Vrije Universiteit Amsterdam

Master Thesis Business Analytics

Scheduling patients at the outpatient clinic

An approximate dynamic programming approach

Author:
Else Wolfswinkel

Supervisor HOTflo:
Dennis Roubos

Supervisor VU:
René Bekker

February 2017

Vrije Universiteit Amsterdam

Master Thesis Business Analytics

Scheduling patients at the outpatient clinic

An approximate dynamic programming approach

Author:
Else Wolfswinkel

Supervisor HOTflo:
Dennis Roubos

Supervisor VU:
René Bekker

Vrije Universiteit Amterdam
Faculty of Sciences
De Boelelaan 1081
1081 HV Amsterdam

HOTflo Company
Bisonspoor 5000
3605 LT Maarssen

February 2017

Preface

This thesis is written to conclude the final stage of the Master programm Business Analytics at the Vrije Universiteit Amsterdam. It contains the results of my research performed at HOTFlo, a company that is a specialised service provider in the field of integrated capacity management in hospitals.

For this research I had to immerse myself in Markov Decision Processess (MDPs) and Approximate Dynamic Programming (ADP). It was a challenging but instructive task, since ADP was a completely new area to me. Fortunately, my supervisor from HOTFlo, Dennis Roubos, knows a lot about MDPs and ADP and I would really like to thank him for his comprehensive and very clear explanation on these topics. He provided me with a lot of insights. I also would like to thank René Bekker, my supervisor from the Vrije Universiteit Amsterdam, for thinking along with my research, for his useful recommendations and for giving me structure, especially when I started this research. Finally, I would like to thank Bram Gorissen for being my second reader.

Else Wolfswinkel
Hilversum, February 2017

Abstract

In this research we develop a model that prescribes the (near) optimal appointment date for a patient at the moment this patient makes his request at the outpatient clinic. In order to differentiate between different types of patients, such as new patients, control patients and several grades of priorities, we distinguish two categories of patients. One category is characterised by a maximum recommended waiting time. Since the actual scheduled appointment time is reflected in costs, the sooner these patients are scheduled the lower the costs and when the maximum recommended waiting time is exceeded extra costs are incurred. The other category is characterised by a specific appointment time. The closer the scheduled appointment time is to the specific appointment time the lower the costs. The objective is to minimise the long-run expected average cost. We model the scheduling process as a Markov Decision Process (MDP) and we solve this MDP to optimality using value iteration. However, due to the curse of dimensionality, it is computationally infeasible to solve our MDP to optimality for the scheduling process over more than three working days and a reasonable available capacity on a day. To overcome the problem of the curse of dimensionality we apply the Bellman Error Minimisation (BEM) method as an Approximate Dynamic Programming technique in order to derive an estimate of the optimal value function of our MDP of which the near optimal policy (appointment date) can be derived. To determine the set of representative states, which is an element of the BEM method, we use the Hartigan and Wong k-means algorithm. We test several approximation functions and find an approximation function that outperforms all other functions in the scheduling process over three, five, ten and twenty working days. In general it holds that the higher the arrival rate of patients at the outpatient clinic, the better our BEM method performs. But if the arrival rate reaches a certain value the load of the system becomes that high that it does not matter what policy is applied, since many patients have to be rejected.

Contents

1 Introduction

1.1 Background information

Many hospitals face the problem of scheduling patients at an outpatient clinic. An essential tradeoff in these problems is between using the available capacity in the short term with a low priority patient or schedule this patient later and the capacity remains available for a high priority patient. The risk is that the high priority patient does not presents itself and the capacity is not utilised. In addition to the tradeoff between the various types of priorities, the difference between new patients and control patients must also be taken into account. New patients are patients who come to the outpatient clinic for the first time and control patients are patients that are seen on a regular basis by a specialist. When and how to schedule new patients cannot be seen independently from scheduling control patients, since new patients bring forth demand for control appointments. For example, scheduling one week with only new patients might be problematic since that generates enormous demand for control appointments later on. Next to that, there are several other factors involved when scheduling patients. Patients might have their own preferences, like a preference for a specialist or a preference for a specific day or time for the appointment. Furthermore, patients may not show up or cancel their appointment. In this case, patients need to be rescheduled or leave the system.

Because so many factors need to be taken into account when scheduling patients, outpatient clinics often fail to schedule their patients in an efficient manner. This could lead to long access times for patients, which is the time between the request and the actual appointment time, and low utilisation levels on some days and overload situation on other days, yielding idle time or overtime for specialists.

1.2 Problem description

HOTflo Company is a specialised service provider in the field of integrated capacity management in hospitals. They employ effective change programs combined with intelligent software and high-quality education and training programs in order to support hospitals in optimising the care they provide, and in structurally reducing costs. Because hospitals encounter many difficulties in scheduling appointments at the outpatient clinic, HOTflo wants to meet the needs of hospitals in here. This, together with the progressive digitising of society, HOTflo aims to develop an online application in which patients can schedule their own appointment at the outpatient clinic. The idea of this application is as follows. After a patient has opened the application and answered some questions, for instance, if the patient comes to the outpatient clinic for the first time or if the patient has a preference for his own specialist, the application should display some appointment date and time op-

tions of which patients can choose their own appointment date and time. The underlying algorithm of the application must ensure that the way patients are scheduled is done in an efficient manner, meaning short access times for patients and a stable utilisation for specialists during the days.

This research is a first step in the development of this algorithm. The goal is to develop a method that prescribes the (near) optimal appointment date for a patient at the moment this patient makes his request. In order to differentiate between different types of patients such as new patients, control patients and several grades of priorities, we distinguish two categories of patients. One category is characterised by a maximum recommended waiting time. Since the actual scheduled appointment time is reflected in costs, the sooner these patients are scheduled the lower the costs and when the maximum recommended waiting time is exceeded extra costs are incurred. Among this category new patients and different grades of priority patients can be considered. The other category is characterised by a specific appointment time. Among this category patients who want to make an appointment in advance, like control patients, can be considered. The closer the scheduled appointment time is to the specific appointment time the lower the costs. The objective is to minimise the long-run expected average cost. We first formulate the scheduling process as a Markov Decision Process (MDP), since scheduling patients at an outpatient clinic is a decision problem with sequential decisions that must be made under uncertainty and MDPs provide a mathematical framework for modeling these decision problems. An MDP can be solved to optimality, using one of the standard solution methods, like value iteration. This results in an optimal value function of which the optimal appointment date can be derived. However, due to the curse of dimensionality, it becomes computationally infeasible to solve our MDP. Therefore, we employ an Approximate Dynamic Programming (ADP) technique, in order to derive an estimate of the optimal value function of our MDP.

1.3 Outline

The remainder of this thesis is organised as follows. In Chapter 2 a review of the literature on scheduling patients at the outpatient clinic is presented. In Chapter 3 and 4 we give a short introduction to MDPs and ADP. These chapter provides the reader with some background information about these topics. The reader who is familiar with MDPs and/or ADP can skip these chapters without loss of continuity. In Chapter 5 we formulate the scheduling process as an MDP and we describe how we applied ADP to our MDP. We describe the results and their interpretations of our research in Chapter 6. Finally, this thesis ends with some conclusions and we give possible and/or necessary extensions to our model in Chapter 7.

2 Literature review

In this chapter we describe the findings from our literature research. The purpose of our literature research was twofold. First, we wanted to get an idea about what is already done in the field of scheduling patients at the outpatient clinic. Second, we wanted to get some inspiration and ideas for our MDP.

One way of classifying literature about scheduling patients at the outpatient clinic is according to the type of waiting patients have to deal with: intra-day scheduling and inter-day scheduling. Intra-day scheduling is about the best timing and sequence of appointments within a given day and concerns direct waiting times, the time the patient spends waiting on the day of his appointment. Inter-day scheduling considers a multi-day planning horizon and is about how to allocate appointment requests arising on the current day into future days and concerns indirect waiting times. The indirect waiting time is the time between the day the patient makes his request and the actual appointment day. The key issues in designing and managing patient appointment systems for health services are discussed by Gupta and Denton [7]. They consider both intra-day and inter-day scheduling problems and state that most of the existing literature is concentrated on intra-day scheduling. Since our work focuses on inter-day scheduling and does not address how intra-day scheduling needs to be done, we briefly discuss literature on intra-day scheduling and we mainly focus on the literature about inter-day scheduling.

2.1 Intra-day scheduling

In 2003 Cayirli and Veral [2] present a comprehensive overview of research on appointment scheduling in outpatient services. They summarise the relevant factors that are encountered in appointment scheduling, such as number of doctors, presence of no-shows, service time distribution and so on. Next to that, they summarise the performance measures used to evaluate appointment systems. They distinguish five categories:

- Cost-based measures;
- Time-based measures;
- Congestion measures;
- Fairness measures;
- Other

of which cost-based measures, like costs for patients direct waiting time, doctor's idle time and doctor's overtime, are most commonly used. Until then, little research took into account the probability of no-shows and walk-ins (patients who arrive without having an appointment, like emergency patients). Subsequently, no shows and walk-ins received more attention, as for example in [3, 9, 10]. Kaandorp and Koole [9] derived a local search procedure, which can incorporate no-shows, to optimise a weighted average of expected

waiting times of patients and idle time and overtime of the doctor. More recently, Koeleman and Koole [10] used a generalisation of this local search procedure with the same objective in a setting where emergency arrivals occur according to a non-stationary Poisson process and where service times can have any given distribution. Cayirli et al. [3] introduces a universal "Dome" appointment rule for finding the appointment times for a given clinic defined by no-shows, walk-ins, desired number of appointments per session, coefficient of variation of service times, and the relative value of doctor's time to patients' time.

2.2 Inter-day scheduling

As mentioned before, literature on inter-day scheduling problems is scarce, but has gained more attention in recent years. Several optimisation or approximation techniques are used for these kind of problems each taking into account other relevant factors. Gupta and Wang [8] developed an MDP for the scheduling process to determine whether or not to admit or when to schedule incoming appointment requests and maximises the revenue obtained on a given day. In their model they take into account the patients' specialist and appointment time preferences. Next to that, they develop five heuristics as well as an upper bound on the optimal revenue. Liu et al. [12] were the first to propose dynamic appointment scheduling policies using a model that takes patient no shows and cancellations into account. They use the idea of applying a single step of the policy improvement algorithm to develop a heuristic method. Feldman et al. [6] developed an appointment scheduling model based on an MDP, which can be used in making appointment scheduling decisions while taking into account both patients' preferences for different appointment days and their no-show and cancellation behavior. Another approach is proposed by Erdogan et al. [5]. They formulate and solve a two-stage stochastic mixed integer programming model for dynamic sequencing and scheduling of appointments to a single stochastic server. Binary decision variables are used to represent the patient sequencing decisions and continue variables represent the inter-arrival times and appointment times.

Most of the literature described above, modelled the scheduling process as an MDP, but we found two papers in the literature that are particularly interesting to our work. Patrick et al. [13] developed an MDP model for dynamically scheduling multi-priority patients. Each patient priority class is characterised by a maximum recommended waiting time and their goal is to minimise the number of patients who does not get an appointment by their maximum wait time target. To deal with the curse of dimensionality they transform their MDP into the related linear program and use ADP techniques to produce an approximate linear program which they solve in order to derive the estimate of the optimal value function. Erdelyi and Topaloglu [4] consider a general capacity allocation problem in a setting with a fixed amount of daily processing capacity. Jobs of different priorities arrive randomly over time and it must be decided which jobs should be scheduled for which days. Jobs that are waiting to be processed incur a holding cost depending on their priority levels. The objective is to find a job scheduling policy that minimises the total expected

cost over a finite planning horizon. They formalise their problem as an MDP which is quite similar to the MDP formulated by Patrick et al. [13]. But they use an ADP techniques that decomposes their MDP to deal with the curse of dimensionality.

