CWI
Centrum Wiskunde & Informatica

VU VRIJE UNIVERSITEIT AMSTERDAM

Master Thesis Business Analytics

Real-time dispatching and relocation of service engineers

Author: A. Pechina

Supervisors: prof. dr. R. D. van der Mei (VU)
dr. ir. P. M. van de Ven (CWI)

Master Thesis Business Analytics

Real-time dispatching and relocation of service engineers

Author: A. Pechina

Supervisors: prof. dr. R. D. van der Mei (VU)
dr. ir. P. M. van de Ven (CWI)

Centrum Wiskunde & Informatica
Science Park 123
1098 XG Amsterdam

Vrije Universiteit Amsterdam
Faculty of Science
Business Analytics
De Boelelaan 1081a
1081 HV Amsterdam

August 2018

Management Summary

Problem:

Given a service region consisting of several machines, repair base stations and service engineers. Costs occur if a machine breaks down and a repairman does not reach it within the established time window.

The goal is to minimize costs by finding a smart policy to manage the service engineers.

Approaches used:

- Compliance tables
- Heuristic approach
- Approximate Dynamic Programming
- Markov Decision Theory

Results:

- Accurate estimation of the repair gives significant advantage only in case of large repair times.
- Compliance tables perform well only for the systems with small distances.
- The heuristic policy increases the fraction of calls answered in time in up to 60% compared to a static policy and compliance tables.
- The ADP approach outperforms all other approaches.
- If the repair times are large, the exact optimal policy is obtained from Markov Decision theory.

Practical advice:

1. In case of limited computational resources, use heuristic approach.
2. If there is one given system and the time is not very limited, use ADP approach to obtain high-performing policy.
3. In case of large repair times use the policy described in Section 9.2.

Contents

1 Introduction

Buying expensive products (such as complex machines or software) a company wants assurance that the product will have a long lifetime with a low downtime percentage. One of the ways a manufacturer can assure this is to provide post-sale support of the product. Post-sale support includes, for example, installation, warranties, provision of spare parts and maintenance service contracts. In the modern world with demanding customers and new products released to the market every day, providing good post-sale support is an important competitive advantage for manufacturers [27].

As it is impossible to guarantee no-failure operation of a machine, corrective maintenance is an essential part of post-sale support. The aim of corrective maintenance is to restore the operation of a failed product in the shortest possible time. The challenge for the manufacturer is that the corrective maintenance is performed unpredictably because the machine's failure time is not known a priori. The time between the failure is detected and a service engineer arrives at the location of the failed machine is a good measure of the quality of corrective maintenance provided by a manufacturer. In practice, the service level (i.e. the percentage of failures that should be fixed within particular time window) is usually fixed in the repair contract between a manufacturer and a buyer. If this service level is not met, the manufacturer pays a penalty to the buyer. This also affects the reputation of the manufacturer.

To meet these service level agreements, the manufacturer should have a dispersed network of service facilities such as spare part warehouses (if relevant) and engineers for field service, which allows a quick response when a failure is reported. Construction and control of this network requires finding a balance between customer satisfaction and low operational costs, which is not always easy [26]. Customer satisfaction requires low response times for new failures. This can only be ensured by having a service facility close to each customer with sufficient spare parts stock and a ready-to-go service engineer. But maintaining a lot of service facilities is costly for a manufacturer. So the limited operational costs lead to a network of a limited number of service facilities that should be wisely operated to provide good coverage of the customers.

Problem

The goal of this thesis is to find a policy to manage field service engineers that helps manufacturers to meet the agreements with customers. We ignore spare parts management as, first, it is well studied in the literature (see [16] for the overview of this area) and, second, does not necessarily apply to all application domains (consider, for example, software systems support). Moreover, real-time management of the service engineers is a difficult problem in itself, which received little attention in the research literature.

The global service network of a manufacturer is typically divided into several service regions with several customer locations operating one or more machines in each of them. These service regions are operated independently by different management centers. Apart from the machines, a service region consists of several repairmen and repair base stations. Base stations are places, where the repairmen spend their free time and prepare for future work.

There are three questions to be answered to design a service network for a given service region:

1. *(strategic) Where should the base stations be located?*
2. *(tactical) How many repairmen are needed?*
3. *(operational) How to manage the repairmen to achieve the best performance?*

Our study is focused on the last question, so we assume that the number of repairmen in the region and the locations of the base stations are given and they are not allowed to change.

When one of the machines breaks down, the management center receives a call from the customer. Then the manager decides how to react to the call: he can either dispatch an idle service engineer to repair the machine, or wait until one of the busy engineers finishes his job and then

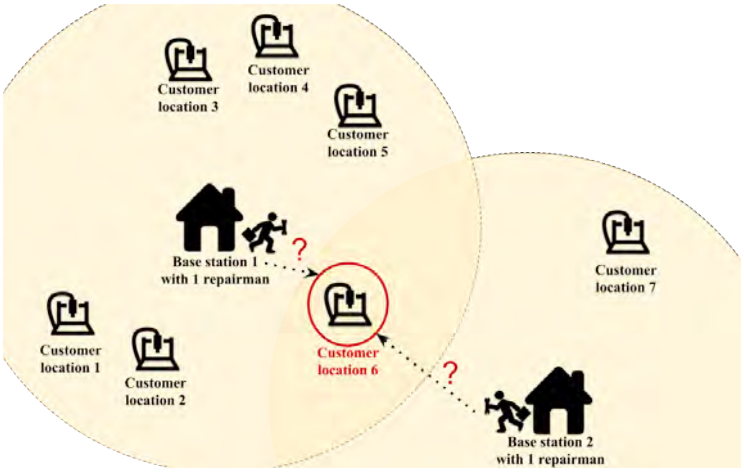

Figure 1.1: Example of a decision to make. A call arrives from customer location 6. Orange circles show the places that can be reached by each repairman within the time limit set in the service agreement.

dispatch this engineer. After the repair the service the manager sends the engineer, that just finished, either to one of the base stations or to one of the other customers with broken machines.

When a repairman is dispatched to repair a broken machine, other customers, that he was responsible for, receive less coverage. This can potentially cause the situation when a call arrives from one of them and there is no repairman to reach the place in time, so a penalty has to be paid. To avoid this situation, it can be beneficial to reposition idle repairmen, so that they provide better coverage of the region.

Our goal is to find the optimal policy to manage the service engineers, where optimal means that this policy minimizes the long-term costs caused by violations of the service level agreement. This is a complicated problem, as making a decision requires finding a good balance between the possible costs from failures that were already reported and future failures. Dispatching and relocation decisions increase the coverage in one part of the region and decreases the coverage of another part.

Consider for example a situation depicted in Figure 1.1, where the call arrives from a customer location 6 and the manager has to decide which repairman (from base station 1 or base station 2) to dispatch. The first repairman is closer to the customer, so he can arrive earlier. But if he is dispatched customer locations 1-5 stay uncovered. If a failure occurs in one of these locations, it can not be fixed in time.

Approach

A policy can be divided into two parts: relocation policy and dispatching policy.

- The relocation policy prescribes the location of the idle service engineers according to the system state. It answers two questions:
  - *Should the idle repairmen be relocated from their current locations to new base stations?*
  - *To which base station should a repairman that just finished a repair be sent?*
- The dispatching policy is responsible for managing received calls, so it answers the following two questions:
  - *When should a repairman be dispatched immediately when a failure is reported or should we wait for some repairmen to finish their jobs?*
  - *Which repairman should be dispatched to the customer?*

Depending on the approach, relocation and dispatching decisions can either be optimized separately, or as a combined policy.

The traditional technique to find optimal policies for such problems is Markov Decision theory. However, the system that we consider is so complicated that these results can not be used in practice due to the high computational complexity and the high memory usage.

For systems where the optimal policy can not be obtained, the goal is to find a policy performing as close to the optimal as possible. There are different approaches to find such policies. We focus our study on heuristics and approximate dynamic programming. The former includes finding separate heuristics for dispatching and relocation that, when used together, show a high performance. The Approximate Dynamic Programming (ADP) approach improves the dispatching and relocation jointly. It approximates the methods of Markov Decision theory in a way that reduces computational times.

Contribution

The contribution of this thesis is as follows:

1. We discuss several scalable heuristics both for relocation and dispatching decisions and compare their performance by means of simulation. This comparison is done for systems with different properties, such as geography of the region and workload, to observe how the performance of each policy depends on the system properties. For small instances the performance of these heuristic policies is also compared to the performance of the optimal policy obtained from Markov Decision theory results.

2. We propose an ADP approach for this problem. This approach helps to overcome the curse of dimensionality for the systems of realistic size and outputs a policy close to optimal.

3. We obtain theoretical results for the systems with heavy and light traffic. Heavy load regimes are associated with either long average service time or high frequency of failures, while light load regimes mean low service time or rare failures. For some regimes these results give the explicit optimal policy and for other regimes a scalable algorithm to numerically compute this policy.

Thesis outline

The rest of the thesis is organized as follows. Section 2 gives an overview of related literature. In Section 3, we describe the model of the system according to Markov process theory. Assumptions and decisions made to construct this model are also discussed in this section. Section 4 introduces an approximation to expected covered demand, an important objective function used to construct policies in next sections.

Different dispatching and relocation policies are discussed in Sections 5 and 6, respectively. These sections also contain the simulation results used to compare the performance of the policies considered. The approximate dynamic programming approach is discussed in Section 7.

In Section 8, we consider a discrete-time version of the considered process. This model gives the opportunity to apply the policy iteration algorithm to find the optimal policy for small instances. The performance of the obtained policy is compared to the performance of the best policies from Sections 5 and 6.

In Section 9, we consider the process under several extreme regimes (such as large repair times or large arrival rate of the calls) and find the optimal policies under these regimes analytically.

Finally, Section 10 contains conclusions and discussion.

2 Related work

To our knowledge, there are just a few works on real-time service engineer management (see, for example, [7] or [20]), but they consider different settings of the problem, for example ignore geographical locations of the engineers and focus only on the assignment problem. However, there is a closely related well-studied field of the emergency service management, and ambulance management specifically. In this section we briefly outline the results from the ambulance management that are relevant to our study. For an extensive overview of recent optimization models in location, relocation and dispatching of ambulances, we refer to Bélanger et al. [6].

We organize our literature review according to the methods used in the discussed studies. First, we discuss the approach when the problem is formulated as an integer linear problem. Then the results obtained from Markov Decision theory, several heuristics, that were successfully used as relocation and dispatching policies. Finally, the approximate dynamic programming approach to the problem.

2.1 ILP-based approaches

The first work in this field was devoted not to operational decisions, but to facility location problems. The results of this work play a key role in future research, so we start our review with several studies about facility location problems and continue with operational decisions research.

The aim of the earliest work was to find a good position of the available servers in order to maximize the number of calls answered in time or to minimize the average response time, i.e. the time between the arrival of a call and the arrival of a server to the call location.

For instance, the maximal coverage problem, first described by Church and ReVelle [8], finds the allocation of the servers that maximizes the number of demand locations that can be reached by at least one server within the time threshold. The problem is formulated as an integer linear programming problem (ILP) and can be solved even for large systems.

Later, Daskin [9, 10] improved this model by taking into account the busy time of the servers and constructed a new model called MEXCLP (maximal expected coverage location problem). First the busy fraction of each server was assumed known from practice, but later the correction factor derived by Larson [22] was used to estimate the probabilities and the objective function of this method.

There are also other approaches to the problem such as Double Standard Model [13], Maximum Availability Location Problem [30] and others. The survey by Li et al. [24] gives a good overview of these methods as well as of the possible heuristics to reduce computational time for big systems.

In all above mentioned policies when one server is going to answer the call, other servers stay at the same locations, which can result in a coverage gap. Relocation policies such as compliance tables deal with this problem. A compliance table is a policy where the location of the servers depends on the number of available servers. Every time this number changes (i.e. when a call arrives and when a service is finished) idle servers are repositioned respectively. In 2006, Gendreau et al. [15] introduced a method called MECRP (maximum expected coverage relocation problem) to compute compliance tables where for each number of available servers the coverage is maximized. It was later extended by Van Barneveld et al. [4] to incorporate busy time of the servers.

The problem of choosing a server that is best to dispatch to the accident can also be formulated as an ILP problem. According to the computational study of Jagtenberg et al., it can even outperform other dispatching policies that use more information about the state of the system [19].

2.2 Markov decision theory approach

One of the approaches to find the exact optimal policy is to model the system as a Markov decision process with either continuous or discrete time. For small systems, with just a few servers the optimal policy can be found by policy or value iteration or by exact dynamic programming [17, 32]. But for realistic-sized systems the state space is too large and the problem is intractable from the

computational perspective. One way to address this problem was considered by Jagtenberg et al. [17]. Instead of finding the exact optimal policy they performed only several steps of value iteration algorithm and compared the results for different numbers of steps with other policies.

Katehakis and Levine [21] formulated an MDP for a server assignment problem, where costs occur every time a job is assigned to a server and they depend on a pair of a job and a server. They considered the system under heavy and light traffic regimes, meaning large service time and small arrival rate respectively. In case of light traffic, they developed an efficient algorithm to compute optimal assignment policy. For a system under heavy traffic, they proved that the policy always choosing a server with minimal costs is the optimal policy.

2.3 Heuristic solutions

Another approach is not to calculate the decision for each situation in advance, but to make decisions in real time. This can be applied to both relocation and dispatching.

Gendreau et al. [14] were the first to propose a real-time server relocation model [6]. It is based on the Double Standard Model, proposed earlier by the same authors, 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 by Andersson and Värbrand [1]. They also proposed a method to find a relocation scheme minimizing the travel times.

In 2015, Jagtenberg et al. [18] proposed a heuristic for redeployment of the servers that just finished the service. The heuristic is based on calculating the expected covered demand and choosing the new location of the server according to the result. This work was continued in later paper by Van Barneveld et al. [3]. They allowed relocation not only after the service completion but also right after the dispatching of a server. They also studied how different restrictions, such as the restriction on the maximum distance of relocation, influence the performance of the system. Two types of regions, a rural and an urban one, 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 the dispatching decisions. Gendreau et al. [14] proposed to choose among all servers, that can reach the incident in time, the one that leads to the minimal relocation time. Anderson et al. [1] proposed to dispatch the ambulance that causes the smallest decrease in preparedness. Jagtenberg et al. used expected covered demand instead of unpreparedness and showed that their dispatching policy outperforms the commonly used closest-first dispatching policy [19]. Also the possibility of waiting for a busy server to finish the service, instead of dispatching an idle repairman, was shown to reduce the costs [2].

2.4 Approximate Dynamic Programming

In 2010, Maxwell et al. [25] used an ADP approach combined with approximate policy iteration for the ambulance management problem. They considered the problem of optimal redeployment of the ambulances after completion of the service for a system with no other types of relocation and fixed closest-first dispatching policy and approximated the value function by a linear combination of basis functions such as the number of uncovered demand locations and etc. An iterative procedure was used to tune the parameters of the approximation. Later, Schmid [29] used the same framework to optimize the dispatching policy and the redeployment of an ambulance after a service.

Finally, Nasrollahzadeh et al. [28] considered the general problem of dispatching and relocation of the ambulances with possibility to reposition the idle ambulances and to put an incoming call into the queue instead of immediate dispatching of a repairman. Their approximation is based on five basis functions that do not only characterize the current state, but also the possible next states. The result was compared to the best practices from the literature and outperformed them for the considered system.

Our study differs from the previous work in two ways. First, we consider the model where once a machine is broken, it can not be an origin of new incidents, so the demand produced from

demand nodes is not unlimited. According to our knowledge there are no previous works on similar systems. Limited population was studied in queuing theory [12], but not in the service network design.

The second difference is that we do not only look for the optimal policy for one given system, but consider several systems with different parameters (such as the average service duration, the average distance and topology of the region) and study how the relations between these parameters affect the performance of different policies.

3 Model description

In this section we model the considered system as a Markov decision process. It is an event-based process with continuous state space. We also discuss the assumptions made and several parameters of the system that affect the performance of different policies.

