optimal lateral transshipment policy for a two location inventory problem. *European Journal of Operational Research*, 272(2):481–495, 2019.

[119] M. Vigoroso. Service parts management: unlocking value and profit in the service chain. Technical report, Aberdeen Group. URL http://www.aberdeen.com/summary/report/benchmark/serviceparts, 2003.

[120] H. Wang. A survey of maintenance policies of deteriorating systems. *European Journal of Operational Research*, 139(3):469–489, 2002.

[121] H. Wong, G.-J. van Houtum, D. Cattrysse, and D. van Oudheusden. Multi-item spare parts systems with lateral transshipments and waiting time constraints. *European Journal of Operational Research*, 171(3):1071–1093, 2006.

[122] M.-C. Wu, Y.-K. Hsu, and L.-C. Huang. An integrated approach to the design and operation for spare parts logistic systems. *Expert Systems with Applications*, 38(4):2990–2997, 2011.

[123] X. Yu and M. Gen. *Introduction to Evolutionary Algorithms*. Springer Science & Business Media, 2010.

[124] O. Zhang, A. Mason, and A. Philpott. Simulation and optimisation for ambulance logistics and relocation. Presentation at the INFORMS 2008 Conference, 2008.

[125] Q. Zhu, H. Peng, B. Timmermans, and G.-J. van Houtum. A condition-based maintenance model for a single component in a system with scheduled and unscheduled downs. *International Journal of Production Economics*, 193:365–380, 2017.

[126] Q. Zhu, H. Peng, and G.-J. van Houtum. A condition-based maintenance policy for multi-component systems with a high maintenance setup cost. *OR Spectrum*, 37(4):1007–1035, 2015.

Summary

Emergencies such as a breakdown of an MRI-scanner or a domestic fire demand a timely response. This means that the resources required for addressing such incidents (spare parts and fire trucks, respectively) need to be stored in relative proximity of potential incidents and dispatched on short notice. This necessitates a network of locations close to potential emergencies where the resources are stored. Operators of such service networks face the following operational questions:

1. How should resources be distributed over the service area and dispatched in response to an emergency?

2. How can the performance of emergency services be improved by proactive relocation of resources?

Answering this type of questions is essential to achieve the required timely response. In this thesis, we introduce the concept of Emergency Resource Networks (ERN's), unifying and extending results from the heretofore disjoint application areas of maintenance services and emergency services. More specifically, we focus on the following two examples of ERN's: *maintenance services for capital goods* and *fire fighting emergency services*. We address a number of operational problems from these application areas, while drawing inspiration from the research in both domains.

In Chapter 2, we start by studying one of the major problems faced by fire fighting services. Fire departments are designed and operated to minimize the response time to fires and other incidents requiring fire department presence. To this end, fire stations are positioned throughout the service area of a fire department to allow for a fast response to any incident, irrespective of its location. However, large gaps in coverage may arise in the case of a major incident that requires many nearby fire trucks over a long period

of time, substantially increasing response times for emergencies that occur subsequently. We address this problem and propose a heuristic for relocating idle trucks during a major incident in order to retain good coverage. This is done by solving a mathematical program that takes into account the locations of the available fire trucks and the historic spatial distribution of incidents. This heuristic allows the user to balance the coverage and the number of truck movements. By introducing the 'willingness to relocate' parameter to the algorithm, we provide the user with a tool to measure the value of making additional relocations. Using extensive simulation experiments we test the heuristic for the operations of the Fire Department of Amsterdam-Amstelland (FDAA), and compare it against three other benchmark strategies in a simulation fitted using ten years of historical data. We demonstrate substantial improvement over the current relocation policy used by FDAA, and show that smartly relocating during major incidents may lead to a significant improvement in performance. Additionally, we numerically illustrate the trade-off between the number of relocations and the system performance.