The MDP we developed is inspired by the models described by Patrick et al. [13] and Erdelyi and Topaloglu [4]. However, both models observe the arrivals of patients/jobs at the beginning of each day, which means that several patients/jobs need to be scheduled at once. In reality patients are scheduled immediately at the moment they make their request. We want our model to be as close as possible to reality and we model this explicitly in our MDP. Next to that, we also use a different ADP technique to estimate the optimal value function than Patrick et al. [13] and Erdelyi and Topaloglu [4]. Instead of transforming our model into a linear program or decomposing our MDP, we apply the BEM method.

3 Markov Decision Processess

Markov Decision Processess (MDPs) provide a mathematical framework for modeling decision problems with sequential decisions that must be made under uncertainty [1, 15]. Scheduling patients in an outpatient clinic is such a decision problem. When a patient makes a request for an appointment, the planner (decision maker) observes the current schedule and makes a decision, e.g. schedule the patient on a given day and time. The consequence of the decision is reflected in costs, therefore for every decision taken, immediate costs might be incurred. For instance, if the patient has to be scheduled within two days, but the first two days of the schedule are fully occupied, cost might be incurred for not being able to met the requirements of the patient. The decision of the planner will create a new schedule (the old schedule with an additional appointment) and when the next patient makes his request for an appointment the planner faces the same problem with this new schedule. Because decisions are sequentially made over time, the goal is to find a strategy for the decision maker, such that applying this strategy minimises a cost criterion. A strategy is also called a policy and the optimal policy is obtained by solving the corresponding MDP.

In this chapter we give a short introduction to MDPs and how they can be solved. In Section 3.1 the basic components of MDPs are described and corresponding terms such as value functions and optimality equations are explained. Section 3.2 describes a well-known algorithm for computing an optimal MDP policy. Aperiodicity and the curse of dimensionality, two issues of MDPs that are important to our research, are discussed in Section 3.3 and 3.4. For more information about MDPs we refer to Puterman [15] and/or Tijms [17], where they are extensively described.

3.1 Introduction to Markov Decision Processes

An MDP is defined by four basic components:

- The state space $\mathcal{X}$, a set of states;
- The action space $\mathcal{A}_x$, a set of actions for each state $x \in \mathcal{X}$;
- A cost function $c(x, a)$, describing the incurred costs when taking action a in state x;
- A transition probability function $p(x, a, y)$, describing the probability that the system will be in state y when taking action a in state x, with $p(x, a, y) \in [0, 1], x, y \in \mathcal{X}, a \in \mathcal{A}_x$ and $\sum_{y \in \mathcal{X}} p(x, a, y) = 1, x \in \mathcal{X}, a \in \mathcal{A}_x$.

Although general MDPs may have infinite or even uncountable state and action spaces, we assume the state space $\mathcal{X}$ and the action space $\mathcal{A}_x$ to be finite. The decision maker chooses an action based on the current state of the system and should take into account the future behaviour of the system. As a result of the chosen action the system makes a transition to another state and the system receives immediate costs. The goal is to find an

optimal policy π^* such that a cost criterion is minimised. As cost criterion we focus on the long-run expected average cost, denoted as g. Other cost criteria are the expected total cost and the expected total discounted cost [17]. Next to the long-run expected average cost as cost criterion, we focus on the class of stationary policies. A stationary policy is a policy that assigns to each state a fixed action and every time the system is in this state the decision maker chooses this action.

Value function, Poisson equations and optimality equations

Value functions are used for the evaluation of a policy π in the set of available policies Π and therefore play a crucial role in determining the optimal policy. The value of a state x under policy π, denoted as $V_\pi(x)$, represents the long-run expected average cost when starting in x and following policy π thereafter. The value function V_π for a fixed policy is the solution of the set of Poisson equations:

$$V_\pi(x) + g_\pi = c(x, \pi(x)) + \sum_{y \in \mathcal{X}} p(x, \pi(x), y) V_\pi(y), \qquad x \in \mathcal{X}, \tag{3.1}$$

where $\pi(x)$ is the action chosen in state x under policy π. The goal is to find the optimal policy π^*, i.e. the policy that minimises g, denoted by g^*. For this policy it holds that $V_{\pi^*}(x) \leq V_\pi(x), x \in \mathcal{X}, \pi \in \Pi$. The optimal solution $V^* = V_{\pi^*}$ satisfies the following equations:

$$V^*(x) + g^* = \min_{a \in \mathcal{A}_x} \{c(x, a) + \sum_{y \in \mathcal{X}} p(x, a, y) V^*(y)\}, \qquad x \in \mathcal{X}. \tag{3.2}$$

These equations are also known as the optimality equations or Bellman equations. The solution of (3.2) is not unique. If (V^*, g^*) is a solution, then $(V^* + c, g^*)$ is also a solution. A unique solution can be obtained by taking $V^*(0) = 0$. Then $V^*(x)$ is the difference in accrued cost when starting the process in state x relative to starting in state 0. Once the optimal value function V^* is computed, the optimal policy π^* can be obtained by:

$$\pi^*(x) = \arg\min_{a \in \mathcal{A}_x} \{c(x, a) + \sum_{y \in \mathcal{X}} p(x, a, y) V^*(y)\}, \qquad x \in \mathcal{X}.$$

There are three well-known algorithms for computing the optimal value function and the corresponding policy:

- Policy Iteration;
- Linear Programming;
- Value Iteration.

Policy iteration and linear programming both require the solving of a system of linear equations in each iteration step. Value iteration computes recursively a sequence of value functions. We describe the algorithm of value iteration in the next section, because we apply this technique to the scheduling process. For more information about policy iteration or linear programming we refer to Puterman [15] or Tijms [17].

Algorithm 1 Value iteration

procedure VALUE ITERATION($\mathcal{X}, \mathcal{A}_x, P, c, \epsilon$)
Inputs: $\mathcal{X}$: the set of all states with $|\mathcal{X}| = N$, $\mathcal{A}_x$: the set of all actions, P: a transition probability matrix specifying $p(x, a, y)$, c: a cost function $c(x, a)$, ϵ: a threshold > 0
Outputs: V: value function, π^*: optimal policy, g^*: minimal long-run average cost per time unit
Local variables: int n, **double** min, max, **Vector** $V(0, \ldots, N), \pi(0, \ldots, N)$
    $V \leftarrow 0$
    $n \leftarrow 0$
    do
        $min = \infty$
        $max = -\infty$
        for each state $x \in \mathcal{X}$ **do**
            $V_{n+1}(x) = \min_{a \in \mathcal{A}_x}\{c(x, a) + \sum_{y \in \mathcal{X}} p(x, a, y)V_n(y)\}$
            $min = \min\{min, V_{n+1}(x) - V_n(x)\}$
            $max = \max\{max, V_{n+1}(x) - V_n(x)\}$
        end for
        $n = n + 1$
    while $max - min > \epsilon$
    $g^* = (min + max)/2$
    for each state $x \in \mathcal{X}$ **do**
        $\pi^*(x) = \arg\min_{a \in \mathcal{A}_x}\{c(x, a) + \sum_{y \in \mathcal{X}} p(x, a, y)V_n(y)\}$
    end for
end procedure

3.2 Value iteration

The value iteration algorithm is displayed in pseudo code in Algorithm 1. It starts with the value function $V_0(x) = 0, x \in \mathcal{X}$. Then the value function is updated iteratively by:

$$V_{n+1}(x) = \min_{a \in \mathcal{A}_x}\{c(x, a) + \sum_{y \in \mathcal{X}} p(x, a, y)V_n(y)\}, \qquad x \in \mathcal{X}.$$

and for each $x \in \mathcal{X}$ the bounds on the minimal costs are computed. These bounds can also be computed at the end of each iteration:

$$\text{minBound}_{n+1} = \min\{V_{n+1}(x) - V_n(x)\}, \qquad \text{maxBound}_{n+1} = \max\{V_{n+1}(x) - V_n(x)\}.$$

If after an iteration maxBound $-$ minBound $< \epsilon$, with $\epsilon > 0$ a prespecified accuracy number, then the value function is converged to its optimum and g^* is obtained by (minBound + maxBound)/2. However, the minBound and maxBound only converge to the same limit (g^*) under certain conditions. One of these conditions is that for each average cost optimal stationary policy the associated Markov decision chain is aperiodic. Since

this is an important condition for our study, we go deeper into aperiodicity in the next section, for more information about the convergence of the bounds we refer to Puterman [15] and Tijms [17].

3.3 Aperiodicity

A Markov decision chain is called aperiodic if for every policy π the greatest common divisor of the length of all paths from x to x equals 1, for some recurrent state $x \in \mathcal{X}$. State x is called recurrent if there is a path from x to x [17]. If for some policy π the greatest common divisor of the length of all paths from x to x is strictly larger than 1 for some recurrent state $x \in \mathcal{X}$, the Markov decision chain is periodic and value iteration does not need to converge. This aperiodicity condition is not restrictive, since all policies can be made aperiodic by a simple data transformation:

$$p(x,a,y) = \begin{cases} \gamma p(x,a,y), & \text{if } y \neq x, a \in \mathcal{A}_x, x \in \mathcal{X}, \\ \gamma p(x,a,y) + (1-\gamma), & \text{if } y = x, a \in \mathcal{A}_x, x \in \mathcal{X}, \end{cases}$$

with $0 < \gamma < 1$. Thus, with probability γ the transition to another state occurs according to the original transition probabilities and with an extra probability $1-\gamma$ the process stays in the current state, irrespective of the state and action. If (V^*, g^*) is a solution of (3.2), then $(V^*/\gamma, g^*)$ is a solution of

$$V^*(x) + g^* = \min_{a \in \mathcal{A}_x} \{c(x,a) + \gamma \sum_{y \in \mathcal{X}} p(x,a,y) V^*(y) + (1-\gamma) V^*(x)\}, \qquad x \in \mathcal{X}. \tag{3.3}$$

Thus, the long-run expected average cost remain the same and V^* is multiplied by the inverse of this constant.

3.4 Curse of dimensionality

What certainly indicates a problem is the so-called curse of dimensionality [14]. Standard methods to obtain the value function, such as value iteration suffer from the curse of dimensionality, which means that as the state and/or action space increases, it becomes computationally infeasible to solve an MDP. Suppose there is a model with a multi-dimensional state space $\mathcal{X} = \{(x_1, \ldots, x_n) | x_i = 1, \ldots, C\}$. Then the number of different states is $|\mathcal{X}| = C^n$, which is already large for relative small values of n and C. Since it becomes computationally infeasible to solve an MDP for large state and/or action spaces it is not possible to compute the optimal value function and to obtain the optimal policy. Approximate Dynamic Programming (ADP) is a method to deal with the curse of dimensionality. The central idea of ADP is extensively discussed in the next chapter.