3.1 Service region

The service region is represented by the set of the demand nodes $\mathcal{K} = \{1, \ldots, K\}$ and the set of the base stations $\mathcal{R} = \{1, \ldots, R\}$. It is possible to travel from any location to any location, the traveling distances are known and deterministic. There is a capital good installed in each demand node. Several machines are installed at the same place are modeled as different demand nodes with the same location.

There are M repairmen in the system. Each repairman can be in one of three states: either resting at a base station, or doing a repair in a demand node, or traveling. It is forbidden for a repairman to stay idle somewhere except for the base stations. The number of repairmen that stay at the same station at the same time is not limited.

The time limit within which a repairman has to reach a broken machine to avoid paying a penalty is given. We consider only the service regions where each demand node can be reached within this time limit from at least one base station.

The distribution of the working time of each machine and the distribution of the service time are known. In practice, these distributions can be obtained from historical data.

3.2 Assumptions and restrictions

There are several assumptions that we made to model the process:

1. The time that a machine works after repair is exponentially distributed with rate λ. This parameter is the same for all machines.
2. The time it takes to repair a machine is exponentially distributed with rate μ. This parameter is the same for all machines.
3. Traveling times are deterministic.
4. There is no preemption. A repairman can not stop a repair process and travel somewhere else. If a repairmen is traveling to some location, he has to reach this location before being relocated somewhere else.

Restrictions determine the set of possible actions. There are restrictions on dispatching decisions and on relocation decisions.

When a call arrives, it can either be immediately answered by one of the idle repairmen or can be put in the queue. There are only two types of moments when a repairman can be dispatched to a call from the queue. First, a repairman that just finished a repair in some demand node can be redeployed to another demand node. Second, when a traveling idle repairman arrives at his destination, he can be dispatched to answer a call from the queue.

The relocation can only be performed at two types of moments: either when repairman is dispatched to answer a call, or when a repairman finished a repair of one of machines. If a new call is put in the queue then relocation is not allowed. In the case when a new call is first put in the queue and then one of the traveling repairmen arrived at his destination and is dispatched to this call, the relocation is again not allowed. Only one repairman can be relocated at a time.

3.3 State space

Each capital good can be in one of three states: *working*, *being in repair* or *waiting for a repairman to come*. As there are K capital goods their state is described by a vector

$$\kappa = (\kappa_1, \ldots, \kappa_K),$$

where

$$\kappa_k = \begin{cases} 0, & \text{if machine } k \text{ is working;} \\ -1, & \text{if machine } k \text{ is in repair;} \\ t, & \text{if machine } k \text{ is waiting for a repairman for time } t. \end{cases}$$

For each repairman m, $m = 1, \ldots, M$, his location $\mathfrak{m}_m$ is described by his destination l_m (a demand node or a base station) and the distance d_m left to this location:

$$\mathfrak{m}_m = (l_m, d_m), \qquad m = 1, \ldots, M.$$

If the repairman is not traveling but he is either resting at a base station or doing a repair at a demand node, his destination is equal to his current location and the distance is equal to 0. So the state of all repairmen is described by the vector

$$\mathfrak{m} = (\mathfrak{m}_1, \ldots, \mathfrak{m}_M).$$

Note that the destination and the remaining distance does not describe the explicit location of the repairman, but since he can not change the destination until he reaches it, the former is sufficient to describe the dynamics of the system.

The state of the system is the combination of the state of the repairmen $\mathfrak{m}$ and the state of the capital goods κ.

To describe the process we consider only the moments, when one of the following events happens:

- a call arrives from a demand node;
- a repair in a demand node is finished;
- a repairman arrives at a demand node;
- a repairman arrives at a base station.

At all other moments of time there is no uncertainty and no actions can be taken.

It is essential to know the type e of the event that happened and the time t, when it happened. Therefore, in each considered moment the state of the process is described by the tuple

$$s = (t, e, \mathfrak{m}, \kappa).$$

We denote by $t(s)$, $e(s)$, $l_m(s)$, $d_m(s)$ and $\kappa_k(s)$ the corresponding components of this tuple.

The set S of all possible states of the process is infinite. Note also that as the time is included in the state s the process never visits the same state twice.

3.4 Action space

The set of possible actions depends on the state s of the system and is mostly determined by the type of event e. We consider each of four types of the possible events described in section 3.3 and describe the set of possible actions for it.

Type 1. A call arrives from demand node k.

In this case the action consists of a dispatching decision and a relocation decision. Denote by $\mathcal{F}(s)$ the set of all idle repairmen, $\mathcal{F}(s) = \{m \in 1, \ldots, M \mid l_m(s) \in \mathcal{R}\}$. Then either one of the repairmen from $\mathcal{F}(s)$ can be dispatched, or the call can be placed in queue. Define $X_m = 1$ if a repairman $m \in \mathcal{F}(s)$ is assigned to the call, and $X_m = 0$ otherwise.

We only allow to relocate one repairman at a time, so the relocation action is described by the repairman that is relocated, and the base station that he is relocated to. Define $Y_{mr} = 1$ if the repairman $m \in \mathcal{F}(s)$ is relocated to the base station r, and $Y_{mr} = 0$ otherwise.

Then an action is described by a pair of a dispatching vector X and a relocation matrix Y. The action space in state s is given by

$$\mathcal{A}_1(s) = \Big\{ (X,Y) \mid \sum_{m\in\mathcal{F}(s)} X_m \le 1; \quad \sum_{\substack{m\in\mathcal{F}(s),\\ r\in\mathcal{R}}} Y_{mr} \le 1; \\ \sum_{\substack{m\in\mathcal{F}(s),\\ r\in\mathcal{R}}} X_m Y_{mr} = 0; \ \Big(1 - \sum_{m\in\mathcal{F}(s)} X_m\Big) \sum_{\substack{m\in\mathcal{F}(s),\\ r\in\mathcal{R}}} Y_{mr} = 0 \Big\},$$

where the constraints ensure that not more than one repairman is dispatched, not more than one repairman is relocated, the relocated repairman differs from the dispatched repairman and the relocation is forbidden if the call is placed in queue.

Type 2. A repair in demand node k by repairman m is finished.

In this case the action consists of two actions: redeployment of the repairman m and relocation. The repairman can be sent either to rest in one of the base stations or to do a repair in one of the demand nodes from the queue. Denote the set of all demand nodes waiting in the queue in state s by $Q(s)$. Define $X_l = 1$, if the repairman that just became idle is dispatched to a location $l \in Q(s) \cup \mathcal{R}$, and $X_l = 0$ otherwise. To describe the relocation decision define $Y_{nr} = 1$, if the repairman $m \in \mathcal{F}(\mathcal{S})$ is relocated to the base station r, and $Y_{nr} = 0$ otherwise.

Then an action is described by a pair of a redeployment vector X and a relocation matrix Y and the set of all possible actions in state s is given by

$$\mathcal{A}_2(s) = \Big\{ (X,Y) \mid \sum_{l\in Q(s)\cup\mathcal{R}} X_l = 1; \sum_{\substack{n\in\mathcal{F}(s),\\ r\in\mathcal{R}}} Y_{nr} \le 1 \Big\},$$

where the constraints ensures that the repairman is redeployed to only one location and at most one repairman is relocated.

Type 3. Repairman m arrives at base station r.

When a repairman m arrives at a base station he can either be left to rest at this station or be dispatched to one the demand nodes from the queue. If we denote again the set of all demand nodes in the queue in state s by $Q(s)$ and define $X_k = 1$, if repairman m is dispatched to demand node k, and $X_k = 0$ otherwise, then an action is described by vector X and the set of all possible actions in state s is

$$\mathcal{A}_3(s) = \Big\{ X \mid \sum_{k\in Q(s)} X_k \le 1 \Big\}.$$

Note that if the queue is empty, then there are no possible actions for this type of states and the repairman is always left at the same base stations.

Type 4. Repairman m arrives at demand node k.

Then the repairman starts the repair process and there are no available actions:

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

3.5 Transitions

The evolution of the process from state s_n can be characterized by action a_n, random element $\omega(s_k, a_k)$ and function Φ as

$$s_{n+1} = \Phi\left(s_n, a_n, \omega(s_n, a_n)\right).$$

The random element determines the event of state s_{n+1}. Denote by $d(s_n, a_n)$ the minimum of all non-zero distances remaining for the repairmen to travel after taking action a_n in state s_n. If no events of first two types (arrival of a call or end of repair) occurs then in state s_{n+1} time is equal to $t(s_{n+1}) = t(s_n) + d(s_n, a_n)$ and the event is the arrival of a repairman with the shortest remaining distance at his destinations. If there are no traveling repairmen in the system after taking action a_n in state s_n then set $d(s_n, a_n) = \infty$.

Denote the set of all working machines after taking action a_n in state s_n by $\mathcal{W}(s_n)$ and the state of all machines in repair by $\mathcal{H}(s_n)$. The time from the state s_n till the first break of each machine from $\mathcal{W}(s_n)$ is exponentially distributed with rate λ. The time from state s_n till the end of repair in each of demand nodes from $\mathcal{H}(s_n)$ is exponentially distributed with rate μ. So the time from state s_n to the closest event of the first two types is exponentially distributed with rate

$$\eta(s_n) = \lambda W(s_n) + \mu H(s_n),$$

where $W(s_n) = |\mathcal{W}(s_n)|$ and $H(s_n) = |\mathcal{H}(s_n)|$, as the minimum of several exponential distributions. If this time is less than $d(s_n, a_n)$ then the next event is either a call arrival or an end of repair.

So, if in state s_n action a_n is taken, the probability that the next event is

- the arrival of a call from demand node $k \in \mathcal{W}(s_n)$ is

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

  if $\mathcal{W}(s_n) \neq \emptyset$, and 0 otherwise;

- the end of repair in demand node $k \in \mathcal{H}(s_n)$ is

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

  if $\mathcal{H}(s_n) \neq \emptyset$, and 0 otherwise;

- the arrival of a repairman at his destination is

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

  if there are any traveling repairmen, and 0 otherwise.

The time until the next event is distributed as the minimum of exponentially distributed with rate $\eta(s_n)$ random variable and $d(s_n, a_n)$.

When the next event and the time until this event are known, the function Φ gives the location of all repairmen and the state of all capital goods. The distances remaining to travel are decreased by this time and the waiting time of the capital goods are increased by this time. If action a_n includes repositioning of some of the repairmen then their destinations and remaining distances change according to the action. If the event in state s_{n+1} is the end of repair at (arrival of a repairman at) demand node k then the state of this demand node is changed from -1 ($\kappa_k(s_n)$) to 0 (-1).

3.6 Costs

If a call arrives from demand node k and a repairman does not reach this demand node in time TL, then a large penalty is paid. In this case also a small penalty is paid per every time unit until a repairman arrives at this demand node. Our goal is to minimize the long run average penalty for a given system. All travel costs and other operational costs are neglected. We assume the large penalty for a late arrival equal to 1 and the small penalty equal to ϵ.

Denote by $c(s_n, a_n, s_{n+1})$ the costs that are charged during the transition from state s_n to state s_{n+1} when action a_n is taken. The costs for being late are equal to the number of machines which

waiting time exceeded time limit in the time period $(t_n, t_{n+1}]$. The small costs for each machine is the time in this time period that this machine stayed broken. Then in total

$$c(s_n, a_n, s_{n+1}) = \sum_{k=1,\dots,K} \mathbb{I}\{\kappa_k(s_{n+1}) \geq \text{TL}\}\,\mathbb{I}\{\kappa_k(s_n) < \text{TL}\} + \\ + \epsilon \sum_{k=1,\dots,K} \mathbb{I}\{\kappa_k(s_{n+1}) \geq \text{TL}\} \min\left(t_{n+1} - t_n, \kappa_k(s_{n+1}) - TL\right),$$

where $\kappa_k(s_n)$ is the state of demand node k in state s_n.

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

3.7 Parameters of the system

In this section we discuss the important parameters of the system and how they affect the optimal policy. Table 3.1 contains all numeric parameters of the system. Another parameter of the system is the map density. Figure 3.1 gives an example of maps of three different densities, from low to high.

The relations between these parameters define the properties of the system. The same general type of policies may be optimal for one system and show low performance for the others. The most important relations are the map density, the relation between $\hat{\mu}$ and λ and the relation between $\hat{\mu}$, and μ.

Map density

The map density represents the relation between the average distance between nodes in the service region and the time limit for answering for a call. The more dense the map is the smaller this ratio is. It can affect the optimal policy in several ways.

If the map is sparse, then the distances between the base stations are large and relocation between them can take undesirably long time. Each demand node is on average covered by only one base station so it is important that at least one repairman is present there.

In dense maps the distances between the demand nodes are rather small, so it can be the case that it is better to wait for some repairman to finish a repair in a demand node nearby instead of dispatching an idle repairman (see section 5.2 for more details).

Relation between $\hat{\mu}$ and λ

If the map and the number of repairmen are fixed, this relation influences the load of the system. If μ is increased with fixed λ the system becomes more loaded and this leads to decrease in number of calls answered in time.

K	number of demand nodes
R	number of repairmen base stations
M	number of repairmen
$1/\lambda$	average time a capital good works after repair
$1/\mu$	average time it takes to repair a capital good
TL	time limit after which the penalty should be paid
$1/\hat{\mu}$	average time it takes to answer a call (includes traveling time and repairing time)

Table 3.1: Parameters of the system

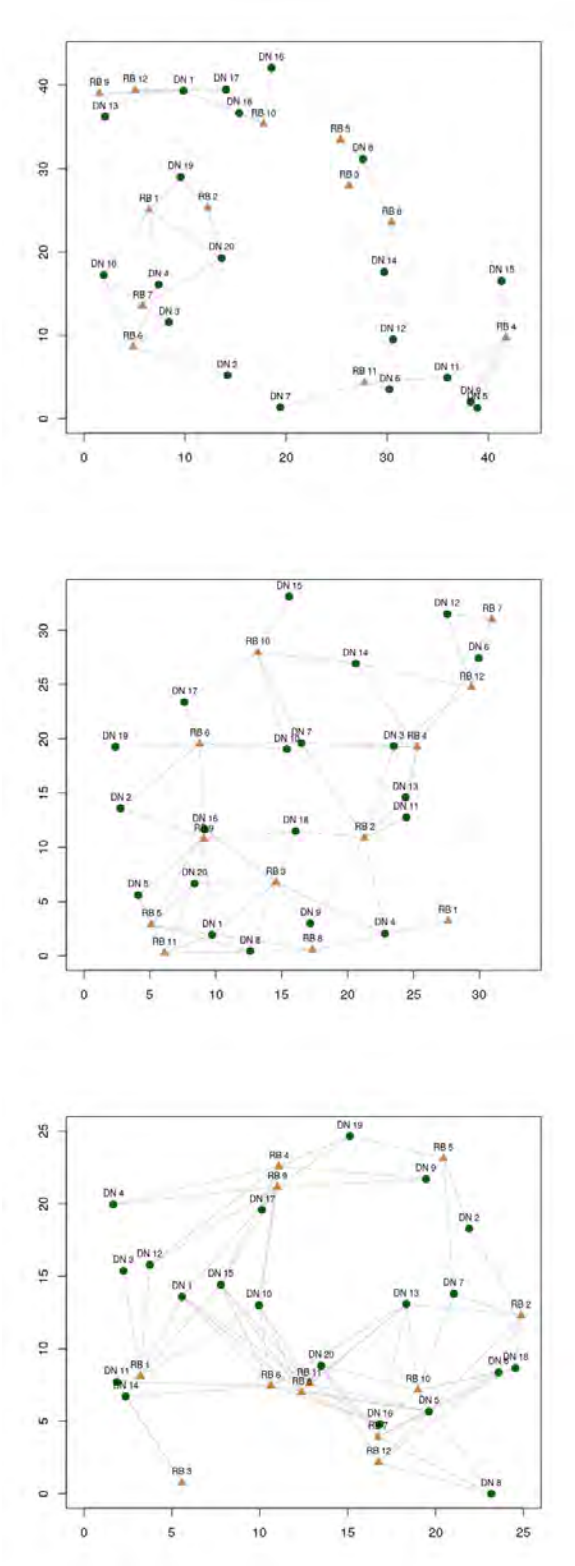

Figure 3.1: Example of maps of density 0.3, 1 and 2 and the same time limit 10. Edges represent distances that can be reached in the time limit.