Fire departments typically dispatch the closest fire truck(s) available whenever a new incident happens. However, it is not obvious that the policy of always dispatching the closest truck(s) minimizes the long-run fraction of late arrivals, since it may leave gaps in the coverage for future incidents. Careful dispatching is even more important when multiple trucks are required, since the potential coverage gap is much larger compared to the single-truck case. Moreover, when dispatching multiple trucks, the uncertainty in the trucks' driving times plays an important role, in particular due to possible correlation in driving times of the trucks in case their routes overlap. In Chapter 3, we discuss optimal dispatching of fire trucks, based on a particular dispatching problem that arises at the FDAA, where two fire trucks are sent to the same incident location for a quick response. We formulate the dispatching problem as a Markov Decision Process, and numerically obtain the optimal dispatching decisions using policy iteration. We show that the fraction of late arrivals can be significantly reduced by deviating from current practice of dispatching the closest available trucks. We also show that driving-time correlation can have a non-negligible impact on decision making, and if ignored, may lead to a drastic performance decrease. As the optimal policy cannot be computed for problems of realistic size due to the computational complexity of the policy iteration algorithm, we propose a dispatching heuristic based on a queueing approximation for the state of the network. We show that the performance of this heuristic is close to the optimal policy for a wide range of parameter settings, and requires significantly less computational effort.

In Chapters 4 and 5, we turn to the field of maintenance services for capital goods. Capital goods such as complex medical equipment, trains and manufacturing machinery are essential to their users' business, and thus have stringent up-time requirements. Responsive maintenance is crucial for meeting these requirements, which in turn relies on

the timely availability of both spare parts and service engineers. In these two chapters, we consider a network of geographically distributed capital goods, maintained by a set of service engineers who can respond quickly to machine breakdowns. We are interested in the question which service engineers to dispatch to what breakdowns, and how to relocate these engineers to maintain good coverage. In Chapter 4, we propose and evaluate a range of scalable dispatching and relocation heuristics inspired by the extensive research literature in the domain of emergency medical services. We compare the proposed heuristics against each other using comprehensive simulation experiments, and benchmark the best combination of dispatching and relocation heuristics against the optimal policy. We find that this heuristic performs close to optimal, while easily scaling to realistic-sized networks, making it suitable for practical applications. In Chapter 5, we develop an approximate dynamic programming (ADP) approach to produce dispatching and relocation policies, and propose two new algorithms to tune the ADP policy. We conduct extensive computational experiments to compare the ADP policy against two benchmark policies, the best heuristic from Chapter 4 and the closest-first dispatching policy. These demonstrate that the ADP approach can generate high-quality solutions that outperform both benchmarks across a wide range of networks and parameters. We observe significant improvements in terms of fraction of late arrivals over the two benchmarks, without increase in average response time.

Finally, in Chapter 6, we consider the problem of dynamic dispatching and relocation of spare parts. We introduce a new model where the concept of condition-based maintenance is incorporated in a network setting with dynamic spare parts management. We consider a network of single-component machines and assume a Markovian degradation process, where a machine moves through a sequence of intermediate states before it reaches the failure state. A number of such machines are spread across a service region, and we optimize corrective and preventive maintenance actions, as well as proactive relocation of spare parts between stock points. We show that, by introducing condition monitoring into a network setting, significant improvements can be achieved in reducing total expected costs, independent of the cost structure.

Samenvatting

Noodgevallen zoals een defect van een MRI-scanner of een woningbrand vragen om een snelle reactie. Dit betekent dat de 'middelen' die nodig zijn om dergelijke incidenten aan te pakken (reserveonderdelen, brandweervoertuigen) aanwezig moeten zijn in de nabijheid van potentiële incidenten en heel snel moeten kunnen worden ingezet. Dit vereist een netwerk van locaties in de buurt van potentiële noodsituaties waar de middelen zijn gestationeerd. Operators van dergelijke servicenetwerken worden vaak geconfronteerd met de volgende operationele vragen:

1. Hoe moeten middelen worden verspreid over het servicegebied en hoe moeten ze worden ingezet in noodgevallen?

2. Hoe kunnen de prestaties van hulpdiensten worden verbeterd door slimme, proactieve relocaties van middelen?

Het beantwoorden van dit soort vragen is essentieel om de vereiste korte responstijden te realiseren. In dit proefschrift wordt het concept van Emergency Resource Networks (ERN's) geïntroduceerd, waarbij de resultaten van de toepassingsgebieden van onderhoudsdiensten enerzijds en van hulpdiensten anderzijds worden geïntegreerd en uitgebreid. Meer specifiek richten we ons op de volgende twee voorbeelden van ERN's: *onderhoudsdiensten voor kapitaalgoederen* en *brandweerhulpdiensten*. We bestuderen een aantal operationele problemen uit deze toepassingsgebieden, terwijl we inspiratie putten uit het onderzoek in beide domeinen.