4 Approximate Dynamic Programming

As stated in the previous chapter the value function is crucial in deriving optimal policies. However, techniques to compute the value function, such as value iteration, become computational infeasible due to the curse of dimensionality. Approximate Dynamic Programming (ADP) is a method to overcome the problem of the curse of dimensionality. In this chapter we describe the central idea behind ADP in Section 4.1. Thereafter, we discuss and compare two ADP methods on the basis of an $M/M/s$ queue. The $M/M/s$ queue we consider is described in Section 4.2. The first ADP method we discuss in Section 4.3 is the Bellman Error Minimisation method. The second ADP method is Approximate Value Iteration which we discuss in Section 4.4. For a thorough understanding of ADP, we actually applied it to several $M/M/s$ queues. We compare both ADP methods and the value iteration method with each other based on the results of the $M/M/s$ queues. The results of this comparison are given in Section 4.5. We end this chapter with an overview of advantages and disadvantages of all the methods in Section 4.6 and discuss which ADP technique we apply to our scheduling process.

4.1 Introduction in ADP

The central idea of ADP is to approximate the true value function $V(x)$ by an approximation function $\tilde{V}(x,r) = \sum_{i=1}^{R} r_i \phi_i(x)$, with r a vector of parameters and $\phi_i(x)$ a set of basis functions that is used for the approximation. The choice of the basis functions depends on the specifics of the problem and is therefore more an art than exact science [16]. After the basis functions are chosen, the goal is to find the optimal parameter vector r such that the difference between the true value function and the approximation function is minimised. One advantage of this approximation function is that only $|r|$ values have to be stored, instead of $|\mathcal{X}|$ values from the value function. A second advantage is that a change in r affects the value for all possible states and hence, only a representative subset of the states $\tilde{\mathcal{X}} \subset \mathcal{X}$ has to be considered to determine the vector r. For instance, the choice of the set of representative states could be to include only the states that have a high probability of being visited. As a result of both advantages, ADP suffers less from the curse of dimensionality. In the next sections we describe two ADP methods for finding the optimal parameter vector r on the basis of an $M/M/s$ queue and we compare the results with each other.

4.2 M/M/s Queue

To become familiar with ADP we apply it first to the $M/M/s$ queue. In the $M/M/s$ queue arrivals occur according to a Poisson process and form a single queue, there are s servers and service times are exponentially distributed. We consider the $M/M/s$ queue with two

types of patients and admission control, thus arriving patients may be rejected. Both types of patients have the same average service time, but differ in the cost per rejected patient. The objective is to minimise the average number of patients in the system and the number of rejected patients. This queue is modelled as an MDP as follows:

State space: $\mathcal{X} = \{0, 1, 2, 3, ...\}$ representing the total number of patients in the system.

Action space: $\mathcal{A} = \{0, 1\}$ representing admission and rejection.

Cost function: There are direct costs x if there are x customers in the system and rejection cost c_i if a patient of type $i \in \{1, 2\}$ is rejected.

Transition probabilities: Patients of type i arrive according to a Poisson process with rate λ_i, $i \in \{1, 2\}$ and leave the system after they are being served during an exponentially distributed amount of time with rate μ. By uniformization, these transition rates are scaled such that $\sum_i \lambda_i + \mu s \leq 1$.

The optimality equations for this queue are given by:

$$\begin{aligned} V(x) + g = x &+ \sum_{i=1}^{2} \lambda_i \min\{V(x+1); V(x) + c_i\} + \mu \min\{x, s\} V((x-1)^+) \\ &+ (1 - \sum_{i=1}^{2} \lambda_i - \mu \min\{x, s\}) V(x), \end{aligned} \tag{4.1}$$

where $(x)^+ = \max\{0, x\}$. The two methods of ADP we discuss next, are explained on the basis of an $M/M/3$ queue with parameters $\lambda_1 = 0.15, \lambda_2 = 0.1, \mu = 0.1, c_1 = 20, c_2 = 25$, with approximation function $\tilde{V}(x, r) = r_1 x + r_2 x^2$ and with $\tilde{\mathcal{X}} = \{0, 1, 2, 3, 4\}$. It should be mentioned that $\tilde{\mathcal{X}}$ in this case is chosen small in order to simplify the equations when explaining both ADP methods.

4.3 Bellman Error Minimisation

In the Bellman Error Minimisation (BEM) method the goal is to minimise the error which is made in the Poisson equations (3.1) by using the approximation function $\tilde{V}(x, r)$ instead of the real value function $V(x)$. When the real value function is used, the Bellman error, given by:

$$D(x) = -g - V(x) + c(x, \pi(x)) + \sum_{y \in \mathcal{X}} p(x, \pi(x), y) V(y),$$

equals 0 for all $x \in \mathcal{X}$. However, when instead of the real value function $V(x)$ the approximation function $\tilde{V}(x, r)$ is used, the Bellman error is given by:

$$D(x, r) = -g - \tilde{V}(x, r) + c(x, \pi(x)) + \sum_{y \in \mathcal{X}} p(x, \pi(x), y) \tilde{V}(y, r). \tag{4.2}$$

The goal is to find a good approximation for the most important part of the state space. This goal can be achieved by minimising the weighted sum of squared Bellman errors for a set of representative states $\tilde{\mathcal{X}} \subset \mathcal{X}$:

$$\min_r \{E(r) = \sum_{x \in \tilde{\mathcal{X}}} w(x) D^2(x, r)\}, \tag{4.3}$$

where the weight $w(x)$ is used to emphasise the importance of the error made in state $x \in \tilde{\mathcal{X}}$. The minimum is determined by calculating the partial derivatives of $E(r)$ with respect to $r_i, i = 1, ..., R$ and setting them to zero. This results in a system of R equations with R unknowns which can be solved.

Note that in order to determine the Bellman error (4.2), an initial policy must be chosen in MDPs where actions are involved. The initial policy we choose in our $M/M/3$ queue is to admit every patient. The Bellman error of our queue is then given by:

$$\begin{aligned} D(x, r) &= -g - \tilde{V}(x, r) + x + 0.15\tilde{V}(x+1, r) + 0.1\tilde{V}(x+1, r) \\ &\quad +0.1 \min\{x, 3\}\tilde{V}((x-1)^+, r) + (1 - 0.25 - 0.1 \min\{x, 3\})\tilde{V}(x, r) \\ &= -g + x + 0.25\tilde{V}(x+1, r) \\ &\quad +0.1 \min\{x, 3\}\tilde{V}((x-1)^+, r) - (0.25 + 0.1 \min\{x, 3\})\tilde{V}(x, r). \end{aligned} \tag{4.4}$$

Now we need to determine g, which can be done in two ways: by simulation or by using the fact that we can set $V(0) = 0$, see Section 3.1. However, g obtained by simulation is preferred, provided that the simulation runs long enough. This g is more accurate than the g obtained by setting $V(0) = 0$. If g is obtained by simulation, then g in (4.4) is replaced by a constant. If we set $V(0) = 0$, then by using the optimality equations (4.1) and our policy to admit every patient, $g = 0.15V(1) + 0.1V(1) = 0.25V(1)$. Inserting g obtained by setting $V(0) = 0$ and inserting our approximation function into the Bellman error gives:

$$\begin{aligned} D(x, r) = &- 0.25[r_1 + r_2] + x + 0.25[r_1(x+1) + r_2(x+1)^2] \\ &+ 0.1 \min\{x, 3\}[r_1(x-1)^+ + r_2((x-1)^+)^2] \\ &- (0.25 + 0.1 \min\{x, 3\})[r_1 x + r_2 x^2]. \end{aligned}$$

If we set the weight function $w(x) = 1, x \in \tilde{\mathcal{X}}$, the sum of squared Bellman errors is given by:

$$\begin{aligned} E(r) = \textstyle\sum_{x \in \tilde{\mathcal{X}}} D^2(x, r) &= 0^2 + (-0.1r_1 + 0.4r_2 + 1)^2 + (-0.2r_1 + 0.4r_2 + 2)^2 \\ &\quad +(-0.3r_1 + 3)^2 + (-0.3r_1 - 0.1r_2 + 4)^2 \\ &= 0.23r_1^2 + 0.33r_2^2 - 0.18r_1 r_2 - 5.2r_1 + 1.6r_2 + 30. \end{aligned}$$

By taking partial derivatives of $E(r)$ with respect to r_1 and r_2 and setting them to zero we obtain two equations with two unknowns:

$$\frac{\partial}{\partial r_1}\sum_{x\in\tilde{\mathcal{X}}} D^2(x) = 0.46r_1 - 0.18r_2 - 5.2 = 0,$$
$$\frac{\partial}{\partial r_2}\sum_{x\in\tilde{\mathcal{X}}} D^2(x) = -0.18r_1 + 0.66r_2 + 1.6 = 0.$$

Solving the above equations, we obtain $r_1 = 11.59$ and $r_2 = 0.74$, and hence $\tilde{V}(x) = 11.59x + 0.74x^2$. Now that we found the approximate value function $\tilde{V}(x)$ a new policy is obtained and the performance of this policy can be compared with the performance of the initial policy. This is called one-step policy improvement. If necessary the BEM can be repeated with the new policy until the results are satisfying. However, policy improvement gives the biggest improvement during the first steps [17], which we show in Section 4.5.

4.4 Approximate Value Iteration

Approximate Value Iteration (AVI) is an algorithm which uses the approximation function $\tilde{V}(x,r)$ to estimate the value function $\hat{V}(x)$ for the set of representative states $\tilde{\mathcal{X}}$ at each iteration in the value iteration algorithm, see Algorithm 1. Thus,

$$\hat{V}_{n+1}(x) = \min_{a\in\mathcal{A}_x}\{c(x,a) + \sum_{y\in\mathcal{X}} p(x,a,y)\tilde{V}_n(y,r)\}, x \in \tilde{\mathcal{X}}.$$

Starting at $n = 0$ with the initial parameter vector $r = 0$, the AVI will update r, and hence $\tilde{V}_n(y,r)$, at the end of each iteration step until r is converged. Updating the parameter vector r can be achieved by solving a weighted least squares problem, because we want to minimise the distance between $\hat{V}_{n+1}(x)$ and $\tilde{V}_{n+1}(x,r)$. Thus,

$$\min_r \sum_{x\in\tilde{\mathcal{X}}} w(x)(\hat{V}_{n+1}(x) - \tilde{V}_{n+1}(x,r))^2. \tag{4.5}$$

If (4.5) is written in matrix notation, where $\tilde{V}_{n+1}(x,r)$ is replaced by $\sum_{i=1}^{R} r_i\phi_i(x)$ we obtain:

$$\min_r (\hat{V}_{n+1} - r\Phi)^T W(\hat{V}_{n+1} - r\Phi),$$

$$\text{with } \Phi = \begin{pmatrix} \phi_1(x_1) & \phi_2(x_1) & \cdots & \phi_R(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \cdots & \phi_R(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(x_{\tilde{N}}) & \phi_2(x_{\tilde{N}}) & \cdots & \phi_R(x_{\tilde{N}}) \end{pmatrix} \text{ and } W = \begin{pmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_{\tilde{N}} \end{pmatrix},$$

where $\tilde{N} = |\tilde{\mathcal{X}}|$. The solution for this weighted least squares problem is given by:

$$r = (\Phi^T W\Phi)^{-1}\Phi^T W\hat{V}_{n+1}. \tag{4.6}$$

Algorithm 2 Approximate Value Iteration

procedure APPROXIMATE VALUE ITERATION($\tilde{\mathcal{X}}, \mathcal{A}_x, P, c, \epsilon$)
Inputs: $\tilde{\mathcal{X}}$: the set of all representative states with $|\tilde{\mathcal{X}}| = \tilde{N}$, $\mathcal{A}_x$: the set of all actions, P: a transition probability matrix specifying $p(x, a, y)$, c: a cost function $c(x, a)$, ϵ: a threshold > 0
Outputs: r: optimal parameter vector
Local variables: int n, **double** min, max, **Vector** $V(0, \ldots, \tilde{N}), r(1, \ldots, R)$
    $V \leftarrow 0$
    $n \leftarrow 0$
    $r \leftarrow 0$
    do
        $min = \infty$
        $max = -\infty$
        for each state $x \in \tilde{\mathcal{X}}$ **do**
            $\hat{V}_{n+1}(x) = \min_{a \in \mathcal{A}_x} \{c(x, a) + \sum_{y \in \mathcal{X}} p(x, a, y) \tilde{V}_n(y, r)\}$
        end for
        $r_{n+1} = (\Phi^T W \Phi)^{-1} \Phi^T W \hat{V}_{n+1}$
        for $i = 1 \ldots R$ **do**
            $min = \min\{min, r_{n+1}(i) - r_n(i)\}$
            $max = \max\{max, r_{n+1}(i) - r_n(i)\}$
        end for
        $n = n + 1$
    while $max - min > \epsilon$
end procedure

The pseudo code of AVI is given in Algorithm 2.