This relation plays an impotant role in steady-state distribution of the approximating process discussed in Section 4.1.

Relation between μ and $\hat{\mu}$

When a repairman is busy, he is either doing a repair in a demand node, or traveling to a demand node. This relation shows which activity takes more time on average.

If μ is very close to $\hat{\mu}$ it means that the traveling distances are very small compared to the time it takes to repair a capital good. In extreme cases we can ignore the traveling times and assume that all relocations are done immediately.

If μ is much larger than $\hat{\mu}$, then the traveling distances are significant and the policy optimal for small distances can be not optimal any more. For example, relocation of the idle repairmen between the base stations can decrease the performance of the system (see Section 6.4 for further details).

4 Expected covered demand

Given the locations of the repairmen, the expected covered demand estimates the long-term fraction of calls that will be answered in time. This is done under the assumption that the repairmen return to the same location after completion of the service, but this estimation is also used for other policies.

Expected covered demand plays an important role in several well-performing heuristic relocation and dispatching policies from the ambulance domain [5, 18, 19]. We use this metric for one of dispatching and several relocation policies (see Sections 5 and 6). It is also used in Section 7 as one the basis functions for Approximate Dynamic Programming approach.

In this section, we introduce an approximation to the expected covered demand for our type of systems and construct the allocation of repairmen that maximizes it. In addition, the behaviour of the approximating process described in Section 4.1 gives an idea of the behaviour of the original process.

4.1 Approximating process

In this section we construct the process $C = \{C_n,\ n = 1, 2, \dots\}$, that approximates the number of broken capital goods in the n^{th} state of the original process s_n, $n = 1, 2, \dots$, so

$$C_n = |\,\{k \in \mathcal{K} \mid \kappa_k(s) \neq 0 \text{ or } e = \text{"a call arrived fro demand node } k\text{"}\}\,|$$

We compute the steady-state distribution of this process and observe its dependence of the system parameters.

The time that a machine stays working is exponentially distributed with rate λ. To make C_n a Markov process, we assume that the time a machine stays broken is also exponentially distributed with rate $\hat{\mu}$. This time includes the traveling time of a repairman and the duration of the repair. If the call was put in queue because there were no available repairmen, the waiting time in the queue is not included.

The state space of the process C is $\{0, 1, \dots, K\}$, so it is a finite-state process. From state C_n with k broken capital goods there are three possible transitions:

- The event in state s_{n+1} is of type *"a call arrives"*. 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 $\hat{\mu} \cdot \#$ capital goods in repair $= \hat{\mu} \cdot \min\{k, M\}$. (This transition is not possible if $k = 0$.)
- The event in state s_{n+1} is of type *"a repairman arrives at his destination"*. Then $C_{n+1} = k$.

If we omit the third type of transitions, then the process C still illustrates the dynamics of the number of the broken machines, but it is a finite-state birth-death process and we can calculate its steady-state distribution.

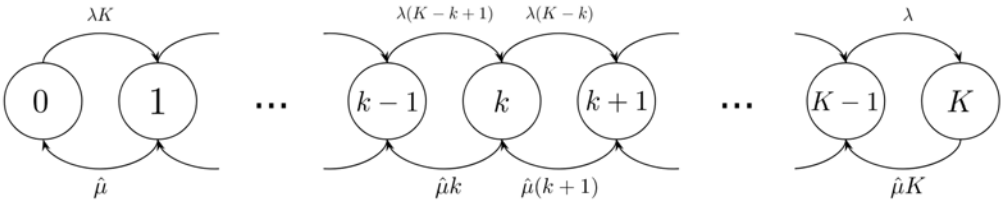

Figure 4.1: State diagram of the process C.

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

$$\begin{cases} \lambda K P(0) = \hat{\mu} P(1), \\ (\lambda(K-k) + \hat{\mu}k)\, P(k) = \lambda(K-k+1)P(k-1) + \hat{\mu}(k+1)P(k+1), & k = 2, \dots, M-1, \\ (\lambda(K-k) + \hat{\mu}M)\, P(k) = \lambda(K-k+1)P(k-1) + \hat{\mu}MP(k+1), & k = M, \dots, K-1, \\ \hat{\mu}MP(K) = \lambda P(K-1). \end{cases} \tag{4.1.1}$$

One can check that

$$P(k) = \begin{cases} \binom{K}{k} \left(\frac{\lambda}{\hat{\mu}}\right)^k P(0), & k = 0, \dots, M-1, \\ \frac{k!}{M!M^{k-M}} \binom{K}{k} \left(\frac{\lambda}{\hat{\mu}}\right)^k P(0), & k = M, \dots, K \end{cases} \tag{4.1.2}$$

is the solution of the balance equations.

Adding the normalization equation $\sum_{k=0}^{K} P(k) = 1$ to the system we get

$$P(0) = \left[\sum_{k=0}^{M-1} \binom{K}{k} \left(\frac{\lambda}{\hat{\mu}}\right)^k + \sum_{k=M}^{K} \frac{k!}{M!M^{k-M}} \binom{K}{k} \left(\frac{\lambda}{\hat{\mu}}\right)^k \right]^{-1}. \tag{4.1.3}$$

Using these formulas one can calculate $P(k)$ for $k = 0, \dots K$. Example 4.1.1 gives the idea of the behaviour of $P(k)$ for different relations between λ and $\hat{\mu}$ and between K and M.

Example 4.1.1. In this example we calculate steady-state distribution for one system and explore how it depends on the number of repairmen and the average working time. The number of capital goods $K = 20$, so there are 21 states, $\hat{\mu}$ is fixed to 1 and λ is fixed to 0.8.

The probability of being in each state for the number of repairmen $M = 3, 8, 11, 15$ is given in Figure 4.2. One can see that the decrease in number of repairmen shifts the distribution to the right. This happens because with less repairmen capital goods have to wait longer to be repaired. But after some value, increasing number of repairmen does not change the distribution a lot (compare $M = 11$ and $M = 15$).

Next, we fix $M = 6$ and vary the parameter λ. Note that we do not have to vary $\hat{\mu}$ after that as $\{P(k), k = 0, \dots, K-1\}$ depends only on ratio $\lambda/\hat{\mu}$. The steady state distributions for $\lambda = 0.2, 0.4, 0.6$ and 0.8 can be found in Figure 4.3. We see again that with increase of λ the load of the system increases and the distribution shifts to the right.

In the next sections we use the probability that there are m busy repairmen to construct compliance tables. If S_m is the event of having m busy repairmen in the system, then

$$\mathbb{P}(S_m) = \begin{cases} P(m), & m = 1, \dots, M-1, \\ \sum_{k=M}^{K} P(k), & m = M. \end{cases} \tag{4.1.4}$$

Finally, one of the interesting parameters of the system is the *load*. We define the load ρ as the average fraction of time that one repairman is busy. Given the steady state distribution, the load can be computed as total busy time of all repairmen divided by the number of repairmen:

$$\rho = \frac{1}{M} \sum_{m=1}^{M} m\mathbb{P}\left(S_m\right). \tag{4.1.5}$$

One can see that from obtained expressions for $P(k)$ the load is increasing in λ and K and decreasing in $\hat{\mu}$ and M, which coincides with common sense.

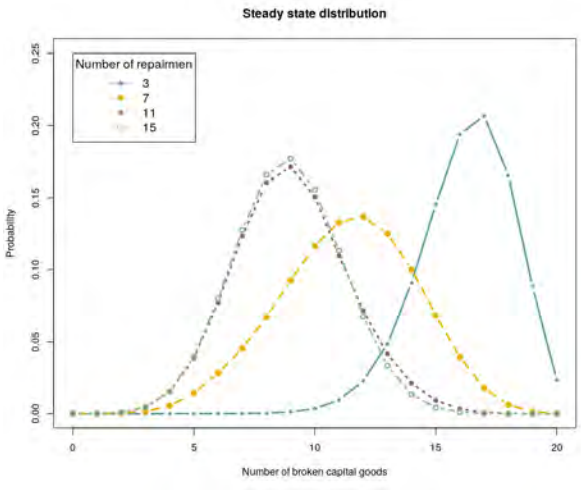

Figure 4.2: The steady state distribution for $K = 20$, $\hat{\mu} = 1$, $\lambda = 0.8$ and various M.

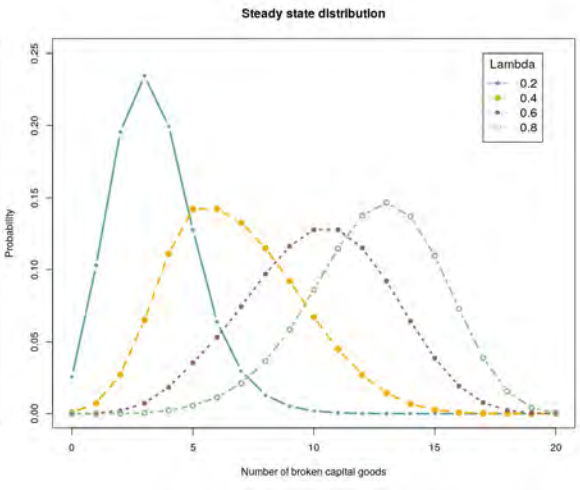

Figure 4.3: The steady state distribution for $K = 20$, $\hat{\mu} = 1$, $M = 6$ and various λ.

Example 4.1.2. We fix again $K = 20$ and $\hat{\mu} = 1$ and observe the load for different values of parameters λ (0.2, 0.8 and 1.2) and M (from 1 to 15). The results can be found in Figure 4.4. Note that for 1 and 2 repairmen in the system the busy fraction equals (or is very close) to 1 for all considered values of λ. And then from a certain number of repairmen the load decreases. This number of repairmen increases on λ.

4.2 Expected covered demand approximation

Let us consider a system where relocation is not allowed. It means that each repairman is assigned to one base station and he returns to this base station every time he becomes idle. Suppose that the system is in state s, when all repairmen are at base stations.

Assume that according to the chosen dispatching policy if a call arrives from demand node k, we first send repairman $m_1^{(k)}$, then, if he is busy, repairman $m_2^{(k)}$, and etc. Assume also that first we try to send the repairmen that can reach the demand node from their base stations and then, if all of them are busy, the repairmen that can not arrive on time. An example of such a dispatching policy is the closest-first dispatching policy that always dispatches a repairman from the closest base station.

The call is answered in time if the dispatched repairman $m_i^{(k)}$ can reach it in time.

To compute the probability that repairman $m_i^{(k)}$ is dispatched (so all repairmen $m_1^{(k)}, \dots, m_{i-1}^{(k)}$ are busy), we follow the procedure introduced by Larson [22, 23] and apply it to the birth-death process introduced in Section 4.1. If B_i is the event that repairman $m_i^{(k)}$ is busy, F_i is the event that he is idle and S_m is the event that there are m busy repairmen in the system, then

$$\begin{aligned}\mathbb{P}\left(B_1 \dots B_{i-1}F_i\right) &= \sum_{m=i}^{M} \mathbb{P}\left(B_1 \dots B_{i-1}F_i \middle| S_m\right)\mathbb{P}(S_m) \\ &= \sum_{m=i}^{M} \mathbb{P}(S_m)\mathbb{P}\left(F_i \middle| S_m B_1 \dots B_{i-1}\right)\mathbb{P}\left(B_{i-1} \middle| S_m B_1 \dots B_{i-2}\right) \dots \mathbb{P}\left(B_1 | S_m\right). \quad (4.2.1)\end{aligned}$$

The probability $P(S_m)$ that m repairmen are busy was computed in Section 4.1. Other terms can be approximated assuming that all repairmen have the same load and are independent of each

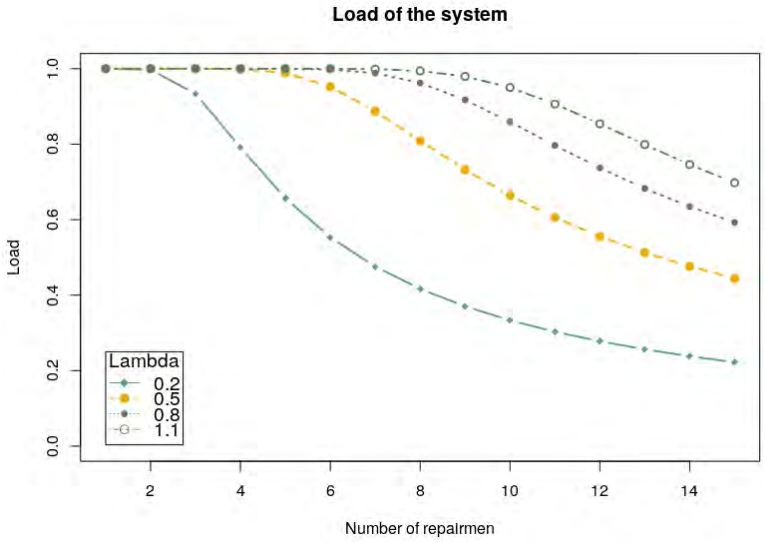

Figure 4.4: The load of the system with $K = 20$ and $\hat{\mu} = 1$ and various parameters λ and M.

other. Under this assumption,

$$\begin{aligned} \mathbb{P}\left(B_1|S_m\right) &= \frac{m}{M} \\ \mathbb{P}\left(B_2|S_m B_1\right) &= \frac{m-1}{M-1} \\ &\vdots \\ \mathbb{P}\left(F_i|\, S_m B_1 \dots B_{i-1}\right) &= 1 - \frac{m-i+1}{M-i+1} = \frac{M-m}{M-i+1}. \end{aligned} \tag{4.2.2}$$

Finally we can approximate

$$\begin{aligned} \mathbb{P}\left(B_1 \dots B_{i-1} F_i\right) &= \sum_{m=i-1}^{M} \mathbb{P}(S_m)\mathbb{P}\left(F_i|\, S_m B_1 \dots B_{i-1}\right)\mathbb{P}\left(B_{i-1}|\, S_m B_1 \dots B_{i-2}\right) \dots \mathbb{P}\left(B_1|S_m\right) \\ &\approx \sum_{m=i-1}^{M} \mathbb{P}(S_m) \cdot \frac{M-m}{M-i+1} \cdot \dots \cdot \frac{m}{M} = \sum_{m=i-1}^{M} (M-m)\,\mathbb{P}(S_m) \cdot \frac{m!(M-i)!}{(m-i+1)!M!}. \end{aligned} \tag{4.2.3}$$

Let us denote

$$P_i = \sum_{m=i-1}^{M} (M-m)\,\mathbb{P}(S_m) \cdot \frac{m!(M-i)!}{(m-i+1)!M!}, \quad i = 1, \dots, M. \tag{4.2.4}$$

Now we introduce the binary variables z_{ki}, where z_{ki} equals 1 only if for the demand node k the i^{th} closest repairmen can reach it in time. The probability that a call from demand node k will be answered in time is $\sum_{i=1}^{M} P_i z_{ki}$ and the total fraction of calls answered in time can be approximated by

$$\frac{1}{K} \sum_{k=1}^{K} \sum_{i=1}^{M} P_i z_{ki}. \tag{4.2.5}$$

Note that for a given system all parameters needed to calculate expression (4.2.5) 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 $\hat{\mu} = 1/(\mathrm{TL} + 1/\mu)$ and then run some iterations of simulation to find a better approximation for $\hat{\mu}$.

4.3 Optimal repairmen allocation

Recall that we consider a system where relocation is not allowed. Fix the dispatching policy to the policy described in Section 4.2, when each demand node has preference list of the base stations. Under this restrictions to optimize the performance of the system we need to find the optimal assignment of the repairmen to the base stations. As the expected covered demand is the fraction of calls that will be answered in time for a given configuration, our goal is to find the assignment that maximizes the expected covered demand.