In Hoofdstuk 2 beginnen we met het bestuderen van één van de grootste problemen waarmee brandweerdiensten te maken hebben. Brandweerdiensten zijn erop gericht om de responstijd op branden en andere incidenten die aanwezigheid van de brandweer vereisen te minimaliseren. Daartoe worden brandweerkazernes over het hele servicegebied van een

brandweer verspreid om snel op elk incident te kunnen reageren, ongeacht de locatie van het incident. Er kunnen zich echter grote gaten in de dekking voordoen in het geval van een groot incident dat gedurende een lange periode veel nabijgelegen brandweerauto's vereist, waardoor de responstijden voor noodgevallen die voordoen zich *tijdens* zo'n groot incident aanzienlijk toenemen. We pakken dit probleem aan en ontwikkelen een heuristiek voor het proactief verplaatsen van inactieve brandweervoertuigen tijdens een groot incident om een goede dekking te behouden. Daartoe formuleren we een optimalisatieprobleem dat rekening houdt met de huidige locaties van de beschikbare brandweerwagens en de geografische verspreiding van incidenten. Door deze heuristiek kan de gebruiker de dekking en het aantal voertuigbewegingen afwegen. Door de parameter 'bereidheid om te verplaatsen' in het algoritme te introduceren, bieden we de gebruiker de mogelijkheid een afweging te maken tussen het aantal relocaties enerzijds en de prestatie anderzijds. Aan de hand van uitgebreide simulatie-experimenten testen we de heuristiek voor de bedrijfsvoering van de brandweer van Amsterdam-Amstelland (FDAA), en vergelijken we deze met drie andere benchmarkstrategieën in een simulatie op basis van tien jaar aan historische ritgegevens. We tonen een substantiële verbetering aan ten opzichte van het huidige relocatiestrategie van de FDAA en laten zien dat slim verplaatsen tijdens grote incidenten kan leiden tot een aanzienlijke verbetering van de prestaties. Bovendien illustreren we de afweging tussen het aantal verplaatsingen en de systeemprestaties aan de hand van numerieke voorbeelden.

Brandweerdiensten sturen meestal de dichtstbijzijnde beschikbare brandweerwagen(s) op weg wanneer zich een nieuw incident voordoet. Het is echter niet vanzelfsprekend dat de strategie om altijd de dichtstbijzijnde vrachtwagen(s) uit te geven de lange-termijn fractie van late aankomsten minimaliseert, omdat deze strategie 'gaten' in de dekking voor toekomstige incidenten kan laten. Zorgvuldige relocatie is nog belangrijker wanneer meerdere wagens nodig zijn, omdat het potentiële 'gat' in de bedekking veel groter is in vergelijking met het geval met één enkel brandweervoertuig. Bovendien speelt bij het uitgeven van meerdere vrachtwagens de onzekerheid in de rijtijd van de brandweertrucks een belangrijke rol, met name vanwege de mogelijke correlatie in rijtijden van de trucks als hun routes elkaar overlappen. In Hoofdstuk 3 bestuderen we de optimale uitgifte van brandweerwagens, op basis van een specifiek toewijzingsprobleem dat zich bij de FDAA voordoet, waarbij twee brandweervoertuigen naar dezelfde incidentlocatie worden gestuurd voor een korte reactietijd. We formuleren het uitgifteprobleem als een Markov-beslissingsprobleem en verkrijgen numeriek de optimale uitgiftebeslissingen met behulp van zgn. policy-iteratie. We laten zien dat de fractie van late aankomsten aanzienlijk kan worden gereduceerd door af te wijken van de huidige praktijk van het uitgeven van de dichtstbijzijnde beschikbare brandweervoertuigen. We laten ook zien dat de correlatie in rijtijden een niet te verwaarlozen invloed kan hebben op de optimale strategie, en dat het negeren hiervan kan leiden tot een drastische prestatievermindering. De optimale strategie kan niet worden berekend voor problemen van realistische omvang vanwege de complexiteit

van het algoritme. Daarom stellen we een uitgifteheuristiek voor op basis van een wachtrijbenadering voor de status van het netwerk. We laten zien dat de prestaties van deze heuristiek dicht bij de optimale strategie liggen en aanzienlijk minder rekenkracht vereisen.