Applying this algorithm to our $M/M/3$ queue, with the estimated value function given by:

$$\begin{aligned}\hat{V}_{n+1}(x) = x &+ 0.15 \min\{r_1(x+1) + r_2(x+1)^2; r_1 x + r_2 x^2 + 20\} \\ &+ 0.1 \min\{r_1(x+1) + r_2(x+1)^2; r_1 x + r_2 x^2 + 25\} \\ &+ 0.1 \min\{x, 3\}(r_1(x-1)^+ + r_2((x-1)^+)^2) \\ &+ (1 - 0.25 - 0.1 \min\{x, 3\})(r_1 x + r_2 x^2),\end{aligned}$$

and where $\Phi = \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ 2 & 4 \\ 3 & 9 \\ 4 & 16 \end{pmatrix}$ and $W = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$,

we obtain $r_1 = 11.54$ and $r_2 = 1.10$, and hence $\tilde{V}(x) = 11.54x + 1.10x^2$. Compared with the

BEM method this is not a one-step policy improvement, since no initial policy is required. The parameter vector r converges to a (near) optimal solution.

4.5 Value iteration vs ADP

Now that we have explained the BEM method and the AVI method, we compare the results of these ADP methods with each other and with value iteration. As explained in Section 3.1 we are interested in finding the optimal policy that minimises the long-run expected average cost g^*. This policy is found when value iteration is applied. However, since value iteration is not applicable for large state spaces, we want to know how both ADP methods behave, relative to value iteration, where the long-run expected average cost g is used as a performance measure. The $M/M/s$ queue can be used for this. The state space of the $M/M/s$ queue is one-dimensional and hence, we can easily limit the state space such that value iteration can be applied. We apply value iteration, the two BEM methods (estimating the unknown g in the Bellman error by simulation and by setting $V(0) = 0$) and AVI to several $M/M/s$ queues with different parameters, which we refer to as cases. The parameters of these cases are given in Table 4.1. As approximation function we choose $\tilde{V}(x, r) = r_1 x + r_2 x^2$. This is based on the approximation function for an $M/M/1$ queue of which it is known that it gives a good approximation of the value function [16]. Furthermore, we choose $\tilde{\mathcal{X}} = \{0, 1, \ldots, 15\}$ as the set of representative states.

Table 4.1: Different cases of the $M/M/s$ queue.

Case	λ_1	λ_2	μ	s	c_1	c_2
1	0.15	0.1	0.1	6	20	25
2	0.15	0.1	0.1	6	5	30
3	0.15	0.1	0.1	6	30	5
4	0.15	0.1	0.1	6	25	20
5	0.15	0.1	0.1	3	20	25
6	0.15	0.1	0.1	3	5	30
7	0.15	0.1	0.1	3	30	5
8	0.15	0.1	0.1	3	25	20
9	0.3	0.1	0.1	6	20	25
10	0.3	0.1	0.1	6	5	30
11	0.3	0.1	0.1	6	30	5
12	0.3	0.1	0.1	6	25	20
13	0.3	0.1	0.15	3	20	25
14	0.3	0.1	0.15	3	5	30
15	0.3	0.1	0.15	3	30	5
16	0.3	0.1	0.15	3	25	20

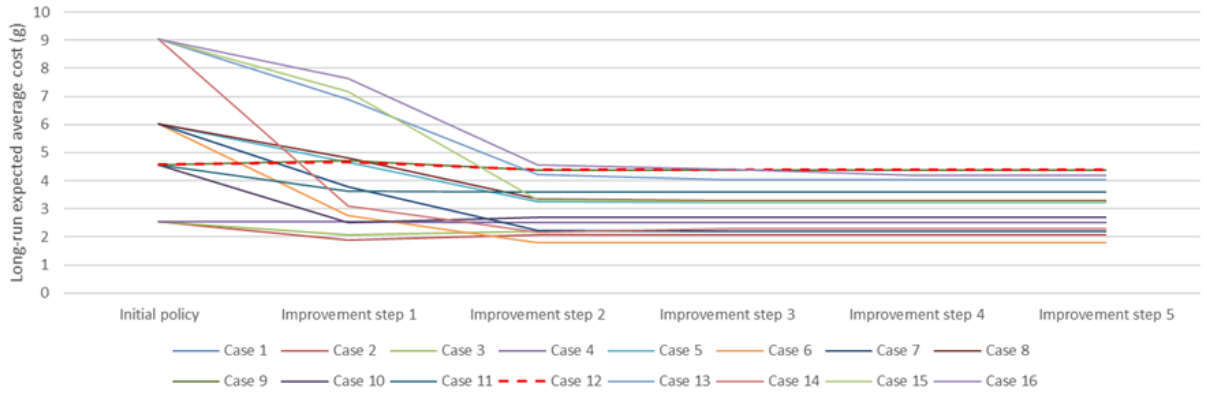


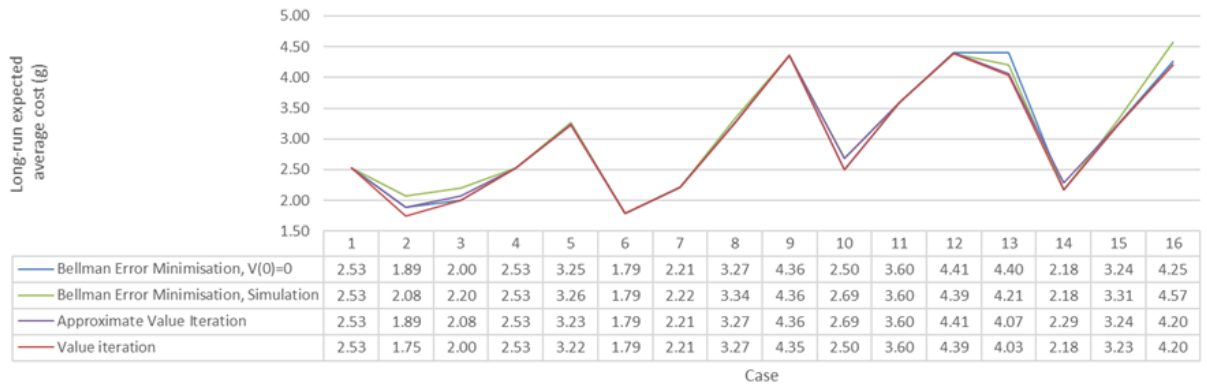
Figure 4.1: Effect on long-run expected average cost per case when the BEM method with simulation is repeated.

First, we compare the results of applying both BEM methods multiple times. The results of the BEM method with simulation are shown in Figure 4.1. As stated before, the results show indeed that policy improvement gives the biggest improvement during the first steps. After two improvement steps the long-run expected average cost is nearly the same as when five improvement steps are performed. Similar results holds for the BEM method with $V(0) = 0$, the strongest decrease in g occurs in the first two steps after which g more or less remains the same. The average improvement over all cases is for both methods in the first two steps around 20% per step, after this it reduces to around zero, see also Table 4.2. It should be noticed that a one-step policy improvement is not necessarily an improvement. In that case the initial policy is already a good policy. This occurs in case 12 in Figure 4.1 (red, dotted line). It shows an increase from 4.57 to 4.68 in g after the first improvement step.

Table 4.2: Average reduction of g per improvement step for both Bellman Error Minimisation methods.

Improvement step	Bellman V(0)=0	Bellman simulation
1	21%	23%
2	21%	18%
3	0%	1%
4	1%	0%
5	-1%	0%

Second, we compare the long-run expected average cost for both BEM methods after two iterations with AVI and value iteration, were value iteration is applied with state space $\mathcal{X} = \{0, 1, \ldots, 500\}$. The results are displayed in Figure 4.2. We see that g of the ADP methods are close to g^* obtained from value iteration. AVI performs slightly better than or is equal to the BEM methods in 13 of the 16 cases. The BEM method with $V(0) = 0$ performs slightly better than or is equal to the BEM method with simulation in 14 of the 16 cases.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bellman Error Minimisation, V(0)=0	2.53	1.89	2.00	2.53	3.25	1.79	2.21	3.27	4.36	2.50	3.60	4.41	4.40	2.18	3.24	4.25
Bellman Error Minimisation, Simulation	2.53	2.08	2.20	2.53	3.26	1.79	2.22	3.34	4.36	2.69	3.60	4.39	4.21	2.18	3.31	4.57
Approximate Value Iteration	2.53	1.89	2.08	2.53	3.23	1.79	2.21	3.27	4.36	2.69	3.60	4.41	4.07	2.29	3.24	4.20
Value iteration	2.53	1.75	2.00	2.53	3.22	1.79	2.21	3.27	4.35	2.50	3.60	4.39	4.03	2.18	3.23	4.20

Figure 4.2: Long-run expected average cost per method.

Finally, for each ADP method we compare the running time of one case for different sizes of the state space. In order to compare the ADP methods with value iteration we take the set of representative states as the total state space, thus $\tilde{\mathcal{X}} = \mathcal{X}$. For the BEM method we compute the running time for one improvement step, since the running time grows linearly in the number of improvement steps. Only the results of the BEM method with $V(0) = 0$, AVI and value iteration are displayed in Figure 4.3, since no difference in running time was obtained between both BEM methods. Value iteration shows the fastest running time over both ADP methods, but from the literature we know that the computational requirements of value iteration grow exponentially with the dimension of the state space [14]. See also Section 6.1.3 for the running time of value iteration of a small instance of our scheduling process. Furthermore, Figure 4.3 shows that the running time of the AVI method increases faster as the state space increases compared to the running time of the BEM method, this is due to the multiple matrix multiplications used to solve the least squares problem, see (4.6).

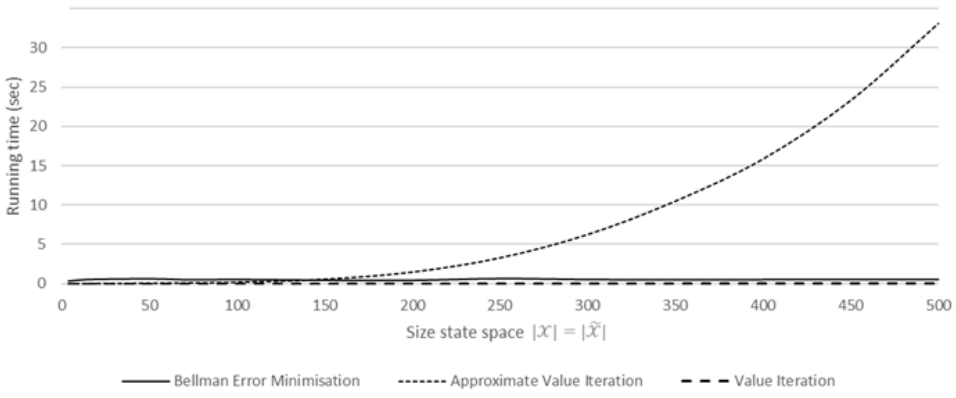


Figure 4.3: Running time of Bellman Error Minimisation, Approximate Value Iteration and value iteration for different sizes of the state space.