This problem can be formulated as an integer linear programming problem with decision variables x_r, $r = 1, \ldots, R$, representing the number of repairmen at base station r, and z_{ki}, the indicators that repairman $m_i^{(k)}$ can reach demand node k in time. The objective function is the approximation (4.2.5) of expected covered demand.

The total number of repairmen is M, so

$$\sum_{r=1}^{R} x_r = M.$$

The variables z_{ki}, $k = 1, \ldots, K$, $i = 1, \ldots, M$, and the variables x_r, $r = 1, \ldots, R$, are connected by the equation

$$\sum_{i=1}^{M} z_{ki} = \sum_{r \in N_k} x_r, \qquad k = 1, \ldots, K,$$

where N_k is the set of all bases from which demand node k can be reached in time.

The whole problem can be formulated as the following integer linear programming problem:

$$\begin{aligned}
\max \quad & \sum_{k=1}^{K} \sum_{i=1}^{M} P_i z_{ki} \\
\text{s.t.} \quad & \sum_{i=1}^{M} z_{ki} = \sum_{r \in N_k} x_r, \quad k = 1, \ldots, K \\
& \sum_{r=1}^{R} x_r = M \\
& x_r = 0, 1, 2, \ldots \qquad r = 1, \ldots R \\
& z_{ki} \in \{0, 1\} \qquad k = 1, \ldots, K, \quad r = 1, \ldots, R.
\end{aligned} \tag{4.3.1}$$

Note that the equalities in the constraints can be relaxed to inequalities, so the final problem is

$$\begin{aligned}
\max \quad & \sum_{k=1}^{K} \sum_{i=1}^{M} P_i z_{ki} \\
\text{s.t.} \quad & \sum_{i=1}^{M} z_{ki} \leq \sum_{r \in N_k} x_r, \quad k = 1, \ldots, K \\
& \sum_{r=1}^{R} x_r \leq M \\
& x_r = 0, 1, 2, \ldots \qquad r = 1, \ldots R \\
& z_{ki} \in \{0, 1\} \qquad k = 1, \ldots, K, \quad r = 1, \ldots, R.
\end{aligned} \tag{4.3.2}$$

The total number of decision variables is $R + KM$ and the total number of constraints equals $K + 1$, so the problem can be solved in $O((R + KM)^2 + (R + KM)(K + 1))$.

Example 4.3.1. Consider a map from Figure 4.5, where $K = 10$ and $R = 5$. Set $M = 6$, $\lambda = 0.01$ and $\hat{\mu} = 0.05$. Then the solution for the linear programming problem is

$$(0, 2, 3, 0, 1)\,,$$

which means that stations 1 and 4 are empty, two repairmen are assigned to station 2, three repairmen – to station 3 and one repairman – to station 5.

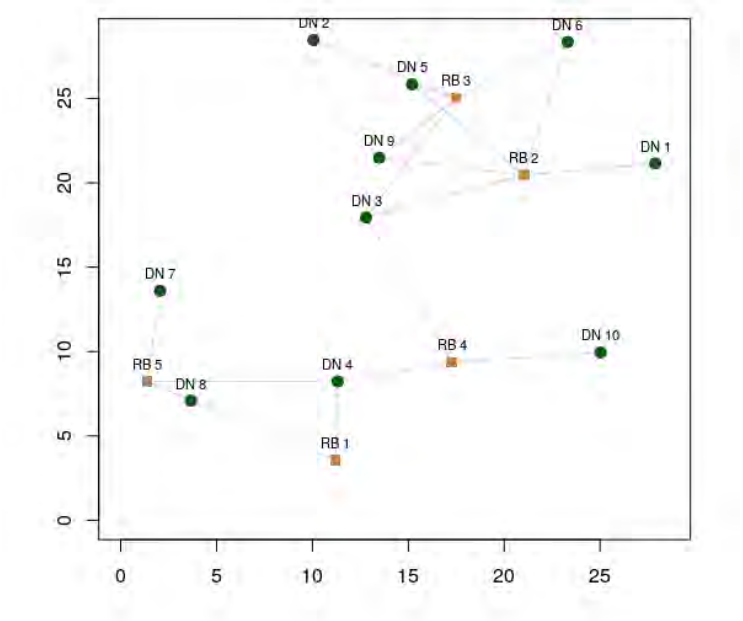

Figure 4.5: Example of a map of the service region with 10 demand nodes and 5 base stations. If a demand node can be reached in time from a base station, they are connected by an edge.

5 Dispatching policy

The dispatching policy is responsible for managing the calls that already arrived. It decides *when* and *which* repairman is assigned to each call. Recall that, once a repairman is assigned to a call, the assignment can not be changed.

When a failure occurs, a customer wants a repairman to arrive as soon as possible. However, when a repairman is dispatched from one of the base stations, the coverage of the customers around this base station decreases, which may lead to high costs in future. So a good dispatching decision finds a balance between immediate costs for a call and possible future costs from low coverage of the region.

In Section 5.1, we consider three heuristic dispatching policies. The first one is the closest-first dispatching policy widely used in practice and two other policies aim to optimize the coverage and the expected covered demand, respectively.

Under all of these policies, an incoming call can be put into the queue only if there are no idle repairmen. In all other cases someone has to be dispatched immediately. Next, we consider a policy where it is allowed to put a call in the queue for any state of the system. A more detailed description of this policy can be found in Section 5.2. A comparison of all considered policies based on the simulation results for systems with different properties can be found in Section 5.3.

In this chapter no relocation is allowed, so after the completion of a service a repairman returns to his previous base station and stays there until new dispatching. The assignment of the repairmen to the base stations is chosen to maximize the expected covered demand and can be found as a solution to the ILP problem 4.3.2 described in Section 4.3.

5.1 Dispatching policies without waiting

Recall that in state s, where the event e is the arrival of a call, 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 chapter relocation is not allowed, all elements of the matrix Y are always set to 0. The relocation matrix Y is also zero for the states with the event e of the second type (*"a repair in demand node k is finished"*).

In this section, once a call arrives in the system with at least one idle repairman, it should be answered immediately. If $\mathcal{F}(s)$ is the set of all repairmen that are not assigned to any call in state s, then for any state s with $\mathcal{F}(s) \neq \emptyset$ and event e of type *"a call arrives from demand node k"*, the set of possible actions is

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

If $\mathcal{F}(s) = \emptyset$, then call is put in queue and no decision should be made.

Note also that under these restrictions the set of possible actions for states with the event of type *"a repairman m arrives at a base station r"* is empty, as a call can not be put in queue is there is a traveling and therefore idle repairman in the system.

Consider a state s with the event $e =$ *"a call arrives from the demand node k"* and $\mathcal{F}(s) \neq \emptyset$. To describe the dispatching policy, we need to calculate the vector X depending on state s.

The most simple and most widely used in practice dispatching policy is the so-called *closest-first* policy. Under this policy, when a call arrives the closest idle repairmen is always dispatched. So

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

where $dist\left(l_n, k\right)$ is the distance between the destination of the repairman n and the source of the call, the demand node k.

Example 5.1.1. Consider a system with $K = 6$ demand nodes, $R = 2$ repairmen bases and $M = 3$ repairmen in the state shown in Figure 5.1. A call from demand node 1 arrives. The repairmen 2 at base station 2 is the closest one, so he is dispatched to fix the machine.

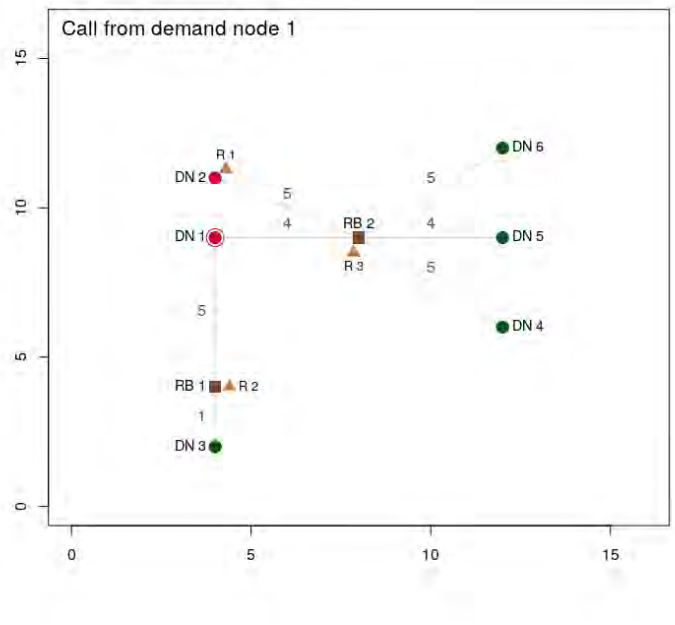

Figure 5.1: System for examples 5.1.1 - 5.2.1. Numbers on edges show the distance between nodes. Repairman 1 is doing a repair at demand node 2. Repairmen 2 and 3 are at base stations.

It is easy to see that the closest-first policy is not always the optimal one. If in example 5.1.1 repairman 2 is dispatched instead of repairman 3, then the resulting system can perform better when the next call arrives. To avoid such situations one can dispatch not the closest repairman, but the one that leaves the system in state with better expected performance.

One of the possible estimators of the expected system performance is *coverage*, i.e. the number of demand nodes covered by at least one repairman. When a call arrives for each idle repairman m, that can reach demand node k in time, the coverage of the system without him is calculated:

$$coverage(m) = \sum_{k':\kappa_{k'}=0} \mathbb{I}\left\{\exists n \in \mathcal{F}(s) : n \neq m, dist(l_m, k') \leq TL\right\}.$$

Then the repairman with the biggest $coverage(m)$ is dispatched. If there are no repairmen that can reach the source of the call in time, then the remaining coverage is calculated for all idle repairmen and then, again, the one with the biggest remaining coverage is dispatched.

Example 5.1.2. Consider again the situation from the Example 5.1.1. Then the coverage after dispatching of the repairman 2 is 3 and after dispatching the repairman 3 is 1. So according to the described policy the repairman 2 should be dispatched.

Note that the coverage only estimates the performance of the system for the next call. To estimate it for longer time one can calculate the expected covered demand, the fraction of calls that will be answered in time by the system. Similar to coverage first we calculate the expected covered demand after dispatching each of the repairmen m, that can reach the demand node in time, and dispatch the one with the highest remaining expected covered demand. If there are no

repairmen that can reach the demand node in time, we do the same for all repairmen from the set $\mathcal{F}(s)$.

As the expected covered demand is hard to compute, for computational study we use the approximation described in Section 4.2.

Example 5.1.3. For the same case as in examples 5.1.1 and 5.1.2 expected covered demand after dispatching of repairmen 2 and 3 is 0.45 and 0.15, respectively. So again the repairman 2 is dispatched.

Even though the closest-first policy is much simpler than the other two dispatching policies, the computational results for ambulance service networks showed that the difference in the performance between the closest-first policy and the expected covered demand based policy is small [17]. We discuss the computational results for our system under these three policies in Section 5.3 below.

5.2 Dispatching policies with waiting

Under the dispatching policies discussed in the previous section, it is mandatory to dispatch an idle repairman when a call arrives. However, if in the situation shown in Figure 5.1 we know that the repairman busy at demand node 2 is going to finish his repair earlier than any other repairman can reach demand node 1, it may be better to wait for him to finish and then dispatch him to demand node 2.

Inspired by this example, we extend the closest-first dispatching policy to the dispatching policy that chooses the repairman with the smallest response time. If it is a repairman that is now busy with a repair at some demand node, then the call is placed in queue and the repairman is dispatched there later.

For a repairman m, whose destination is a base station r, i.e. $l_m = r$, the response time to a call from the demand node k is

$$rt(k, m) = d_m + dist(l_m, k).$$

For a repairman m, whose destination is a demand node k', $l_m = k'$, the response time consists of the distance still left to demand node k', the length of repair and the distance from demand node k' to demand node k. If the length of repair t_{repair} is not known, it can be estimated from its distribution. Then the expected response time equals

$$rt(k, m) = d_m + \mathbb{E}t_{repair} + dist(k', k).$$

We consider two situations: when the length of repair can be estimated upon arrival of the repairman to the demand node and when it stays unknown. For the computational study of the second situation it is estimated by the 80th percentile of the repair time distribution.

Repairman $m = \text{argmin}_n\, rt(k, n)$ that minimizes the response time is assigned to the call. If $m \in \mathcal{F}(s)$ then $X_m = 1$ and he is dispatched immediately (so now $l_m = k$ and $d_m = d_m(s) + dist(l_m, k)$). If repairman m is busy with some repair then the call is placed in queue.

Example 5.2.1. Consider the situation from Figure 5.1. Assume that we know that the repairman busy at demand node 2 is going to be free in 1 time unit and that the traveling speed is one. Then repairmen 1, 2 and 3 can be in demand node 1 in time 3, 4 and 5, respectively. Repairman 2 has the smallest response time and so he is dispatched, which means that he first finishes his repair at demand node 2 and then goes to answer the new call.

More sophisticated dispatching policies based on maximizing the coverage and the expected covered demand can also be extended to the case when waiting is allowed. To this end, for all busy repairmen the distances to other demand nodes should be increased by remaining busy time. After this adjustment the coverage and the expected covered demand are calculated as before.

5.3 Computational results and conclusions

In this section we consider the performance of policies described in Sections 5.1 and 5.2. To this end, we generate systems with different parameters and compare the fraction of calls answered in time under each of the policies.

In particular, we consider the following five dispatching policies:

1. Closest-first dispatching policy (without waiting).
2. Dispatching policy based on maximizing the coverage (without waiting).
3. Dispatching policy based on maximizing expected covered demand (without waiting).
4. Minimal response time dispatching policy with unknown remaining repair time.
5. Minimal response time dispatching policy with known remaining repair time.

The last two policies allow waiting and the first three do not. The difference between policies 4 and 5 is in estimation of response time of repairmen busy at some demand node. For policy 4, the remaining repair time is set to 80th percentile of repair time distribution. For policy 5, we assume that once a repairman arrived at demand node he knows how much time the repair will take.

Relocation is not allowed and the starting state for all policies is the same, chosen to maximize expected covered demand (see Section 4.3).

Table 5.2 contains results of the simulation for the first three policies, numbered P1, P2 and P3, respectively, for different value of M, TL, ST and the map density. One can see that there is no policy that performs best for all systems. In most cases the third policy performs better than the second, but the relation with the first policy can differ depending on the system.

Next, we compare the first and the last two policies. Simulation results for these policies can be found in Table 5.2. For most of the systems both policies with waiting outperform (or at least perform equal) the traditional closest-first policy. Observed improvement increases with the increase of map density. This happens because in dense maps it is more likely that the demand nodes are close to each other. For most of the systems the improvement also increases with the increase of traveling times (but not when traveling times are too big).

Comparing the performance of policies 4 and 5 one can see that for most of the considered system accurate estimation of remaining repair time gives only a small improvement.

M	Map dens.	TL	ST = 5			ST = 10			ST = 20			ST = 50		
			P1	P2	P3	P1	P2	P3	P1	P2	P3	P1	P2	P3
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 5.1: Fraction of calls answered in time for dispatching policies P1-P3.

M	Map dens.	TL	ST = 5			ST = 10			ST = 20			ST = 50		
			P1	P4	P5	P1	P4	P5	P1	P4	P5	P1	P4	P5
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 5.2: Fraction of calls answered in time for dispatching policies P1, P4 and P5.

6 Relocation policy

The relocation policy is responsible for the location of idle repairmen. The simplest relocation policy is static policy, where each repairman is assigned to a base station and he returns there every time he becomes idle. In Section 4.3, we discussed how to compute the assignment of the repairmen that maximizes the expected covered demand.

However, when a repairman is dispatched to a call, a large area of the region may become uncovered and it may be optimal to change the location of other repairmen. Recall that in our system we allow one idle repairman to change his destination at the moment when another repairman is dispatched or when another repairman becomes idle.

In this section we consider three relocation policies and compare them with the static policy, and among each other. The first two policies are compliance tables constructed according to different algorithms. A compliance table relocation policy is a policy where the location of the repairmen depends only on the number of idle repairmen. The advantage of the compliance tables is the fact that they are precomputed (so can be used even for big systems) and easy to use.