In Hoofdstukken 4 en 5 richten we ons op het gebied van onderhoudsdiensten voor kapitaalgoederen. Kapitaalgoederen zoals complexe medische apparatuur, treinen en productiemachines zijn essentieel voor de activiteiten van hun gebruikers en stellen daarom strikte eisen aan de up-time. Goede onderhoudsstrategieën zijn cruciaal om aan deze eisen te voldoen, en zijn op hun beurt afhankelijk van de tijdige beschikbaarheid van zowel reserveonderdelen als onderhoudsmonteurs. In deze twee hoofdstukken beschouwen we een netwerk van geografisch verspreide kapitaalgoederen, onderhouden door een set onderhoudsmonteurs die snel kunnen reageren op machinestoringen. We zijn geïnteresseerd in de vraag welke servicemonteurs naar welke storingen moeten worden gestuurd en hoe deze monteurs het best kunnen worden gereloceerd om een goede dekking te behouden. In Hoofdstuk 4 ontwikkelen en evalueren we een reeks schaalbare uitgifte- en relocatieheuristieken, geïnspireerd door de uitgebreide onderzoeksliteratuur op het gebied van medische noodhulp. We vergelijken de heuristieken met elkaar met behulp van uitgebreide simulatie-experimenten en vergelijken de beste combinatie van uitgifte- en relocatieheuristieken met de optimale strategie. We laten zien dat deze heuristiek bijna optimaal presteert, terwijl hij gemakkelijk kan worden opgeschaald naar realistische netwerken, waardoor het geschikt is voor praktische toepassingen. In Hoofdstuk 5 ontwikkelen we een ADP-benadering om relocatie- en uitgiftestrategieën te berekenen en stellen we twee nieuwe algoritmen voor om de ADP-benadering te finetunen. We voeren uitgebreide numerieke experimenten uit om het ADP-benadering te vergelijken met twee benchmarkstrategieën, de beste heuristiek uit Hoofdstuk 4 en de *closest-first* strategie. De resultaten laten zien dat de ADP-aanpak uitstekende oplossingen kan genereren die beter presteren dan de beide benchmarks over een breed scala van parametersettings. We zien aanzienlijke verbeteringen in termen van fractie van late aankomsten ten opzichte van de twee benchmarks, zonder dat de gemiddelde responstijd toeneemt.

Tot slot beschouwen we in Hoofdstuk 6 het probleem van dynamische uitgifte en verplaatsing van reserveonderdelen. We introduceren een nieuw model waarbij het concept van *condition-based maintenance* is opgenomen in een netwerkomgeving met dynamisch management van reserveonderdelen. We beschouwen een netwerk van machines met één component en gaan uit van een Markov-degradatieproces, waarbij een machine een reeks tussenliggende toestanden doorloopt voordat deze de foutstatus bereikt. Een aantal van dergelijke machines is verspreid over een serviceregio en we optimaliseren zowel correctieve als preventieve onderhoudsstrategiën, evenals proactieve relocatie van reserveonderdelen tussen voorraadpunten. De resulaten laten zien dat door het monitoren van de toestand in een netwerkomgeving significante verbeteringen kunnen worden bereikt bij het verlagen

van de totale verwachte kosten, onafhankelijk van de kostenstructuur.

About the Author

Dmitrii Usanov was born in Ryazan, Russia, on August 25, 1988. He finished his secondary education in 2005 at Lyceum #52 in Ryazan, a specialized school with advanced mathematics and physics programs. He then proceeded with obtaining his Bachelor's degree between 2006 to 2010 in Applied Mathematics and Physics from Moscow Institute of Physics and Technology, Russia. Upon completion of the Bachelor's program, he pursued a Master's degree in Applied Mathematics and Physics from the same university, with a specialization in Control theory, Cybernetics and Operations Research. He received the Master's degree in 2012. Later, in 2015 he obtained his second Master's degree in Econometrics and Management Science from Erasmus University Rotterdam, The Netherlands, with a specialization in Operations Research and Quantitative Logistics.

He started his PhD project in November 2015 within the Stochastics group at Centrum Wiskunde & Informatica in Amsterdam, The Netherlands, under the supervision of Rob van der Mei and Peter van de Ven. His research focused on studying operational problems arising within Emergency Response Networks, such as dispatching and proactive relocation of resources. The results of his PhD project are presented in this dissertation.