4.6 (Dis)advantages of value iteration and ADP

Now that we have discussed the performance of the ADP methods, we give an overview of advantages and disadvantages of all the methods.

Value iteration

Advantages: Gives an optimal policy and no initial policy is required.
Disadvantage: Long running time if the state space and/or action space is large, also known as the curse of dimensionality.

Bellman Error Minimisation with g obtained by simulation and by setting $V(0) = 0$.

Advantages: Short running time and the biggest improvement is achieved during the first steps.
Disadvantages: Initial policy required and if the this policy is already a (near) optimal policy, than a one-step policy improvement does not necessary give an improvement.
In general: g obtained by simulation is preferred, provided that the simulation runs long enough. This g is more accurate than the g obtained by setting $V(0) = 0$.

Approximate Value Iteration

Advantages: No initial policy required and the real parameters are obtained at once. It seems to perform better than the BEM methods.
Disadvantages: Long running time if the reduced state space is large.

Conclusion

Since AVI has a long running time if the reduced state space is large, we choose to apply the BEM method to our MDP. Because g obtained by simulation is preferred over g obtained by setting $V(0) = 0$, we apply the BEM method with simulation as ADP technique to our scheduling process.

5 Method and parameters

In this chapter we formulate our scheduling process as an MDP and we describe how we apply the BEM method with simulation as ADP technique to our MDP. Our MDP is formulated in Section 5.1. In Section 5.2 we give the Bellman Error for our model and describe how we determine the different elements needed for the BEM method. Then, in Section 5.3 we discuss how we choose the parameters we defined in our MDP. Ultimately, we are interested in an approximation function that shows in general the best improvements for the one-step policy improvement for different cases. How we address this is described in Section 5.4.

5.1 Scheduling process as an MDP

Before we formulate the scheduling process as an MDP, we first describe how we model the requests for appointments from different types of patients such as new patients, control patients and several grades of emergency patients. In order to differentiate between these patients in our model we distinguish two categories of patients. One category is characterised by a maximum recommended waiting time. Among this category new patients and emergency patients can be considered. The other category is characterised by a specific appointment time. Among this category patients who want to make an appointment in advance, like control patients, can be considered. For the first category the goal is to minimise the waiting time between the scheduled appointment time and the time the request for an appointment is made. For the second category the goal is to minimise the time between the scheduled appointment time and the specific appointment time. Furthermore, each patient type is characterised by several variables like service time and arrival rate. More formally, a patient can be of type $(i, j), i \in \{1, 2\}, j \in \{1, \dots, J_i\}$, where i represents the category the patient belongs to. If $i = 1$ the patient belongs to the category with a maximum recommended waiting time and if $i = 2$ the patient belongs to the category with a specific appointment time. Each patient of type (i, j) is characterised by the variables:

- k_{1j} Extra costs for exceeding the maximum recommended waiting time for a patient of type $(1, j), k_{1j} > 0$;
- p_{ij} The probability that a patient of type (i, j) will arrive during a time interval, $p_{ij} > 0$;
- r_{ij} The penalty cost of rejecting a patient of type $(i, j), r_{ij} > 0$;
- w_{ij} If $i = 1$, this represents the maximum recommended waiting time for a patient of type j and if $i = 2$ this represents the specific appointment time for a patient of type $j, w_{ij} \in \{1, 2, \dots\}$ and is expressed in working days;
- β_{ij} The required scheduled service time for a patient of type $(i, j), \beta_{ij} \in \{1, 2, \dots\}$ and is expressed in the number of blocks, see the next section about the state space.

Now, we can formulate the scheduling process as an MDP, which is described by a state space, an action space, a cost function and its transition probabilities, see Section 3.1.

State space

On any given day the scheduling process takes place over the next N working days, described by the set $\mathcal{N} = \{1, \ldots, N\}$. To keep our model simple, $n = 1$ represents the next working day so it is not possible to schedule patients at the same day they make their request for an appointment. Furthermore, patients are only assigned to an appointment day, not an appointment time, this is outside the scope of this thesis. Each day we have a fixed amount of capacity available, C. In our model, the available capacity is described as the number of available blocks, where a block is defined as a time period of 5 minutes. As stated in Section 2.2 our model is inspired by the models described by Erdelyi and Topaloglu [4] and Patrick et al. [13]. However, both models observe the arrivals of patients at the beginning of each day, which means that several patients need to be scheduled at once. In reality patients are scheduled immediately at the moment they make their request. We want our model to approach reality as closely as possible. Therefore, we cut the current day into T intervals, where T is large enough such that we may assume that in one interval only one event can occur, namely one patient arrival or no arrival. Thus in our model the state space takes the form:

$$\mathcal{X} = (\vec{x}; t) = (x_1, x_2, \ldots, x_N; t),$$

with $x_n \in \{0, 1, \ldots, C\}$, the number of blocks already booked n working days ahead and $t \in \{1, \ldots, T\}$ the time of the current day.

We determine T as follows. During one day, patient arrivals (requests) occur according to a Poisson process with parameter λ. If we cut the day into T intervals, then in each interval patient arrivals occur according to a Poisson process with parameter λ/T. For our MDP it is important that the probability of more than one arrival in an interval is small, where we define small as less than 5%. More formally:

$$\begin{aligned} & P(\text{number of arrivals} > 1) < 0.05 \\ \Rightarrow\ & 1 - P(\text{number of arrivals} = 0) - P(\text{number of arrivals} = 1) < 0.05 \\ \Rightarrow\ & 1 - \frac{(\lambda/T)^0}{0!} e^{-\lambda/T} - \frac{(\lambda/T)^1}{1!} e^{-\lambda/T} < 0.05 \\ \Rightarrow\ & (1 + \lambda/T) e^{-\lambda/T} \geq 0.95. \end{aligned} \tag{5.1}$$

To determine T we solve this equation numerically.

Action space

When a patient arrives, the planner needs to decide to which day the patient will be assigned. Additionally, if there is not enough capacity available, then there must be an option to reject the patient or to schedule the patient in overtime. Thus the action space takes the form:

$$\mathcal{A}_{\vec{x},t} = (\vec{a}) = (a_1, a_2, \ldots, a_N, a_{N+1}),$$

with $a_n \in \{0,1\}$ is the action of booking the patient n working days ahead, $n \in \mathcal{N}$ and $a_{N+1} \in \{0,1\}$ is the action of rejecting the patient or serve the patient during overtime. From here, we only write about the action of rejecting a patient, just for simplicity. For a patient of type (i,j) it must hold that: $x_n + \beta_{ij} a_n \leq C$, $n \in \mathcal{N}$, which means that the available capacity on day n is not exceeded. Also, $\sum_{n=1}^{N+1} a_n = 1$, which means that the patient can be scheduled at most on one day and if the patient is not scheduled then the patient is rejected.

Cost function

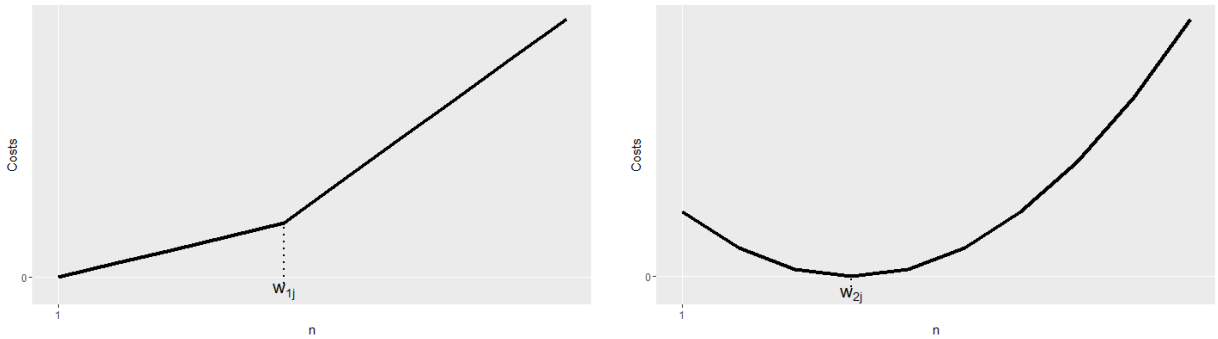
The cost function depends on the category to which a patient belongs. If $i = 1$, a patient needs an appointment within its maximum recommended waiting time (w_{1j}); the sooner this patient is scheduled the better, but when w_{1j} is exceeded extra costs are incurred with factor k_{1j}. If the patient is rejected, rejection costs r_{1j} are incurred. We can write the cost function for type $i = 1$ patients as:

$$c_{1j}(\vec{a}) = \Big[\sum_{n \in \mathcal{N}} (n - 1 + k_{1j}(n - w_{1j})^+) a_n \Big] + r_{1j} a_{N+1}.$$

Note that no costs are incurred when the patient is scheduled on the first working day to come. If $i = 2$, a patient needs an appointment around its specific appointment time (w_{2j}); the closer the scheduled appointment time is to the specific appointment time the better. We choose here for a quadratic structure to penalise large deviations more severely. If the patient is rejected, rejection costs r_{2j} are incurred. We can write the cost function for type $i = 2$ patients as:

$$c_{2j}(\vec{a}) = \Big[\sum_{n \in \mathcal{N}} (w_{2j} - n)^2 a_n \Big] + r_{2j} a_{N+1}.$$

Figure 5.1 displays the structure of both cost functions when patients are scheduled.



(a) Category 1 (b) Category 2

Figure 5.1: Cost function structure by patients category.

Transitions and transition probabilities

With probability p_{ij} a patient of type (i, j) makes a request for an appointment during a time interval $t \in \{1, \ldots, T\}$. In one interval at most one patient can make an appointment request. Hence, with probability $\sum_{i,j} p_{ij}$ there is one request in a time interval and with probability $1 - \sum_{i,j} p_{ij}$ there is no request in a time interval. The transition to the next state also depends on the value of state t. If $t < T$, there is a transition to the next time interval, but if $t = T$, which means it is the end of a day, there is a complete shift in the schedule. The first day disappears from the schedule, the second day becomes the first day, the third day becomes the second day and so on, and finally a new empty day enters the schedule and we start with $t = 1$. Hence, we distinguish four different kinds of transitions:

1. If $t < T$ and a patient of type (i, j) arrives the transition can be written as:
$$(x_1, x_2, \ldots, x_N; t) \to (x_1 + \beta_{ij} a_1, x_2 + \beta_{ij} a_2, \ldots, x_N + \beta_{ij} a_N; t + 1).$$
2. If $t < T$ and there is no arrival the transition can be written as:
$$(x_1, x_2, \ldots, x_N; t) \to (x_1, x_2, \ldots, x_N; t + 1).$$
3. If $t = T$ and a patient of type (i, j) arrives the transition can be written as:
$$(x_1, x_2, \ldots, x_N; t) \to (x_2 + \beta_{ij} a_2, x_3 + \beta_{ij} a_3, \ldots, x_N + \beta_{ij} a_N, 0; 1).$$
4. If $t = T$ and there is no arrival the transition can be written as:
$$(x_1, x_2, \ldots, x_N; t) \to (x_2, x_3, \ldots, x_N, 0; 1).$$

Notice that because of these transition probabilities the associated Markov decision chain is periodic with period T; therefore we need to apply a data transformation as explained in Section 3.3.