The third policy is heuristic relocation policy based on the DMEXCLP heuristic introduced by Jagtenberg et al. [18]. The main difference of this policy from the compliance table is that decisions are made in real time, so that more information about the state of the system can be used. We consider two versions of this policy, with no restrictions and with restrictions on relocation, in Section 6.3.

Section 6.4 contains simulation results for all of this policies combined with dispatching policy described in Section 5.2.

6.1 MCRP compliance tables

The compliance table relocation policy is the policy where the location of the repairmen depends only on the number of idle repairmen. The location and the number of the busy repairmen is neglected, as well as the state of the capital goods. It allows to decrease the number of considered situations and compute the actions even for big systems.

For a system with M repairmen, a compliance table consists of M levels. Level m contains the allocation of the repairmen when there are m idle repairmen in the system. If one of them is dispatched to a demand node, other repairmen are relocated according to level $m-1$. If a repairman finishes a repair in a demand node, then he is redeployed and the other repairmen are relocated according to level $m+1$.

As the configuration of the idle repairmen may change after dispatching, we do not have to optimize the long run fraction of calls answered in time, but only the probability that the next call is answered in time. So if we denote by z_{mk} the indicator of the fact that at the level with m idle repairmen the demand node k is covered (meaning that at least one repairmen can reach it in time), then on the level m we want to maximize

$$\sum_{k=1}^{K} z_{mk}.$$

If x_{mr} is the number of repairmen at the base station r at level m then

$$z_{mk} \leq \sum_{r \in N_k} x_{mr}, \quad k = 1, \ldots, K,$$

where N_r is the set of all base stations from which the demand node k can be reached in time.

So, to construct the level m of the compliance table one should solve the following integer linear

programming problem:

$$\begin{aligned} \max \quad & \sum_{k=1}^{K} z_{mk} \\ \text{s.t.} \quad & z_{mk} \leq \sum_{r \in N_k} x_r, \qquad k = 1, \ldots, K \\ & \sum_{r=1}^{R} x_{mr} \leq M \\ & x_{mr} = 0, 1, 2, \ldots \quad r = 1, \ldots, R \\ & z_{mk} \in \{0, 1\} \qquad k = 1, \ldots K. \end{aligned} \tag{6.1.1}$$

Recall that in our system the relocation is restricted by one relocated repairman at a time. To include this restriction in the ILP formulation we are also going to introduce non-negative variables α_{mr} that represent the number of repairmen that arrived at base station r after going from level $m+1$ to level m. Then

$$\alpha_{mr} \leq x_{mr} - x_{m+1,r} \quad r = 1, \ldots, R, \quad m = 1, \ldots, M-1$$

and

$$\sum_{r=1}^{R} \alpha_{mr} \leq 1, \quad r = 1, \ldots, R, \quad m = 1, \ldots, M-1.$$

As these constraints connect different levels of the compliance table, the ILP problems can not be solved separately for each level and we have to construct a new ILP formulation for the whole table. The fraction of time that the system stays on level m equals to the probability that there are m idle repairmen in the system and the approximation of this probability was computed in Section 4.1. So the final objective function is

$$\sum_{m=1}^{M} \mathbb{P}(S_m) \sum_{k=1}^{K} z_{mk},$$

where S_m is the event of having m idle repairmen in the system.

The final ILP formulation is

$$\begin{aligned} \max \quad & \sum_{m=1}^{M} \mathbb{P}(S_m) \sum_{k=1}^{K} z_{mk} \\ \text{s.t.} \quad & z_{mk} \leq \sum_{r \in N_k} x_{mr}, && k = 1, \ldots, \quad K, m = 1, \ldots, M \\ & \sum_{r=1}^{R} x_{mr} \leq M, && m = 1, \ldots, M \\ & \alpha_{mr} \leq x_{mr} - x_{m+1,r} && r = 1, \ldots, R, \quad m = 1, \ldots, M-1 \\ & \sum_{r=1}^{R} \alpha_{mr} \leq 1, && m = 1, \ldots, M-1 \\ & \alpha_{mr} \geq 0, && r = 1, \ldots, R, \quad m = 1, \ldots, M-1 \\ & x_{mr} = 0, 1, 2, \ldots && r = 1, \ldots, R, \quad m = 1, \ldots, M \\ & z_{mk} \in \{0, 1\} && k = 1, \ldots K, \quad m = 1, \ldots, M. \end{aligned} \tag{6.1.2}$$

In total, there are $MK + RM + (M-1)R = M(K+2R) - R$ decision variables and $KM + M + (M-1)R + (M-1) + (M-1)R = M(K+2R+2) - 2R - 1$ constraints. So the computational complexity of solving this problem is $O\left(M^2 (K+2R)^2\right)$.

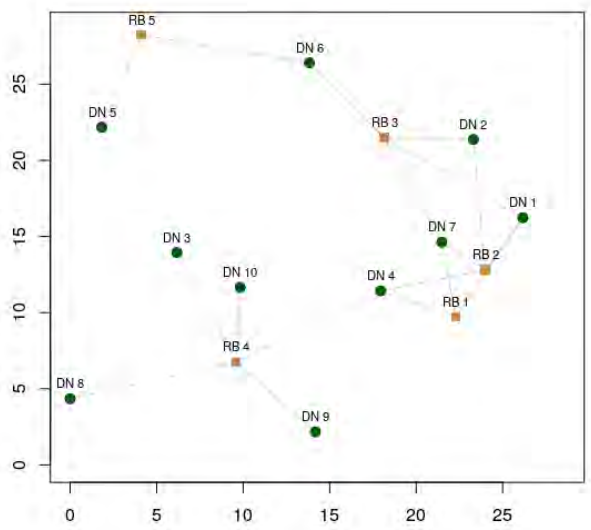

Figure 6.1: Example of a map with K = 10 and R = 5. Edges show distances that can be reached in time limit.

Example 6.1.1. Consider a system depicted in Figure 6.1 with K = 10, R = 5, M = 6, $\lambda = 0.01$, average repair time equals to 5 and TL = 10. Then the solution for the ILP problem 6.1.2 is the following compliance table:

Num. of free repairmen	RB1	RB2	RB3	RB4	RB5
1	0	0	0	1	0
2	0	0	1	1	0
3	0	0	1	1	1
4	1	0	1	1	1
5	2	0	1	1	1
6	3	0	1	1	1

6.2 AMEXPREP compliance tables

Consider a system where the number of repairmen is larger than the number required to cover all demand nodes. According to MCRP approach, when some repairmen are placed so that they cover all demand nodes, the location of the remaining repairmen does not affect the coverage. This leads to the situation when they are allocated in inefficient way and causes unnecessary number of relocations.

The hypothesis is that AMEXPREP compliance tables introduced in [4] can solve this problem. In this algorithm, the main goal is to optimize expected covered demand, not the number of covered demand nodes. The problem can again be formulated as an ILP problem.

Denote the number of repairmen located at the repair base r in the configuration for m idle repairmen by x_{mr}, $m = 1, \ldots, M$, $r = 1, \ldots, R$. The indicator y_{mki}, $m = 1, \ldots, M$, $k = 1, \ldots, K$, $i = 1, \ldots, m$, equals to 1 only if in the configuration for m idle repairmen the demand node k is covered by at least i repairmen.

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 repairmen a call from the demand node k is answered by the i^{th} closest

repairman. Same as in section 4.2, this probability is approximated by P_i computed for the system with $M' = m$. We denote this approximation by P_{mi}.

The ILP problem is formulated as follows:

$$
\begin{aligned}
\max \quad & \sum_{m=1}^{M}\sum_{k=1}^{K}\sum_{i=1}^{m} P_{mi} y_{mki} \\
\text{s.t.} \quad & \sum_{i=1}^{m} y_{mki} \le \sum_{r\in N_k} x_{mr}, && k=1,\dots,K, \quad m=1,\dots,M \\
& \sum_{r=1}^{R} x_{mr} \le m, && m=1,\dots,M \\
& \alpha_{mr} \le x_{mr} - x_{m+1,r} && r=1,\dots,R, \quad m=1,\dots,M-1 \\
& \sum_{r=1}^{R} \alpha_{mr} \le A, && m=1,\dots,M-1 \\
& \alpha_{mr} \ge 0, && r=1,\dots,R, \quad m=1,\dots,M-1 \\
& x_{mr} = 0,1,2,\dots && r=1,\dots,R, \quad m=1,\dots,M \\
& y_{mki} \in \{0,1\} && k=1,\dots,K, \quad m=1,\dots,M.
\end{aligned}
\tag{6.2.1}
$$

The third and the fourth constraints limit the number of relocations by A and α_{mr} is the indicator that a repairman is relocated from the base station r when the number of idle repairmen increases from m to $m+1$.

Example 6.2.1. Consider the same system as in Example 6.1.1. Then the solution for the ILP problem 6.2.1 is the following compliance table:

Num. of free repairmen	RB1	RB2	RB3	RB4	RB5
1	0	0	0	1	0
2	0	0	1	1	0
3	0	0	1	1	1
4	0	0	1	2	1
5	0	1	1	2	1
6	0	1	1	2	2

6.3 Relocation heuristic

When the compliance table relocation policy is used, the state that should be achieved after relocation does not depend on the current state of the system, only on the number of idle repairmen. On the contrary, heuristic approach to relocation uses all the information about the current state to optimize the decision. According to this approach, when the decision should be made, it is made to optimize some objective function under some restrictions. This approach is more flexible than the compliance tables. We consider the DMEXCLP heuristic relocation policy introduced in [18] and adjusted for our system.

There are two types of decision moments:

1. When a service is completed and there are no jobs assigned to the repairman that became idle. In this case he must be dispatched to one of the base stations.

2. When an idle repairmen is dispatched to an incident, it should be decided whether other idle repairmen should be relocated or not.

We limit the number of relocations by one per decision moment. For the first situation, only the repairmen that just became idle can be relocated. For the second situation, only one repairman can change his base station.

According to the DMEXCLP relocation heuristic policy the action that maximizes the expected covered demand is always chosen. In the first case one repairman is added subsequently to all base stations, the expected covered demand is calculated and the base station that leads to the biggest result is chosen.

For the second situation all pairs of the base stations (r_1, r_2), where there is at least one repairman at the base station r_1, are considered. We calculate the improvement in the expected covered demand after the relocation of a repairman from the station r_1 to the 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 repairman from the station r_1' to the station r_2'. If the maximum improvement in not positive then no relocation happens.

Note that for both situations the expected covered demand is computed only for the working capital goods.

The problem of large relocation times leading to the possibly poor performance of the compliance tables, can appear for this relocation policy as well. In [5] imposing restrictions on the relocation is proposed as a solution to this problem. There are three possible parameters that can be used to describe these restrictions.

In the first situation the maximum relocation distance can be set. In this case, the best station is chosen among base stations in this distance from the demand node where the repairman is now. If there are no such base stations then the choice is made from all base stations.

In the second situation 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 this or less distance between r_1 and r_2 are considered. If there are no such pairs the relocation is forbidden. Note that this distance can differ from the distance for the first situation. If the minimum improvement threshold is set then the relocation happens only if the improvement in the expected covered demand exceeds this threshold. Setting this threshold equal to 0 means no restriction. The threshold larger than the number of demand nodes 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 6.4 we consider this policy with different parameters and study the improvement that can be gained by parameter tuning.

6.4 Computational results and conclusions

In this section we present the results of the simulation of the system with different relocation policies. The dispatching policy is fixed to the policy that always dispatches a repairman with the smallest response time (see Section 5.2) without remaining repair time estimation. Five relocation policies are considered:

1. Static relocation policy;
2. MCRP compliance tables;
3. AMEXPREP compliance tables;
4. DMEXCLP heuristic relocation without constraints;
5. DMEXCLP heuristic relocation with constraints.

The number of relocations is limited to one relocation for each decision moment.

The fifth policy requires more explanation. As mentioned before there are three parameters that define restrictions for the heuristic relocation policy: the maximum distance of the relocation

after the end of the service, the maximum distance of the relocation upon dispatching and the minimum performance improvement for which the relocation upon dispatching is allowed. As it is not known how to optimize these parameters, we simulated the system for the first and the second parameters equal to 0.5TL, TL, 2TL and 100TL and the third parameter equal to 0, 1, 5, 100. The best result for each type of the system was chosen and used as a result for the fourth policy.

We considered 27 types of the system with three possible values for each of such parameters as the average service time, the time limit and the map density. The number of repairmen is set to 13 and the break down rate is set to $\lambda = 0.01$. For each type of the system 10 maps were generated and 10 iterations of the simulation were performed for each map. The results of the simulation can be found in Table 6.1.

One can see that compliance tables perform badly for most of the systems compared to other policies. The only type of systems for which they perform better than the static policy is the systems with dense map and service times much bigger than TL. The reason for that is the fact that both MCRP and AMEXPREP algorithms ignore the distances, so when distances are not neglectable it leads to inefficient relocation and poor performance.

Policy 4 (DMEXCLP heuristic policy without restrictions) outperforms the compliance tables for all systems, because it uses more information about the state of the system to make the relocation decision. However, it also ignores the distances, so for maps with large distances and small density we observe that it performs worse than the no-relocation policy.

Finally, implementing Policy 5 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 results in less than 60% of calls answered in time. The difference in performance between Policy 5 and Policy 4 shows the importance of relocation restrictions and accurate tuning of the parameters of these restrictions.

Service Time	Time limit	Map density	P1	P2	P3	P4	P5
ST = 5	TL =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
	TL =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
	TL = 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
ST = 10	TL =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
	TL =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
	TL = 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
ST = 20	TL =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
	TL =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
	TL = 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 6.1: Simulation results. Fraction of calls answered in time for policies P1-P5.

7 Approximate dynamic programming

To construct an optimal policy for the Markov decision process described in Section 3 one can use such algorithms as policy iteration or value iteration. However, as the state space of the process is continuous, these algorithm are impossible to implement computationally. Even if we discretize the state space, the number of possible states for realistic-sized systems is large and the problem remains intractable. We consider this approach in Section 8 below and show that it is computationally difficult, even for relatively small systems.

On the other hand, in Sections 5 and 6 we saw that making decisions based on such metrics as the expected covered demand and the response time leads to well performing policies. The problem is that we do not know what is the optimal way of using this metrics in making decisions.

The Approximate Dynamic Programming (ADP) approach combines the ideas of Markov Decision theory and the heuristic approach. The goal is to find an approximation of the value function of the process as a combination of several basis functions, such as the expected covered demand and response time, and choose actions based on this approximation. Section 7.1 contains more detailed description of the approach. Used basis functions are discussed in Section 7.2. Two approaches to finding a good approximations are discussed in Sections 7.3 and 7.4, respectively. The computational results including comparison to the DMEXCLP heuristic policy (see Section 6.3) are presented in Section 7.5.

Our adaptation of the ADP approach to the problem is inspired by the studies applying ADP to the real-time ambulance management, especially the papers of Maxwell et. al. [25] and Nasrollahzadeh et al. [28].

7.1 Approximate solution

Let us consider a discounted version of the process $\{s_n,\ n = 0, 1, \dots\}$, described in Section 3 with discount factor γ. Fix a policy π and denote by $V_\pi(s)$ the expected total discounted costs when $s_0 = s$ under this policy:

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

If policy π is the optimal policy, then $V(s)$ 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 $s' = \Phi\left(s, \pi(s), \omega(s, a)\right)$ is the next state of the process when action a is taken, and

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

Hereinafter, we denote $V_\pi(s)$ for the optimal policy π by $V(s)$.

As the state space of process $\{s_n,\ n = 0, 1, \dots\}$ is infinite, the optimal action can not be computed in advance for each state. But if for each action $a \in \mathcal{A}(s)$ we can calculate $\mathbb{E}_a\left[c(s, a, s') + \gamma^{t(s')-t(s)} V(s')\right]$, then being in state s we can find the optimal action a. Note that given state s and action a the distribution of the state s' is known (see Section 3.5 for details), so the only problem is to calculate $V(s')$.