Optimality equations

Given the state space, action space, cost function and transition probabilities and the fact that the associated Markov decision chain is periodic, the optimality equations of this MDP are given by:

$$\begin{aligned} V(\vec{x}, t) + g = \\ \mathbb{1}_{\{t<T\}} \Bigg[\sum_{i,j} \gamma p_{ij} \min_{\vec{a} \in \mathcal{A}_{\vec{x},t}} \Big\{ c_{ij}(\vec{a}) + V(x_1 + \beta_{ij} a_1, x_2 + \beta_{ij} a_2, \ldots, x_N + \beta_{ij} a_N; t + 1) \Big\} \\ + \gamma (1 - \sum_{i,j} p_{ij}) V(\vec{x}, t + 1) + (1 - \gamma) V(\vec{x}, t) \Bigg] \\ + \mathbb{1}_{\{t=T\}} \Bigg[\sum_{i,j} \gamma p_{ij} \min_{\vec{a} \in \mathcal{A}_{\vec{x},t}} \Big\{ c_{ij}(\vec{a}) + V(x_2 + \beta_{ij} a_2, x_3 + \beta_{ij} a_3, \ldots, x_N + \beta_{ij} a_N, 0; 1) \Big\} \\ + \gamma (1 - \sum_{i,j} p_{ij}) V(x_2, x_3, \ldots, x_N, 0; 1) + (1 - \gamma) V(\vec{x}, t) \Bigg] \end{aligned} \tag{5.2}$$

Notice that in Section 3.3 we stated that $(V^*/\gamma, g^*)$ is a solution of (3.3). However, in our model the costs $c_{ij}(\vec{a})$ do not only depend on the action that is chosen, but they also depend on which type of patient makes a request (p_{ij}). This means that the costs are also multiplied with γ. Hence, it holds that if (V^*, g^*) is a solution of the model without a data transformation, then $(V^*, \gamma g^*)$ is a solution of (5.2). Also notice that the dimension of the state space is $(C+1)^N T$ which is extremely large for reasonable values of C, N and T and therefore value iteration becomes computational infeasible even for relatively small problem instances to obtain the optimal policy. Therefore, we apply the BEM method with simulation as ADP technique to our MDP.

5.2 Bellman Error Minimisation for the scheduling process

The Bellman error of our MDP is given by:

$$
\begin{aligned}
D(\vec{x}, t, \vec{r}) = &-g - \tilde{V}(\vec{x}, t, \vec{r}) \\
&+\mathbb{1}_{\{t<T\}}\Bigg[\sum_{i,j}\gamma p_{ij}\min_{\vec{a}\in\mathcal{A}_{\vec{x},t}}\Big\{c_{ij}(\vec{a}) + \tilde{V}(x_1+\beta_{ij}a_1, x_2+\beta_{ij}a_2, \ldots, x_N+\beta_{ij}a_N; t+1; \vec{r})\Big\} \\
&\qquad + \gamma(1-\sum_{i,j}p_{ij})\tilde{V}(\vec{x}, t+1, \vec{r}) + (1-\gamma)\tilde{V}(\vec{x}, t, \vec{r})\Bigg] \\
&+\mathbb{1}_{\{t=T\}}\Bigg[\sum_{i,j}\gamma p_{ij}\min_{\vec{a}\in\mathcal{A}_{\vec{x},t}}\Big\{c_{ij}(\vec{a}) + \tilde{V}(x_2+\beta_{ij}a_2, x_3+\beta_{ij}a_3, \ldots, x_N+\beta_{ij}a_N, 0; 1; \vec{r})\Big\} \\
&\qquad + \gamma(1-\sum_{i,j}p_{ij})\tilde{V}(x_2, x_3, \ldots, x_N, 0; 1; \vec{r}) + (1-\gamma)\tilde{V}(\vec{x}, t, \vec{r})\Bigg]
\end{aligned}
$$

As stated in Section 4.3 the goal of the BEM method is to find a good approximation for the most important part of the state space, which can be achieved by minimising the weighted sum of squared Bellman errors for a set of representative states $\tilde{\mathcal{X}} \subset \mathcal{X}$, see (4.3). Before we can apply the BEM method we need to determine the different elements needed for this method. As described in Section 4.3 these elements are:

- An initial policy;
- The long-run expected average cost for the initial policy, g;
- The set of representative states, $\tilde{\mathcal{X}} \subset \mathcal{X}$;
- The weights, $w(x), x \in \tilde{\mathcal{X}}$;

Initial policy

As initial policy we choose a greedy policy. Patients of type $i = 1$ are scheduled as soon as possible and patients of type $i = 2$ are scheduled as closely as possible to their specific

appointment time. Patients are only rejected if there is not sufficient capacity available on any day.

Long-run expected average cost

We use simulation to determine the long-run expected average cost, g, belonging to a certain policy. The simulation is performed as follows: starting with the first day, this day is cut into T intervals. In each interval $t \in \{1, \ldots, T\}$ there is one patient arrival of type (i, j) with probability p_{ij} or no patient arrival with probability $1 - \sum_{ij} p_{ij}$. If $t \neq T$ and there is a patient arrival, this patient is scheduled according to

$$\min_{\vec{a} \in \mathcal{A}_{\vec{x},t}} \left\{ c_{ij}(\vec{a}) + \tilde{V}(x_1 + \beta_{ij} a_1, x_2 + \beta_{ij} a_2, \ldots, x_N + \beta_{ij} a_N; t+1; \vec{r}) \right\}$$

and the corresponding costs, $c_{ij}(\vec{a})$, are incurred. After the patient is scheduled we move to the next interval. If no arrival occurs we move directly to the next interval. If $t = T$ and there is a patient arrival, this patient is scheduled according to

$$\min_{\vec{a} \in \mathcal{A}_{\vec{x},t}} \left\{ c_{ij}(\vec{a}) + \tilde{V}(x_2 + \beta_{ij} a_2, x_3 + \beta_{ij} a_3, \ldots, x_N + \beta_{ij} a_N, 0; 1; \vec{r}) \right\}$$

and the corresponding costs, $c_{ij}(\vec{a})$, are incurred. After the patient is scheduled we move to the next day and shift the schedule. The first day disappears from the schedule, the second day becomes the first day, the third day becomes the second day and so on, and finally a new empty day enters the schedule and we start with $t = 1$. If no arrival occurs we move directly to the new day and shift the schedule. We let the simulation run over 100.000 days. At the end of the simulation we can obtain g by dividing the total costs incurred by the length of the simulation, $100.000 * T$. Note that the initial policy can be achieved by setting the parameter vector $\vec{r}$ to zero. In this case $\tilde{V}(\vec{x}, t, \vec{0})$ equals zero, $\forall \vec{x}, t$ and hence, the actions to choose only depends on the cost function. To make sure we only reject patients if there is no sufficient capacity available in any day, the rejection costs must be chosen higher than the highest costs that can be obtained when patients are scheduled.

Set of representative states and the corresponding weight vector

The set $\tilde{\mathcal{X}}$ should contain the most important states in the state space, while w should represent the importance of the states in $\tilde{\mathcal{X}}$. As mentioned in Section 4.1 the choice of the set of representative states could be to include only the states that have a high probability of being visited. Since, it takes too much time to calculate the stationary distribution, due to the large number of different states, we applied another approach which we discuss here.
Step 1 Determine the last S states that are visited in the simulation.
In the simulation which is performed to determine g, we keep track of the last S states that are visited.
Step 2 Create a list of important states.

The last S states are added to what we call the list of important states with probability 0.1. We have two reasons for doing this randomly with a low probability. First, we want to exclude the dependency between successive states in the simulation. Two successive states are dependent to each other, because given a certain state it is only possible to reach a few other states. Second, we want to decrease the chance that states that are rarely visited are included in the list of important states, whereas we want to increase the chance that states that are often visited are in this list. For instance, if the probability of a rarely visited state is 1:500 than the probability that this state is added to the list of important states equals 0.002%. If the probability of a state that is often visited is 100:500 than the probability that this state is added to the list of important states equals 0.2%

Step 3 K-means clustering

Now that we have a list of important states we want to cluster these states into K clusters. As cluster technique we use the Hartigan and Wong k-means algorithm. The details of this algorithm are discussed in Li [11]. This algorithm returns, among other things, the number of states in each cluster and the cluster centres.

Step 4 Determine the set of representative states and the corresponding weight vector

For each cluster centre we want to find the state with the shortest Euclidean distance as the most representative state of this cluster. Hence, after this step, we have a set of K representative states, which will be our $\tilde{\mathcal{X}}$ and the number of states in each cluster will determine our weight vector w.

5.3 Parameters

Patient characteristics

For our model, we need to determine the patient characteristics as defined in Section 5.1. To determine p_{ij}, the probability that a patient will arrive during a time interval; w_{ij}, the maximum recommended waiting time or the specific appointment time for a patient and β_{ij}, the required service time for a patient, we explored the data set from a hospital. This data set contains one year of information about patients request for appointments. For each request the following details are given:

- Date of request of the appointment;
- Date of the appointment;
- Number of working days between the appointment date and the date of request;
- Duration of the appointment;
- Appointment type, which consists of:
  - New patient;
  - Control patient;
  - Consult by telephone.

A preview of the data set is given in Table 5.1. First the data set is cleaned. Patients who have their appointment date before their request date are removed from the data. Also patients who have a duration of 0 minutes are removed from the data.

Table 5.1: Preview of the request data set.

Patient	Request date	Appointment date	Working days	Duration	Type
1	12-11-2014	03-12-2014	15	10 min.	New
2	08-12-2014	23-01-2015	34	20 min.	Control
3	26-05-2015	29-05-2015	3	10 min.	Telephone
4	17-02-2015	04-03-2015	11	20 min.	Control
5	26-05-2014	02-06-2014	5	10 min.	New

Table 5.2: Proportion between new patients, control patients and consults by telephone.

Type	Percentage
New Patients	23%
Control Patients	64%
Consults by telephone	13%

Table 5.2 shows the proportion between the different types of appointments in this data set. For each type of patient we explore the distribution of 'Working days' to determine the maximum recommended waiting time or the specific appointment time, w_{ij}. We also explore the distribution of 'Duration' for each type of patient to determine the required service time, β_{ij}. Figure 5.2 displays the distribution of 'Working days' for control patients. The highest peak is just before zero, which means that these control patients get an appointment the same day they make their request. Furthermore, there is a peak every five working days. This means that when the patients make their requests, most of the time they get their appointment $5, 10, 15, \ldots$ working days later. Hence, for the group of control patients ($i = 2$) we set w_{2j} to 1*, 5, 10, 15 and so on.

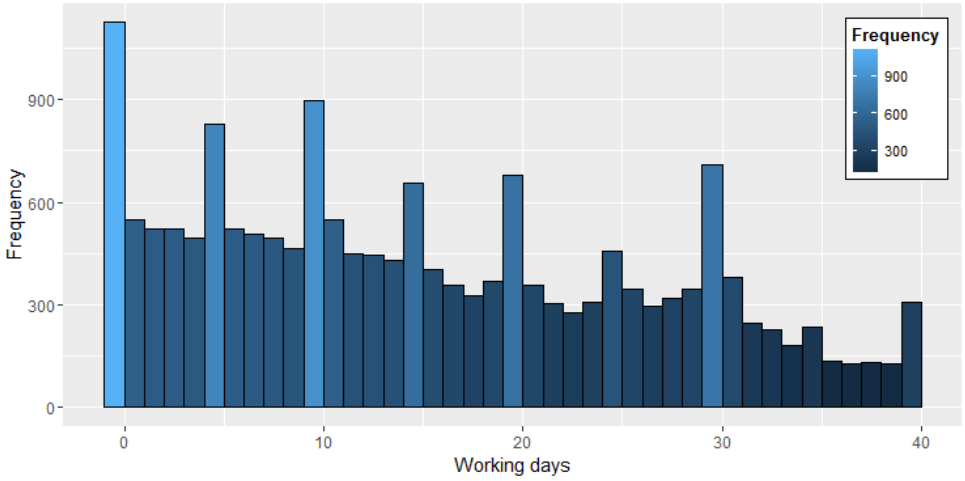


Figure 5.2: Distribution of 'Working Days' for control patients.

* In our model it is not possible to schedule patients at the same day they make their request for an appointment, for these patients w_{ij} is set to 1.

Figure 5.3 displays the distribution of 'Duration' for control patients. As can be seen 64% of the control patients have an appointment time of 10 minutes. Hence, for all control patients we set β_{2j} to 2 blocks**. The figures of the distribution of 'Working days' and 'Duration' for new patients and consults by telephone are shown in Appendix A. For the consults by telephone the distribution of 'Working days' shows the same pattern as the control patients except that there is also a large group that have a consult by telephone the next day they make their request. Furthermore, 77% of the consults by telephone have an appointment time of 5 minutes. We split the consults by telephone over the two categories. For $(i = 1)$ we set w_{1j} to 1, for $(i = 2)$ we set w_{2j} to 5 and 10. For both categories we set β_{ij} to 1 block. For new patients the distribution of 'Working days' shows the highest peak around 5 working days. 49% of the new patients have an appointment time of 10 minutes. 23% have an appointment time of more than 20 minutes, with an average of 25 minutes. Hence, for the group of new patients $(i = 1)$ we set w_{1j} to 5 working days and we set β_{1j} to 2 and 5 blocks. An overview of the patient types we created is given in Table 5.3. By determining the proportion of each patient type we created, we set p_{ij} to this proportion, whereas the proportions of the three appointment types, displayed in Table 5.2, is preserved.

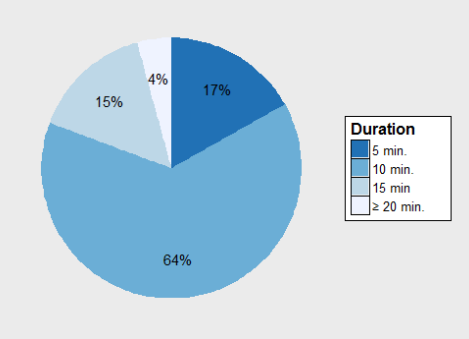


Figure 5.3: Distribution of 'Duration' for control patients.

Table 5.3: Overview of the created patient types.

i	w_{ij}	β_{ij}
1	1	1
1	5	2
1	5	5
2	5	1
2	10	1

i	w_{ij}	β_{ij}
2	1	2
2	5	2
2	10	2
2	15	2
⋮	⋮	⋮

Cost parameters

Next to p_{ij}, w_{ij} and β_{ij} we need to set k_{1j} the extra costs for exceeding the maximum recommended waiting time and r_{ij}, the rejection costs. We assume that patients with a lower maximum recommended waiting time have a higher urgency than patients with a higher recommend waiting time. Hence, patients with a lower maximum waiting time have a higher k_{1j}. As mentioned in Section 5.2, the rejection costs must be chosen higher than the highest costs that can be obtained when patients are scheduled to make sure that in our initial policy patients are only rejected if there is not sufficient capacity available on any day. The highest costs that can be obtained when patients are scheduled depend on

** Remember that one block consists of five minutes.

N and k_{1j}. Moreover, for patients with a higher service time, the rejection costs are higher than for patients with a lower service time. This is in order to ensure that there is no incentive to reject these patients because of the fact that they take up a relatively large portion of the schedule.

Other parameters

Besides the patient-specific parameters, our model has a few other parameters that need to be determined. These parameters are: N, the number of working days which covers the scheduling process; C, the fixed amount of capacity available on any day; λ, the rate for patient arrivals and from which parameter T can be determined, see (5.1) and γ needed for the data transformation to overcome the problem of aperiodicity. We set C to 24 blocks. Based on experiments with a small MDP we set γ to 0.9. A high value of γ was less time consuming than lower values of γ as can be seen in Figure 5.4. The parameters N and λ are discussed in the next section.

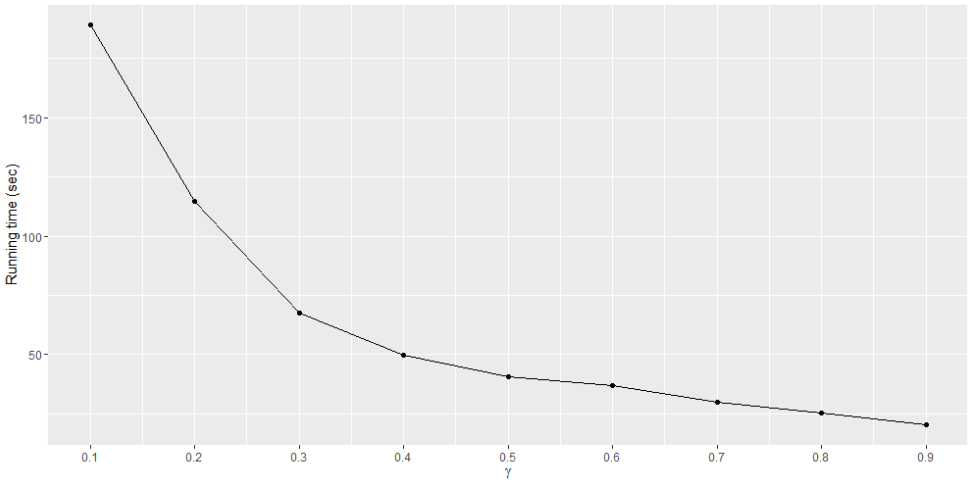


Figure 5.4: Running time of a small MDP for different values of γ.

5.4 Method

We start with a schedule which consists of three working days, in other words $N = 3$. The reason we start with this scenario, is that value iteration can be applied, and hence, the optimal long-run expected average cost, g^*, can be computed and compared with the results obtained from the BEM method. In this way we get an idea about how the BEM method behaves compared to value iteration. We apply the BEM method with different values for λ, the rate for patient arrivals; S, the last states visited in the simulation and K, the number of clusters. The values for each parameter used in the experiments are given in Table 5.4.

Table 5.4: Values for different parameters

Parameter	**Values**
λ	$\{16, 16.5, 17, 17.5, 18, 18.5, 19\}$
S	$\{3.500.000, 4.000.000, 4.500.000, 5.000.000\}$
K	$\{1.000, 5.000, 10.000\}$

Each value for λ results in a different load of the system, denoted as ρ. The higher λ, the higher ρ. Table 5.5 shows for each value for λ the load of the system.

Table 5.5: Load of the system for each value of λ

λ	ρ
16	1.37
16.5	1.41
17	1.45
17.5	1.49
18	1.54
18.5	1.58
19	1.62

For each combination of the parameters in Table 5.4 we apply the BEM method with different approximation functions. An approximation function consists of a set of basis functions. The basis functions we use are displayed in Table 5.6.

Table 5.6: Different types of basis functions

Number	Basis function
0	$t + \sum_{p=1}^{3} \sum_{n=1}^{N} x_n^p$
1	$\sum_{n=1}^{N-1} x_n * x_{n+1}$
2	$\sum_{n=1}^{N-2} x_n * x_{n+1} * x_{n+2}$
3	t^2

Number	Function
4	$\sum_{n=1}^{N} x_n * t$
5	$\sum_{n=1}^{N} x_n^2 * t$
6	$\sum_{n=1}^{N-1} x_n * x_{n+1} * t$
7	$\sum_{n=1}^{N-2} x_n * x_{n+1} * x_{n+2} * t$

Basis function 0 is a polynomial of degree three, which is some kind of basis polynomial needed for a one-dimensional process with a polynomial cost function of at most degree two [16]. Therefore, each approximation function we use contains basis function 0. The

other basis functions contain several cross terms between different parts of the state space. If we refer to approximation function 037, than this approximation function consist of the basis functions: 0, 3 and 7, see Example 5.4.1 for an illustration.

Example 5.4.1. If $N = 3$ and we refer to approximation function 037, than

$$\tilde{V}(\vec{x}, t, \vec{r}) = tr_1 + x_1 r_2 + x_2 r_3 + x_3 r_4 + x_1^2 r_5 + x_2^2 r_6 + x_3^2 r_7 + x_1^3 r_8 + x_2^3 r_9 + x_3^3 r_{10} + t^2 r_{11} + x_1 x_2 x_3 t r_{12}.$$

We start with the following approximation functions: $\{0, 01, 02, 03, 04, 05, 06, 07\}$. From here we use a so-called bottom up approach. We take the functions that show the best improvements overall and then add the other remaining functions one at a time. For instance, if function 04 performs best, than we make the the following new combinations: $\{041, 042, 043, 045, 046, 047\}$. This is repeated until no further improvement occurs. At the end we have a set of approximation functions which shows in general the best improvements for a scheduling process over three working days. To test if these set of approximation functions also perform well for a larger scheduling process we expand our model to $N = 5, 10$ and 20. The parameters S and K are set to the value that in general performs best in the model with $N = 3$.

6 Results

In this chapter the results of our research are displayed and interpreted. We start with the results of the scheduling process over three working days in Section 6.1, followed by the results of the scheduling process over five, ten and twenty working days in Section 6.2.

6.1 Three working days

6.1.1 Bellman Error Minimisation

Remember from Section 5.4 that for each combination of the parameters in Table 5.4 we apply the BEM method. We can make $7*4*3 = 84$ combinations*. However, for high values of λ it was not possible to perform the k-means clustering with $K = 10.000$. This is because high values of λ result in a high load of the system and as a result of this high load, certain states are visited so often that our list of important states did not contain more than 10.000 unique states. This was the case with $\lambda = 18.5$ and $S \in \{3.500.000, 4.000.000, 4.500.000\}$. With $\lambda = 19$ this holds for all values of S. In the end, we have $84-7 = 77$ combinations and for each combination we apply the BEM method with 8 different approximation functions. Each time we apply the BEM method, we compare g obtained from our initial policy with g obtained after the one-step policy improvement and compute the improvement that is made. We refer to this as the improvement of the BEM method.

Table 6.1 shows for each approximation function the median, average and variance of the improvement of the BEM method over 77 combinations. Both, the median and average for functions $\{0, 01, 02, 03\}$ do not show any improvement. Moreover, the average and variance of functions 01 and 02 are considerably lower (average) and higher (variance) compared to the other functions. Because we do not want these results to have a strong influence on the rest of our outcomes, we remove the results of functions 01 and 02 from our results.

Table 6.1: Median, average and variance of the improvement for each approximation function.