To overcome the problem of infinite 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 consider an affine combination of several basis functions $\varphi_i(\cdot),\ i = 1, \dots, I$, as such an approximation. So the approximate value function is

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

where $\alpha_i, i = 0, \ldots, I$, are coefficients, and the approximate optimal policy is defined by

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

The choice of basis functions is very important for the performance of the approach. First, they should be possible to compute explicitly for any state s. Second, they should be able to characterize the optimal value function, as otherwise it is impossible to achieve an accurate approximation to it. We discuss our choice of basis functions in Section 7.2.

When the set of basis functions is chosen, the approximation is improved by tuning the vector of coefficients $\alpha_i,\ i = 0, \ldots, I$. We consider two algorithms to do this: approximate policy iteration and a genetic algorithm. We discuss these approaches in Sections 7.3 and 7.4, respectively.

In practice, the expected costs after taking action a in state s, $\mathbb{E}_a \left[c(s, a, s') + \gamma^{t(s')-t(s)} \hat{V}(s') \right]$, can be still hard to compute, even when the distribution of s' depending on s and a is known. To reduce the computational time, we estimate this value by simulation. We simulate the next state s' several times, then for each realization compute expected costs and use the average of these costs as an approximation.

7.2 Basis functions

In our approach we use six basis functions. Some of them describe the ability of the system to respond for future demand, for example coverage and expected covered demand. These functions were already used in Sections 5 and 6 to construct heuristic policies. Other functions (for example, the number of unassigned calls and the number of unreachable calls) approximate future penalties for decisions made in the past.

Number of unreachable calls

Consider a call for which a repairman was already dispatched but he is still on his way and he is not going to reach the place in time. Imagine that this call did not pass the time limit yet. Then this call will cause a penalty in the future. The first basis function $\varphi_1(\cdot)$ computes the number of such calls in state s:

$$\varphi_1(s) = \left| \{k \in \mathcal{K} \mid 0 < \kappa_k(s) < \mathrm{TL}\} \cap \{l_m(s),\ m = 1, \ldots, M\} \right|.$$

Number of unassigned calls

Each call unassigned in the current state may cause costs later. First, this call may be answered late and lead to paying a penalty. Second, sooner or later a repairman is going to be dispatched to this call, which will lead to decrease in coverage. So the second basis function counts the number of unassigned calls in state s:

$$\varphi_2(s) = \left| \{k \in \mathcal{K} \setminus \{l_m(s),\ m = 1, \ldots, M\} \mid \kappa_k(s) > 0 \text{ or } e(s) = \text{"a call arrived from dem. node } k\text{"}\} \right|.$$

Number of missed unassigned calls

Opposite to unreachable calls for which we already paid, the penalty missed unassigned calls still require dispatching of a repairman. This will lead to a decrease in coverage that we can not see from current coverage metrics. At the same time as part of the penalty caused by this calls was already paid they can not be considered equal to other unassigned calls. So the third basis function is the number of unassigned calls that already passed the time limit:

$$\varphi_3(s) = \left| \{k \in \mathcal{K} \setminus \{l_m(s),\ m = 1, \ldots, M\} \mid \kappa_k(s) \geq \mathrm{TL}\} \right|.$$

Uncovered demand nodes

If a demand location is not covered in state s and a failure occurs there in the near future, it causes costs, so the number of uncovered demand nodes is an important metric of the state. We consider only the demand nodes with working machines as only they can generate demand:

$$\varphi_4(s) = |\{k \in \mathcal{K} \mid \kappa_k = 0 \text{ and } rt(k,m) > \text{TL } \forall m = 1,\dots,M\}|,$$

where $rt(k,m)$ is the estimation of time when repairman m can reach demand node k. For idle repairmen, this is equal to the distance to the demand node and for repairman assigned to a call it is estimated as distance to demand node k plus remaining service time estimated by 80th percentile of the service time distribution.

Expected covered demand

In Section 6, we saw that a heuristic relocation policy based on maximizing the expected covered demand shows high performance. This is why we included expected covered demand in the set of basis functions. It is calculated according to approximation from Section 4, where the distance from repairman m to demand node k is replaced by response time $rt(k,m)$. It ignores broken machines and is calculated based on working machines only.

Average response time

The last function is the average response time for all working machines. For a pair of a repairman and a machine the response time $rt(k,m)$ is estimated in the same way as for function $\varphi_4(\cdot)$. Then for each demand node we choose the smallest response time and calculate the average. So if $\mathcal{W}(s)$ is the set of working machines and $W(s)$ is the number of working machines, then

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

Future basis functions

All basis functions described above characterize the *current* state of the system. However, when we make relocation decision we are not only interested in the state right after the decision is made, but also in the state upon arrival of the relocated repairman. So, following [28], we introduce basis functions φ_7 to φ_{12} which characterize the state of the system after the arrival of one of the traveling repairmen.

To compute them, we find the repairman that will arrive at his destination first of all traveling repairmen. Then we construct the state of the system at the moment of his arrival assuming that no other events happened and compute functions φ_1 to φ_6 for this state. If there are no traveling repairmen, then the values of the functions φ_7 to φ_{12} are equal to the values of the functions φ_1 to φ_6.

We performed the computational study both with and without the future functions to see their impact on the result.

7.3 Approximate policy iteration

When the set of basis functions is determined, the next challenge is to find a vector of coefficients α_i, such that the approximation

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

is close to $V(s)$, the real value function of the process.

Approximate policy iteration extends the policy iteration algorithm used to compute the optimal policy. In the policy iteration algorithm each step improves the current policy. Each step

of approximate policy iteration algorithm improves not only the policy, but the approximation as well.

The algorithm is formulated as follows:

1. Initialize a vector of coefficients $\alpha^{(1)}$ and a set of states $\mathcal{S}^{(1)}$. Set $n = 1$.
2. Consider the policy $\hat{\pi}^{(n)} = \hat{\pi}(\alpha^{(n)}, \cdot)$ based on approximation $\hat{V}(\alpha^{(n)}, \cdot)$.
3. For every state s from set $\mathcal{S}^{(n)}$, perform N_{sim} simulation runs of the process with policy $\hat{\pi}^{(n)}$ starting from state s over time period $[0, T]$. Let $C(s, j)$ be the total discounted costs and let $s'(s, j)$ be the last state of the j^{th} run of the simulation.
4. Set
$$\alpha^{(n+1)} = \underset{\alpha}{\operatorname{argmin}} \sum_{s \in \mathcal{S}^{(n)}} \sum_{j=1}^{N_{sim}} \Big(C(s, j) - \hat{V}(\alpha, s) \Big)^2 .$$
Set $\mathcal{S}^{(n+1)} = \big\{ s'(s, 1),\ \forall s \in \mathcal{S}^{(n)} \big\}$.
5. Increase n by 1. If n is less than the chosen maximum number of iterations N_{max}, then go to step 2. If $n = N_{max}$, then stop and output the current vector α.

Let us discuss each step of the algorithm in detail.

First, we initialize some arbitrary vector of coefficients $\alpha^{(1)}$ and set of states $\mathcal{S}^{(1)}$. The vector of initial coefficients can be constructed so that the resulting policy behaves according to common sense (for example, set the coefficient for unassigned calls equal to 1 to impose dispatching and all other coefficients to 0). The initial set of states $\mathcal{S}^{(1)}$ can be obtained, for example, as a set of final states of a series of simulation runs starting from the same state.

During each iteration of the algorithm, a new policy is determined by new vector α. During the third step we observe the real costs under this new policy and then in Step 4 choose new coefficients of the approximation to improve the fit. The latter is achieved by minimizing the sum of squared differences between the simulated and the approximated costs.

Along with the vector of coefficients, the set of states is also updated through the iteration procedure. This is necessary as under the new policy the process may not visit old states any more.

7.4 Genetic algorithm

Genetic algorithms are widely used for optimization problems to find good approximations to the optimal solutions [31]. We use a genetic algorithm to tune vector of coefficients α. Using this approach, we do not look for a close approximation of the value function, but are interested in a vector of coefficients α that leads to high-performing policy $\hat{\pi}(\alpha, \cdot)$.

To define a genetic algorithm, we need to define population, mutation operator, crossover operator and selection of the next generation. In our case, population is a set of vectors $\alpha^{(n)}$, $n = 1, \ldots, N$, where N is the size of population. At the beginning, we initialize population by adding random vectors to the same vector $\alpha^{(0)}$.

During each iteration of the algorithm, a new population is constructed from the current population by joining N candidate solutions from the current population, N mutated solutions and N child solutions, which are results of the crossover operator.

Mutation is applied to each candidate solution from the current population. Given a solution, the mutation operator adds a vector, each component of which is normally distributed with mean 0. The variance is chosen according to the size of the system.

For the crossover operator, we select randomly N pairs of candidate solutions and compute the average in each pair. This is a good crossover operator, since α is a vector of coefficients of a linear combination.

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

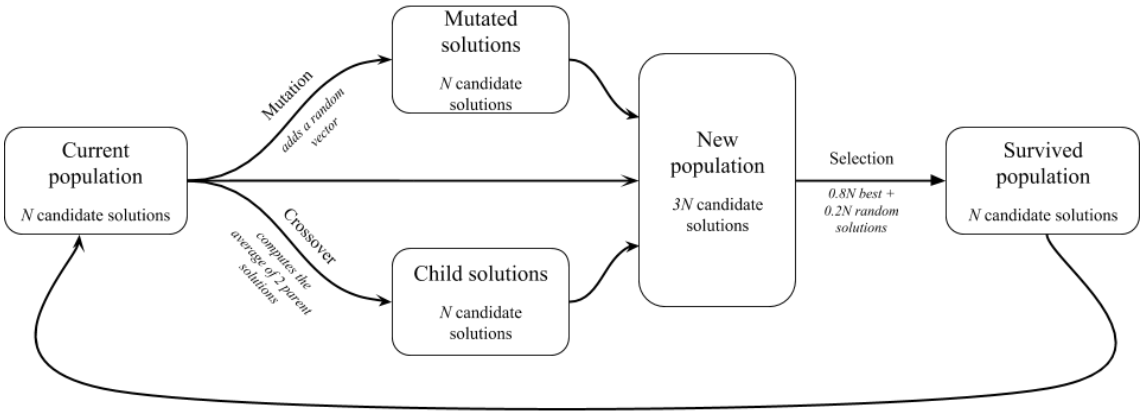

Figure 7.1: Genetic algorithm.

in time is observed. Then only the $0.8N$ best performing and $0.2N$ randomly chosen candidate solutions from the new population survive. The scheme of the algorithm is depicted in Figure 7.1. The algorithm stops after a certain number of iterations.

7.5 Computational results and conclusions

Due to the computational complexity, we did not investigate the performance of the ADP approach for several different maps, as we did for dispatching and relocation heuristics in Sections 5.3 and 6.4. We fixed a service region depicted in Figure 7.2 with three repairmen. The failure and the service rate were set to $\lambda = 0.01$ and $\mu = 0.2$, respectively.

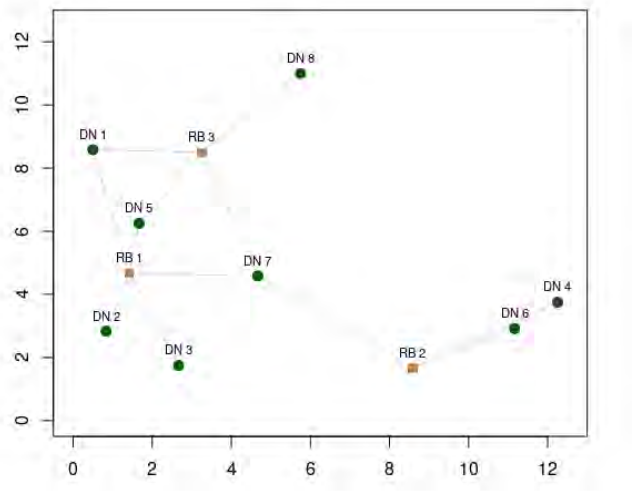

Figure 7.2: Map for the computational experiments.

We discussed two approaches for finding the vector α resulting in high performance of the respective policy: the approximate policy iteration and the genetic algorithm. We ran 15 iterations

Figure 7.3: Approximate policy iteration. Average error plotted against iterations.

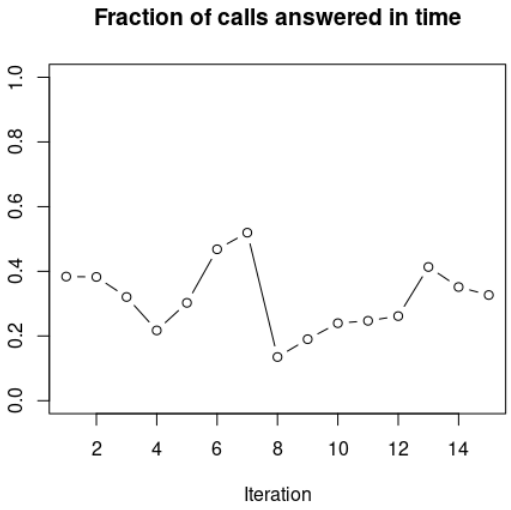

Figure 7.4: Approximate policy iterations. Fraction of calls answered in time.

of both of them. The initial α was set to $(0,\ 1,\ 0.8,\ 0,\ -0.1,\ 0.01)$ for both approaches.

The approximate policy iteration was performed for six basis functions and it showed poor performance. The average square error after each iteration is shown in Figure 7.3. As one can see, there is no convergence. After each iteration, we also simulated the system 10 times, starting from the state with a repairman at each base station and all machines working, and computed the average fraction of calls answered in time. The results are depicted in Figure 7.4. The performance is unstable and it does not improve from iteration to iteration.

Next, we performed 15 iterations of the genetic algorithm. The variance of the normal distribution of mutation operator was set to 2. The size of population is $N = 50$ candidate solutions. The initial population was constructed by adding random vectors to the initial vector $(0,\ 1,\ 0.8,\ 0,\ -0.1,\ 0.01)$. Candidate solutions were evaluated by simulation of the system starting from the state with one repairman at each base station.

The genetic algorithm was performed in case of 6 and in case of 12 basis functions, meaning with and without future basis functions. At each iteration we tracked the fraction of calls answered in time by policy corresponding to the best candidate solutions. The results can be found in Figure 7.5. After 15 iterations, the best result in case of 6 basis functions is on average 80% of calls answered in time. In case of 12 basis functions the best result is 93% of calls answered in time, so the future functions play an important role in the performance of the approach.

For this map we also performed the simulation of the system with heuristic policy, where the repairman with the minimum response time is always dispatched and the relocation is done according to DMEXCLP heuristic. The best result achieved by this policy is 89% of calls answered in time. So we conclude that the ADP approach leads to higher-performing policy than the heuristics approach.

The disadvantage of the ADP is that it is more time consuming than the heuristic policy. For this example, it took 1 day to perform 15 iterations of the genetic algorithm, whereas it takes only one hour to tune the restriction parameters in the heuristic policy.

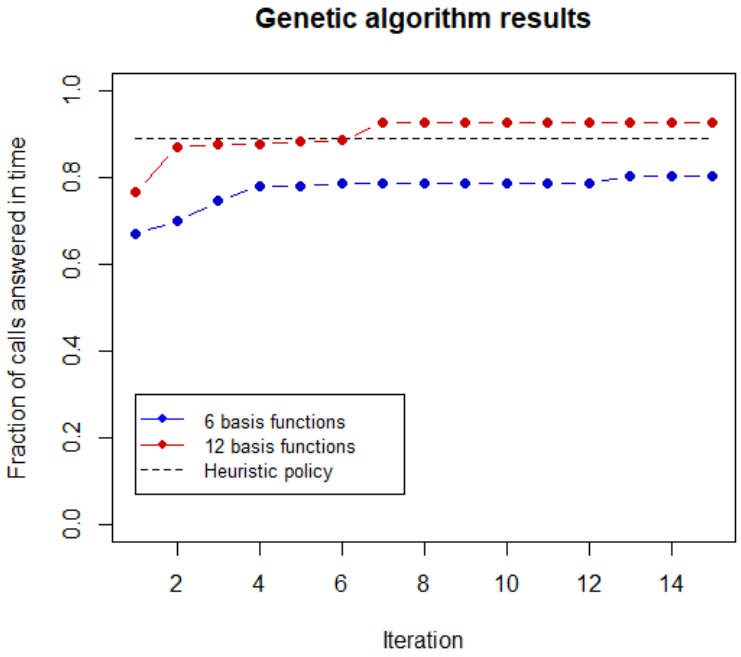

Figure 7.5: Genetic algorithm. Fraction of calls answered in time plotted against iterations.

8 Discrete-time model

The continuous-time process described in Section 3 gives the most accurate model of the system under our assumptions. However, this process has an infinite state space, which makes it impossible to perform the policy iteration computationally.

In this section, we discuss a discrete-time approximation of the original process. To that end, we discretize the map of the regions. Time is also discretized. This leads to the fact that several events can happen between subsequent states of the process, so we no longer restrict ourselves to the states with at least one event, but consider the state of the process at each time unit.

The resulting process has a finite state space that allows us to perform the policy iteration. However, the state space of the realistic systems is still too big, so the policy iteration was performed only for a small instance. For this instance, the performance of the optimal policy is compared to the performance of the best heuristic policy obtained in Sections 5 and 6 and the performance of the output policy of the ADP approach.

In Section 9 below, we use the discrete-time model introduced in this section to obtain theoretical results for the systems under heavy and light traffic regimes.

8.1 Process description

In this section we construct a discrete-time model of the system. Recall that for the continuous time model, we consider the system at each moment when some event happens. The main difference of the discrete-time model from the continuous-time one is that the time is discretized and we consider the system every time unit, no matter how many events happened since the last state.

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 intractable from computational perspective (for example, requires big amount of memory). Our computational study is based on simulated maps, not the real data, so we avoided the problem of choosing the correct time unit duration.

First, we approximate the original map by a new map where all distances between locations are the original distances rounded to the integer numbers of time units. It is better to use the ceiling because for each state of the new process we look at the events in the past.

Then we need to define the state space. Recall that for continuous-time process we consider only the moments when some event happens and the state of the process is described by a tuple

$$s = (t, e, \mathfrak{m}, \kappa)\,,$$

where t is the time, e is the event, $\mathfrak{m}$ describes the location of the repairmen and κ describes the state of the capital goods.

For the discrete-time process we consider the state of the process at each time unit. So the time of state s_n is always $t_n = n$ and so, as the optimal action in state s should not depend on time $t(s)$, we can ignore the time component in the discrete-time version of the process.

The vector $\mathfrak{m}$ remains unchanged. It again contains the pairs of destination and the distance to this destination for each repairman. For the discrete-time setting distances take only integer values, so there in only a finite number of possible locations of a repairman.

Vector κ again contains the state of each of the machines: the time in queue if the machine is broken, 0 if it is working and -1 if it is in repair. To have finite number of possible states of each machine the time in queue is bounded by the time limit. If the time limit for a broken machine k already passed we set $\kappa_k = \mathrm{TL}$.

For the continuous-time model we assumed that there is only one event happening at a time. However, for discrete-time process during transition from state s_n to state s_{n+1} more than one event may happen (for example, two machines may break down). So now the event e is a triple of sets:

$$e = (\mathcal{K}_1, \mathcal{K}_2, \mathcal{M}_a)\,,$$

where $\mathcal{K}_1$ is the set of machines that broke down during the last time unit, $\mathcal{K}_2$ is the set of the demand nodes where a repair ended during the last time unit, and $\mathcal{M}_a$ is the set of all repairmen that arrived to their destination during the last time unit. As we consider the state of the process at each time unit it is possible that no events happened and all three sets are empty.

The probability that a working machine will break down during a time unit is $p = 1 - e^{-\lambda}$ and the probability that a repair in progress will end during a time unit is $q = 1 - e^{-\mu}$. Denote by $W(s)$ number of all working machines in state s and by $H(s)$ the number of demand nodes where there is a repairman doing a repair. Then the probability of having event e with particular $\mathcal{K}_1$ and $\mathcal{K}_2$ is

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

The last component of the event, set $\mathcal{M}_a$ of the arrived repairmen is deterministic and depends only on the previous state of the system.

To keep the decision-making process close to that of the continuous-time model, it is beneficial to choose a time unit so that there is usually not more than one event happening per time period.

As the set of possible states of a repairman, the set of possible states of the machines and the set of possible events are all finite, the state space of the process is finite as well. However, the number of states grows exponentially in the number of machines and in the number of repairmen, so for realistic systems the state space can be very large.

A decision a in state s consists of three binary matrices X, Y and Z. Matrix X describes the dispatching decision for all call from set $\mathcal{K}_1$: $X_{mk} = 1$ only if repairman m is dispatched to demand node k. Matrix Y describes the redeployment decision for the repairmen that became idle at demand nodes from the set $\mathcal{K}_2$: $Y_{ml} = 1$ only if repairman m is redeployed to location l. Matrix Z describes the relocation decision: $Z_{mr} = 1$ only if repairman m is relocated to base station r.

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

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

The transition costs from state s_n to state s_{n+1} are again defined as

$$c(s_n, a_n, s_{n+1}) = |\left\{k \in \mathcal{K} \mid \kappa_k(s_n) < \mathrm{TL} \text{ and } \kappa_k(s_{n+1}) = \mathrm{TL}\right\}| + \epsilon|\left\{k \in \mathcal{K} \mid \kappa_k(s_n) = \mathrm{TL}\right\}|.$$

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.

8.2 Computational results

In this section we provide the results of a computational study. Again, as for the ADP approach in Section 7.5, we fixed one small system and performed the computations for this system.

Consider the service region depicted in Figure 8.1 with two service engineers. The working time rate is $\lambda = 0.01$ and the service rate is 1, so the probability that a working machine breaks down during one time is $p = 0.095$ and the probability that an on-going repair finishes in one time unit is $q = 0.6$. Time limit is $\mathrm{TL} = 3$, which means that once a machine is broken a repairman should be dispatched immediately to avoid paying a penalty.

This a small unrealistic example, but even for this system the number of possible states is 78 432 an it takes more than two days to define the process. The main problem is computing and storing the transition probabilities. It is hard to estimate how much time and memory it will take to compute and store this matrix for realistic systems.

After defining the process, we performed policy iteration for this system. Then we performed 10 iterations of simulation of the system under the policy obtained from the policy iteration. The average resulting penalty is 0.006 per time unit. The average fraction of calls answered in time is 0.87. To compare, the fraction of calls answered under DMEXCLP + minimum response time heuristics policy is 0.82, which shows that this policy performs close to optimal at least for this instance.

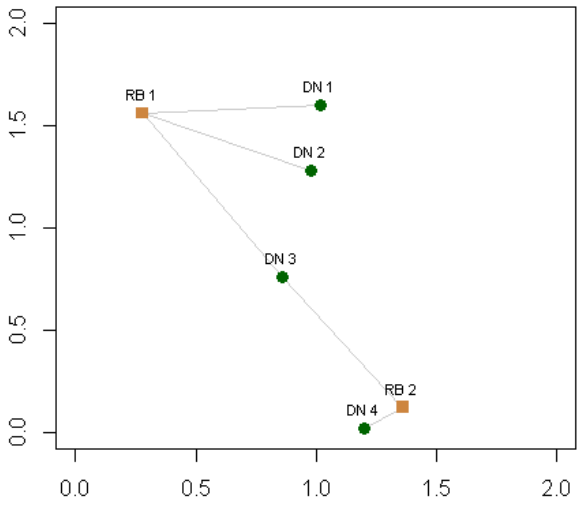

Figure 8.1: Map for computational experiments.

9 Structural results

In this section, we investigate the performance of discrete-time process described in Section 8 under extreme regimes. We consider two regimes: either when service times are large compared to traveling times and working times, or when working times are large compared to other parameters.

Using techniques proposed by Katehakis and Levine [21], we obtain some theoretical results about the properties of the optimal policies under these regimes.

9.1 Asymptotically optimal policies

Consider a discounted version of the discrete-time process s_n described in Section 8.1 with discount rate β. If the policy π is fixed, then, given that the state at time 0 is s the expected discounted cost and the average expected cost for the system are defined by

$$V_\pi(s) = \mathbb{E}\left[\sum_{n=0}^{\infty} e^{-\beta t} c(s_n) \mid s_0 = s\right],$$
$$U_\pi(s) = \lim_{N\to\infty} \frac{1}{N}\mathbb{E}\left[\sum_{n=0}^{\infty} c(s_n) \mid s_0 = s\right].$$

Under a fixed policy π, the value function $V_\pi(s),\ s \in S$, is the unique solution of the following system of equations

$$V_\pi(s) = c(s) + e^{-\beta} \sum_{\substack{(\mathcal{K}_1,\mathcal{K}_2),\\ \mathcal{K}_1\subset\mathcal{W}(s),\\ \mathcal{K}_2\subset\mathcal{H}(s)}} P(s,\mathcal{K}_1,\mathcal{K}_2) V_\pi\left(\Phi\left(s,\pi(s),\mathcal{K}_1,\mathcal{K}_2\right)\right) \tag{9.1.1}$$

The following proposition follows from the fact that the state space and the action space are finite.

Proposition 9.1.1. *There exists a β_0 and a policy $\pi_0 \in \Pi$ such that $V_{\pi_0}(s) \le V_\pi(s)$ for all s, for all $\pi \ne \pi_0$ and for all $\beta \in (0,\beta_0)$ and $\lim_{\beta\to 0} \beta V_{\pi_0}(s) = U_{\pi_0}(s)$.*

Proof. See [11]. □

Thus, a policy which minimizes $V_\pi(s)$ for small values of β also minimizes $U_\pi(s)$.

To obtain the results the extreme regimes, we multiply the probability of failure or the probability of the end of repair by parameter ρ and find policies that are optimal for small values of ρ. As after multiplying either p or q by ρ, the probability $P(X,\mathcal{K}_1,\mathcal{K}_2)$ is polinomial in ρ, we can regroup all terms to get the following expression:

$$V_\pi(s) = \sum_{i=0}^{\infty} \nu_i(s)\rho^i, \tag{9.1.2}$$

where the number of non-zero coefficients ν_i is finite. Note that as ρ goes to 0 the leading terms dominate. Let us call the policy, that maximizes the leading coefficients, asymptotically optimal policy.

Proposition 9.1.2. *There exists $\rho_* > 0$ such that the asymptotically optimal policy π_* is optimal for $\rho \in (0,\rho_*)$.*

Proof. See Theorem 1 in [21]. □

Thus, once we found the optimal policy for $\rho = 0$, it is also optimal for small values of ρ. In next sections we consider three different extreme regimes and will see that if $\rho = 0$ the system is equal to a simpler system for which the optimal policy can be found explicitly.

9.2 Large repair time

For this section we set the additional costs $\epsilon = 0$. Note that this results can be extended to the case of non-zero ϵ. However, we are not going to discuss it.

In our model large repair times are represented by q close to 0. So to study this regime we multiply q by ρ and consider small values of ρ. Then substituting (9.1.2) into (9.1.1) and sending $\rho \to 0$, we obtain the following equations for ν_0:

$$\nu_0(s) = c(s) + e^{-\beta} \sum_{\mathcal{K}_1 \subset \mathcal{W}(s)} P\left(s, \mathcal{K}_1, \emptyset\right) \nu_0\left(\Phi\left(s, \pi(s), \mathcal{K}_1, \emptyset\right)\right). \tag{9.2.1}$$

Note that these equations describe the aggregated costs for the system where repairs never end, and once a repairman is dispatched to a call he is lost forever. So computing ν_0 for a state s we can ignore all machines that are already in repair or have a repairman assigned to them, as well as the repairmen that are already assigned somewhere.

Denote by $\Omega(s)$ the set of all machines that are either working in state s or have been waiting in the queue for no longer than TL time units. Not reached in time these machines cause costs in future. If a working machine breaks down in τ time units and it is not reached within the time limit, it adds $e^{-\beta(\tau+\mathrm{TL})}$ to $\nu_0(s)$. If a machine that broke down τ time units ago is not reached in time it adds $e^{-\beta(\mathrm{TL}-\tau)}$ to $\nu_0(s)$. So due to discounting, the later the machine breaks down the smaller costs it causes.

As dispatched repairmen are lost, starting from state s we can reach only a limited number of machines in time. To minimize the costs, it is beneficial to reach in time those machines that break down first.

In order to describe the optimal policy precisely we need to introduce the following notation.

$$\begin{aligned} K_{covered}(s) = {} & \text{the maximum number of machines from } \Omega(s) \\ & \text{that are covered in state } s, \text{ if one repairman} \\ & \text{can cover only one machine.} \end{aligned}$$

Finding $K_{covered}$ is the same as finding the size of the maximum matching in a bipartite graph of repairmen and machines from $\Omega(s)$, where an edge represents a distance, less than TL, between a repairman and a demand node. If in state s there are traveling repairmen whose destinations are some base stations, the distance from such repairmen to demand nodes changes in time, so $K_{covered}$ should be compared as the expected maximum number of covered demand nodes conditioning on the failure time of working machines.

Next we define

$$\begin{aligned} \Theta(s) = {} & \text{the number of subsets of } \Omega(s) \text{ of size } K_{covered}(s), \\ & \text{such that each of these subsets is covered in state } s, \\ & \text{if one repairman can cover only one demand node.} \end{aligned}$$

This number corresponds to the number of maximum matchings in the bipartite graph mentioned above. For states with traveling repairmen it is again computed as an expectation.

The next theorem implies that the $\nu_0(s)$ is defined by a triple $(|\Omega(s)|,\ K_{covered}(s),\ \Theta(s))$.

Theorem 9.2.1. *For small values of discount rate β, the optimal policy for the large repair times regime can be described as follows:*

- *If a repairman can reach a broken machine in time, he should be dispatched.*
- *Among all possible actions always choose the one that leads to maximum $K_{covered}$ and then, if there are several action resulting in the same $K_{covered}$, to maximum Θ.*

Proof. Recall that we prefer to reach in time those machines that break down first. Each repairman can cover only one machine and the earlier he does it, the better. This implies the first statement of the theorem.

For small β, a machine from $\Omega(s)$ not reached in time adds almost 1 to $\nu_0(s)$. As only $K_{covered}$ machines can be reached in time from state s, other $|\Omega(s)| - K_{covered}(s)$ machines will cause penalties. So

$$\forall \epsilon > 0 \quad \exists \beta_\epsilon \ : \ \forall \beta < \beta_\epsilon \quad \nu_0(s) > |\Omega(s)| - K_{covered} - \epsilon.$$

On the other hand, if each repairman is assigned to a machine according to the maximum matching and waits for this particular machine to break down, then the covered machines will never cause a penalty, so

$$\nu_0(s) \leq |\Omega(s)| - K_{covered}.$$

Note that if all distances are bigger than one time unit and the process is currently in state s, then all possible next states s' have the same $\Omega(s')$. Also $K_{covered}(s')$ is determined by the current state and the action taken.

Consider two possible next states s_1 and s_2 with $K_{covered}(s_1) > K_{covered}(s_2)$. For small enough β it holds that

$$\nu_0(s_2) > |\Omega(s_2)| - K_{covered}(s_2) - (K_{covered}(s_1) - K_{covered}(s_2)) = |\Omega(s_1)| - K_{covered}(s_1) \geq \nu_0(s_1).$$

So it is beneficial to choose the action that leads to maximum $K_{covered}$ of the next state.

Consider now two possible next states s_1 and s_2 with the same $K_{covered}$. Then if $\Theta(s_1) > \Theta(s_2)$, the probability, that we can cover the machines that break down first, is higher for state s_1. As the earlier the machine breaks down, the bigger costs it causes, it implies that $\nu_0(s_1) < \nu_0(s_2)$.

If both $K_{covered}$ and Θ are the same for states s_1 and s_2, then the configurations of these states are the same and so $\nu_0(s_1) = \nu_0(s_2)$. □

Note that under the optimal policy the process never visits states s with both an uncovered machine from $\Omega(s)$ and idle repairmen at base stations from which they can not reach any machine from $\Omega(s)$ in time. It is also not optimal to visit a state where there is a machine in the queue for more than 1 time unit and an idle repairman that can reach it in time.

9.3 Large working time

To obtain the result for the regime with low rate of new failures occurrence, we multiply the probability of a break down p by ρ. Then from equations (9.1.1) for V, we get the following equations for ν_0:

$$\nu_0(s) = c(s) + e^{-\beta} \sum_{\mathcal{K}_2 \subset \mathcal{H}(s)} P\left(s, \emptyset, \mathcal{K}_2\right) \nu_0\left(\Phi\left(s, \pi(s), \emptyset, \mathcal{K}_2\right)\right) \tag{9.3.1}$$

Note that in state $\Phi\left(s, \pi(s), \emptyset, \mathcal{K}_2\right)$ the number of broken machines is always less or equal to the number of broken machines in state s and it is the same only if $\mathcal{K}_2 = \emptyset$. The same can be said for the number of unassigned calls.

If there are no traveling repairmen after taking action $\pi(s)$ in state s, then $\Phi\left(s, \pi(s), \emptyset, \emptyset\right) = s$ and equation (9.3.1) can be rewritten as

$$\nu_0(s) = \frac{1}{1 - P(s, \emptyset, \emptyset)} \left[c(s) + e^{-\beta} \sum_{\substack{\mathcal{K}_2 \subset \mathcal{H}(s) \\ \mathcal{K}_2 \neq \emptyset}} P\left(s, \emptyset, \mathcal{K}_2\right) \nu_0\left(\Phi\left(s, \pi(s), \emptyset, \mathcal{K}_2\right)\right) \right], \tag{9.3.2}$$

where the right part contains only the states with smaller number of broken machines. So if ν_0 is known for all states with smaller number of broken machines, $\nu_0(s)$ can be computed from this equation.

If there are traveling repairmen after taking action $\pi(s)$ in state s, but $\mathcal{M}_a(s) = \emptyset$, then the total distance left for all repairmen to travel to their current destination in state $\Phi\left(s, \pi(s), \emptyset, \emptyset\right)$ is less then in state s. So if for state s we know ν_0 for all states with either smaller number of broken

machines or the same number of broken machines and smaller total distance left to travel, then we can compute $\nu_0(s)$ from equation (9.3.1).

The only case in which the total remaining distance in state $\Phi(s, \pi(s), \emptyset, \emptyset)$ can be bigger than in state s is when there were some repairmen that arrived to their destinations (so $\mathcal{M}_a \neq \emptyset$) and then were dispatched to some calls. But in this case, the number of unassigned calls in state $\Phi(s, \pi(s), \emptyset, \emptyset)$ is less than in state s and, knowing ν_0 for all state with smaller number of unassigned calls, we can compute $\nu_0(s)$.

As a conclusion all states can be ordered, based on the number of broken machine, the number of unassigned calls and the total remaining distance to travel, in such a way that knowing ν_0 for all states before state s we can compute $\nu_0(s)$ from equation (9.3.1).

For states s with all machines working $\nu_0(s) = 0$ as there are no reasons for costs. So starting from them one can compute $\nu_0(s)$ and construct the optimal policy based on this values.

9.4 Small working and repair times

To consider this regime we multiply $(1 - p)$ and $(1 - q)$ by ρ. Then the equations for ν_0 take the following form:

$$\nu_0(s) = c(s) + e^{-\beta}\nu_0\left(\Phi\left(s, \pi(s), \mathcal{W}(s), \mathcal{H}(s)\right)\right). \tag{9.4.1}$$

So ν_0 is equal to the value function of the system where a machine always breaks down within the next time unit after repair and all repairs end within time unit after the beginning.

As the system is deterministic and contains no uncertainty, under the optimal policy each repairman has a route around the region. A route may consist of several demand nodes and base stations. For the systems with an excessive number of repairmen, a route may consist of only one base station where the repairman stays all the time. Traveling along his route a repairman stays for one time unit in each demand node of the route and from 0 to several time units in each base station of the route.

When a repairman leaves a demand node, a machine installed there breaks down in one time unit, so either he or another repairman should visit it during TL time units to avoid paying a penalty. So all penalties are associated with demand nodes that are visited more rarely than every TL time units.

Finding the optimal policy is equal to finding good routes for all repairmen. It is an interesting problem in itself for which we did not find any references in the research literature.

10 Conclusions

In this section we summarize the results of this thesis and discuss the final conclusions. We describe how different approaches to the problem the we studied relate to each other and formulate practical advice based on our results. In the second part of this section, we discuss the limitations of our study and propose directions for further research.

10.1 Results

Recall that our goal was to find a policy for real-time management of service engineers that leads to low costs associated with late arrivals to the calls. Moreover, we were not interested in finding such a policy for a particular service network, but rather to obtain an algorithm to construct this policy suitable for networks with different properties.

To tackle this problem, we proposed two models of the system: a continuous-time and a discrete-time ones. The continuous-time model provides the most accurate approximation to the real service network, whereas the discrete-time model is used for finding the exact optimal policy when possible.

First, we studied dispatching policy while the relocation policy was fixed to a static one (where relocation is not allowed). Four dispatching policies were compared by simulation for systems with different parameters: such as the number of repairmen, the average duration of the repair, the time limit and the map density. The policy that assigns the repairman with the smallest response time outperformed other policies for all systems. So, it may be beneficial not to dispatch an idle repairman immediately when a call arrives, but wait for another repairmen to finish the service and then dispatch him to the call.

We also compared the performance of this policy for two situations: when a repairman can estimate the duration of repair upon arrival to the failed machine and when the duration of the repair remain unknown until the repair is finished. We concluded that the accurate estimation of the remaining repair time is only important for the systems with large service times and high load.

Second, we compared the performance of four relocation policies. The first policy was the static policy, where relocation is not allowed. The second and third policies were compliance tables constructed according to different approaches. The fourth policy was a heuristic maximizing the expected covered demand for the system. These policies were combined with the dispatching policy that always chooses the repairman with the minimum response time. For most of the systems, both compliance tables showed worse performance than the static policy did. They outperformed it only for the systems where the distances are small compared to the average service time.

The performance of the fourth policy depends on the choice of the restriction parameters, such as the maximum distance of relocation and the minimum improvement of the expected covered demand. Without these restrictions for systems with large distances it performs worse than the static policy. However, when the restriction parameters are carefully chosen, it outperforms other three policies with up to 60% relative increase of the fraction of calls answered in time.

Next, we proposed the ADP approach to this problem. According to this approach the real costs were approximated by the combination of twelve basis functions. Two ways to tune this approximation were studied: an approximate policy iteration and a genetic algorithm. The approximate policy iteration showed poor performance, whereas the genetic algorithm resulted in a high-performing policy. For the considered example, this policy leads to on average 93% of calls answered in time compared to 89% answered in time by the heuristic policy.

Finally, we described a discrete-time model of the system for which the results of Markov Decision theory (such as policy iteration) were applied. We saw that even for a very small system of 4 demand nodes, this method is too computationally demanding to be used in practice.

Using this discrete-time model, we proved the results about the optimal policy for three extreme regimes. For the regime with large repair times, the optimal policy chooses the action that maximizes the number of covered working machines and the number of sets of working machines of the maximum size. For the regime with large working times, we did not find find the explicit

optimal policy. However, we proved that all the states can be ordered in such a way that the value function of a state is a linear combination of the value function of the previous states. Using this result, one can compute the value function for all states of the system and construct an optimal policy based on that. For the regime with small working times and small repair times, we discussed the structure of the optimal policy.

As a practical conclusion of our study, we advice to use either the heuristic policy (DMEXCLP + minimum response time dispatching) or the ADP approach. The ADP gives the best results of all the considered methods. However, it is computationally difficult, so if the computational resources are limited, one can use the heuristic policy, where the restriction parameters can be tunes relatively fast by means of simulation.

10.2 Future research

We suggest several directions in which our research can be extended.

First, one can relax our assumptions. The most restricting assumption is that preemption is not allowed. Once a repairman is relocated from one base station to another, he can not return to the first base station until after he reaches the latter. He also can not be dispatched directly to new calls even if he is traveling close to them. This restriction is not always close to practice and it leads to a cautious relocation strategy. As during relocation the repairman is not available, the relocation is not performed as often as it may be optimal.

Second, some of our approaches can be studied in more detail. For example, we observed that the performance of DMEXCLP heuristic policy heavily depends on the choice of the restriction parameters. But up to our knowledge, there are no studies on how to find the optimal values of these parameters for a given system. The ADP approach also allows further investigation. One can study how the choice of the basis functions, approach to finding the approximation and the parameters of the system affect the performance of this approach.

The third direction is to study more complex systems than we did. For example, in practice, when repairmen are not busy with corrective maintenance, they do preventive maintenance. The goal of the preventive maintenance is to avoid the failure of the machines by regular examination. It is usually performed according to some schedule. Building the preventive maintenance schedule that also takes into account the possibility of the demand for the corrective maintenance is one of the possible extensions of our study.

Another extension is to include spare parts management in the scope of the problem. Spare parts are often needed to perform the corrective maintenance. They are usually kept either in the storage or in the trucks of the repairmen. Making the decision about dispatching of a service engineer, the manager should also take into account the inventory level of spare parts needed for this repair. So to minimize the costs, it is beneficial to study the spare parts and the service engineers together. This is an interesting problem and, to our knowledge, there is no available research on this topic.

References

[1] Andersson, T., & Värbrand, P. (2007). Decision support tools for ambulance dispatch and relocation. *Journal of the Operational Research Society, 58*(2), 195-201.

[2] Van Barneveld, T. C., Bhulai, S., & Van der Mei, R. D. (2017). A dynamic ambulance management model for rural areas. *Health Care Management Science, 20*(2), 165-186. doi:10.1007/s10729-015-9341-3.

[3] Van Barneveld, T. C., Bhulai, S., & Van der Mei, R. D. (2016). The effect of ambulance relocations on the performance of ambulance service providers. *European Journal of Operational Research, 252*(1), 257-269.

[4] Van Barneveld, T. C. (2016). The minimum expected penalty relocation problem for the computation of compliance tables for ambulance vehicles. *Informs Journal on Computing, 28*(2), 370-384.

[5] Van Barneveld, T. C., Jagtenberg, C. J., Bhulai, S., & Van der Mei, R. D. (2018). Real-time ambulance relocation: Assessing real-time redeployment strategies for ambulance relocation. *Socio-Economic Planning Sciences, 62*, 129-142. doi:10.1016/j.seps.2017.11.001.

[6] Bélanger, V., Ruiz, A., & Soriano, P. (2018). Recent optimization models and trends in location, relocation, and dispatching of emergency medical vehicles. *European Journal of Operational Research*. doi:10.1016/j.ejor.2018.02.055.

[7] Benmerzouga, A., & Harous, S. (1999). Optimal (m-FailureP-Repairmen) policies with random repair time. *Stochastic Analysis and Applications, 17*(3), 327-338.

[8] Church, R. and ReVelle, C. (1974), The maximal covering location problem. *Papers in Regional Science, 32*: 101-118. doi:10.1111/j.1435-5597.1974.tb00902.x.

[9] Daskin, M. S. (1982). Application of an expected covering model to emergency medical service system design. *Decision Sciences 13*(3): 416–16.

[10] Daskin, M. S. (1983). A maximum expected covering location model: formulation, properties and heuristic solution. *Transportation Science 17*(1): 48–70.

[11] Derman, C. (1970). *Finite state markovian decision processes* (Mathematics in science and engineering, vol. 67). New York: Academic Press.

[12] Efrosinin, D., Spannring, C., & Sztrik, J. (2014). Optimal allocation problem in the machine repairman system with heterogeneous servers. *Communications in Computer and Information Science, 487*(487), 113-122.

[13] Gendreau, M., Laporte, G., & Semet, F. (1997). Solving an ambulance location model by tabu search. *Location Science, 5*(2), 75-88. doi:10.1016/S0966-8349(97)00015-6.

[14] Gendreau, M., Laporte, G., & Semet, F. (2001). A dynamic model and parallel tabu search heuristic for real-time ambulance relocation. *Parallel Computing, 27*(12), 1641-1653. doi:10.1016/S0167-8191(01)00103-X.

[15] Gendreau, M., Laporte, G., & Semet, F. (2006). The maximal expected coverage relocation problem for emergency vehicles. *The Journal of the Operational Research Society, 57*(1), 22-22. doi:10.1057/palgrave.jors.2601991.

[16] Van Houtum, G.-J. J. A. N., & Kranenburg, B. (2015). *Spare parts inventory control under system availability constraints* (International series in operations research & management science, volume 227). New York: Springer. doi:10.1007/978-1-4899-7609-3.

[17] Jagtenberg, C. J., Bhulai, S., & Van der Mei, R. D. (2017). Dynamic ambulance dispatching: Is the closest-idle policy always optimal? *Health Care Management Science, 20*(4), 517-531. doi:10.1007/s10729-016-9368-0.

[18] Jagtenberg, C. J., Bhulai, S., & Van der Mei, R. D. (2015). An efficient heuristic for real-time ambulance redeployment. *Operations Research for Health Care, 4*, 27-35.

[19] Jagtenberg, C. J., Van den Berg, P. L., & Van der Mei, R. D. (2017). Benchmarking online dispatch algorithms for emergency medical services. *European Journal of Operational Research, 258*(2), 715-725. doi:10.1016/j.ejor.2016.08.061.

[20] Katehakis, M. N., & Derman, C. (1984). Optimal repair allocation in a series system. *Mathematics of Operations Research, 9*(4), 615-623. Retrieved from http://www.jstor.org/stable/3689466.

[21] Katehakis, M. N., & Levine, A. (1986). Allocation of distinguishable servers. *Computers & Operations Research 13*(1): 85–85. doi: 10.1016/0305-0548(86)90066-3.

[22] Larson, R. C. (1974). A hypercube queuing model for facility location and redistricting in urban emergency services. *Computers and Operations Research, 1*(1), 67-95. doi:10.1016/0305-0548(74)90076-8.

[23] Larson, R. C. (1975). Approximating the performance of urban emergency service systems. *Operations Research 23*(5): 845–45.

[24] Li, X., Zhao, Z., Zhu, X., & Wyatt, T. (2011). Covering models and optimization techniques for emergency response facility location and planning: A review. *Mathematical Methods of Operations Research, 74*(3), 281-310. doi:10.1007/s00186-011-0363-4.

[25] Maxwell, M. S., Restrepo, M., Henderson, S. G., & Topaloglu, H. (2010). Approximate dynamic programming for ambulance redeployment. *Informs Journal on Computing, 22*(2), 266-281. doi:10.1287/ijoc.1090.0345.

[26] Murthy, D. N. P., Solem, O., & Roren, T. (2004). Product warranty logistics: Issues and challenges. *European Journal of Operational Research, 156*(1), 110-126.

[27] Murthy, D. N. P., & Jack, N. (2014). *Extended warranties, maintenance service and lease contracts: modeling and analysis for decision-making* (Springer series in reliability engineering). London: Springer. doi:10.1007/978-1-4471-6440-1.

[28] Nasrollahzadeh, A. A., Khademi, A., & Mayorga, M. E. (2018). Real-time ambulance dispatching and relocation. *Manufacturing & Service Operations Management, 20*(3), 467-480. doi:10.1287/msom.2017.0649.

[29] Schmid, V. (2012). Solving the dynamic ambulance relocation and dispatching problem using approximate dynamic programming. *European Journal of Operational Research, 219*(3), 611-611.

[30] ReVelle, C., & Hogan, K. (1989). The maximum availability location problem. *Transportation Science, 23*(3), 192-200. Retrieved from http://www.jstor.org/stable/25768379.

[31] Yu, X., & Gen, M. (2010). *Introduction to evolutionary algorithms* (Decision engineering). London: Springer. doi:10.1007/978-1-84996-129-5.

[32] Zhang, O., Mason A. J., & Philpott A. B.. *Simulation and optimization for ambulance logistics and relocation.* Presented at the INFORMS 2008 Conference, 2008.