Function	Med (%)	Avg (%)	Var
0	-9.5	-14.0	3.76
01	-18.3	-229.4	888.31
02	-3.5	-47.0	169.10
03	-6.2	-8.9	2.52
04	16.3	16.3	0.37
05	10.7	13.5	0.47
06	6.7	8.6	0.43
07	5.6	5.0	0.16

* 7 values for λ, 4 values for S and 3 values for K.

Figure 6.1 shows for each K a box plot of the improvement of the BEM method of the results without functions 01 and 02. In this box plot each point represents the improvement of the BEM method made in one of the combinations of S, λ and the remaining approximation functions. Although the median and average between the number of clusters are close to each other, there are more negative outliers if 1.000 clusters are used and these outliers have a higher negative value. This is easily explained, since the number of clusters are in fact the number of representative states used for the BEM method. If the set of representative states is chosen too small than this is not a representative set of states anymore. Therefore, we also remove the results of the clusters with parameter 1.000.

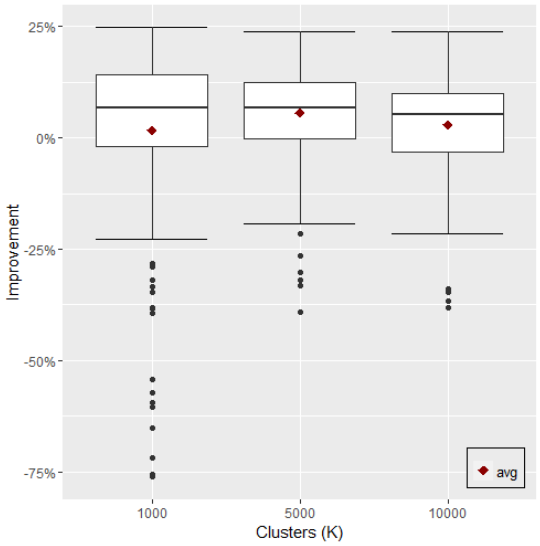


Figure 6.1: Box plots of the improvement for each number of clusters (K).

The box plots in Figure 6.2 display the improvement of the BEM method of the remaining results for each number of last states (S). There seems to be no substantial difference between the number of last states (S). Figure 6.3 shows for each λ a box plot of the improvement of the BEM method. It shows that the lower the value of λ, the worse the outcome. This can be explained because a low value of λ results in a relatively low load of the system, ρ, see 5.5. With a low ρ it seems plausible that our initial policy will work fine and a better policy can not be obtained using the BEM method. Therefore, we also remove the results with $\lambda < 18$. With the remaining results we apply the bottom up approach in order to find a good approximation function that shows in general the best improvement.

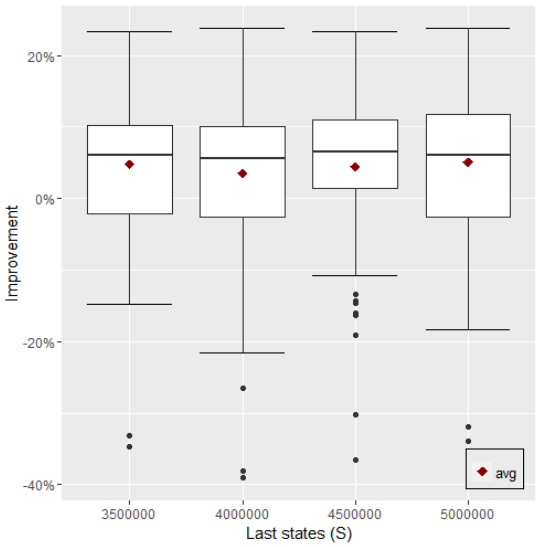


Figure 6.2: Box plots of the improvement for each number of last states (S).

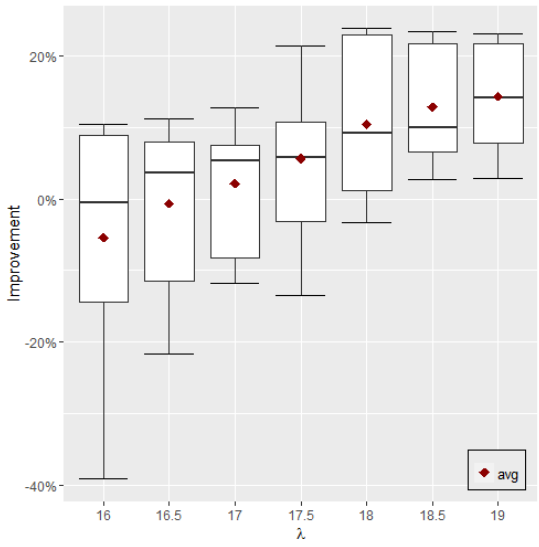


Figure 6.3: Box plots of the improvement for each λ.

6.1.2 Approximation function

Since we removed the results of $K = 1.000$ and $\lambda \in \{16, 16.5, 17, 17.5\}$ we have $3*4*2-7 = 17$ combinations remaining[**]. Table 6.2 shows for the remaining approximation functions the median, average and variance of the improvement of the BEM method over 17 combinations. As can be seen functions 04 and 05 give the best results with an average improvement of 22.8% and 22.0%, respectively. Both functions also have the lowest variances. Therefore, we start with our bottom up approach with function 04 and function 05 over the 17 combinations.

Table 6.2: Median, average and variance of the improvement of the remaining results for different functions.

Function	Med (%)	Avg (%)	Var
0	3.1	3.7	0.21
03	2.7	3.1	0.25
04	22.9	22.8	0.005
05	22.5	22.0	0.03
06	13.7	14.4	0.07
07	6.0	6.3	0.03

Table 6.3 shows the results from the first step of the bottom up approach. For each approximation function the median, average and variance of the improvement of the BEM method are given. For function 04 it holds that only the median increases slightly from 22.9% to 23.1% when function 6 is added. Adding one of the other functions does not

[**] 3 values for λ, 4 values for S, 2 values for K and 7 combinations are not possible.

improve the median, average and/or variance. For function 05 an improvement is made by adding function 6 or 7. Both, the median and average are higher and the variance decreases when function 6 is added and remains the same when function 7 is added. Therefore, in our second step of the bottom up approach we start with the functions 046, 056 and 057. Although only the median shows an improvement when function 6 is added to function 04, we want to investigate this further to see if there is a further improvement possible.

Table 6.3: Median, average and variance of the improvement for the different functions after one step.

(a) Function 04

Function	Med (%)	Avg (%)	Var
041	21.4	21.5	0.01
042	22.2	22.0	0.01
043	20.6	19.1	0.08
045	22.2	20.0	0.12
046	23.1	22.7	0.01
047	22.5	22.3	0.01

(b) Function 05

Function	Med (%)	Avg (%)	Var
051	21.0	16.1	1.96
052	22.1	20.9	0.15
053	20.9	20.3	0.10
054	22.2	20.0	0.12
056	23.3	22.9	0.01
057	22.8	22.5	0.03

Table 6.4 shows the results of the second step of the bottom up approach. Adding function 7 to function 046 improves the median from 23.1% to 23.2%, the average and the variance remain the same. Adding one of the other functions does not show an improvement at all. Adding function 1 to function 056 shows the same results as adding function 1 to function

Table 6.4: Median, average and variance of the improvement for the different functions after two steps.

Function	Med (%)	Avg (%)	Var
0461	20.4	20.3	0.01
0462	20.6	16.9	1.85
0463	17.6	1.2	18.40
0465	22.6	20.7	0.18
0467	23.2	22.7	0.01
0561	23.5	20.9	0.52
0562	23.8	23.3	0.02
0563	21.5	21.2	0.03
0564	22.6	20.7	0.18
0567	23.3	23.3	0.01
0571	23.0	18.2	1.74
0572	22.7	22.3	0.05
0573	22.1	20.6	0.10
0574	22.4	21.3	0.05
0576	23.3	23.3	0.01

Table 6.5: Median, average and variance of the improvement for the different functions after three steps.

Function	Med (%)	Avg (%)	Var
04671	20.4	20.3	0.01
04672	17.6	17.9	0.04
04673	17.6	-4.4	29.26
04675	22.7	20.6	0.19
05621	23.9	22.8	0.06
05623	22.2	22.0	0.01
05624	20.3	19.5	0.05
05627	23.6	23.1	0.02
05761	23.5	22.3	0.13
05762	23.6	23.1	0.02
05763	21.8	21.5	0.02
05764	22.7	20.6	0.19

057; the median is improved, but the variance increases from 0.01 to 0.52 and from 0.03 to 1.74, respectively. Adding function 2 or 7 to function 056 both show an improvement in median and average, the variance remains more or less the same. The same holds for adding function 6 to function 057. In our third step of the bottom up approach we start therefore with the functions {0467, 0562, 0567}. The result of this step are given in Table 6.5. As can be seen no further improvement is obtained.

So, functions {0467, 0562, 0567} are the functions that give the best improvements during the one-step policy improvement, based on the median, average and variance for the scheduling process over three working days. Therefore, we apply these functions to the scheduling process over five, ten and twenty working days. Since functions 04 and 05 perform also very good for the scheduling process over three working days and in order to keep our approximation function as simple as possible we also apply these functions to the scheduling process over five, ten and twenty working days. To simplify the figures in the following sections, we create a translation table, see Table 6.6. From here, if we write about function A, we actually mean function 04. Before we discuss the results of functions {A, B, C, D, E} on the scheduling process over five, ten and twenty working days, we first describe the optimal results obtained from value iteration and we compare these results with the results obtained from the BEM method.

Table 6.6: Translation table for the different functions.

New function name	Old function name
A	04
B	05
C	0467
D	0562
E	0567

6.1.3 Value iteration

Since no difference in improvement is obtained between the number of last states and between 5.000 and 10.000 clusters, we fix the parameters S and K to 3.500.000 and 5.000, respectively. We compare for each function in Table 6.6 and for each λ the results obtained from the BEM method with the results obtained from value iteration. In the BEM method g is obtained by simulation. To handle the fluctuations in the simulation, we perform the BEM method 10 times for each λ and each function in order to obtain a more reliable result for the BEM method.

Table 6.7 shows the results obtained from value iteration. For each λ the corresponding T, the optimal long-run expected average cost, g^*, and the running time for solving the MDP is given. Since the dimension of the state space is given by $(C+1)^N T$, it holds that the higher the value of λ, resulting in a higher value of T, the longer the running

time. The number of states of the state space with $\lambda = 19$ equals: $25^3 * 53 = 828125$ with a running time of more than seven hours. Imagine what the running time will be if we expand the scheduling process to twenty working days, where the number of states of the state space exceeds 4.8×10^{29}. For comparison, the average running time of the scheduling process over three working days for the whole BEM method, including 2 simulations and the k-means algorithm, is 7 minutes and this is independent of the value of λ.

Table 6.7: Running time and g^* of value iteration for different λ's.

λ	T	g^*	Time (hh:mm)
16	45	0.24	03:56
16.5	46	0.33	04:24
17	47	0.45	04:42
17.5	49	0.61	05:21
18	50	0.75	05:44
18.5	52	0.92	06:33
19	53	1.07	07:03

Figure 6.4 shows for each λ the average improvement of the BEM method by the different functions. Remember that with the improvement of the BEM method, we mean the improvement that is made when we compare g obtained from our initial policy with g obtained after the one-step policy improvement. In general, it applies that if $\lambda \leq 18$, the average improvement for each function increases as λ increases. If $\lambda > 18$ the average improvement for each function decreases as λ increases. As mentioned before, the lower λ, the lower ρ. It seems plausible that the lower ρ, the better our initial policy performs and hence, less improvement is possible. The higher ρ, the worse our initial policy performs

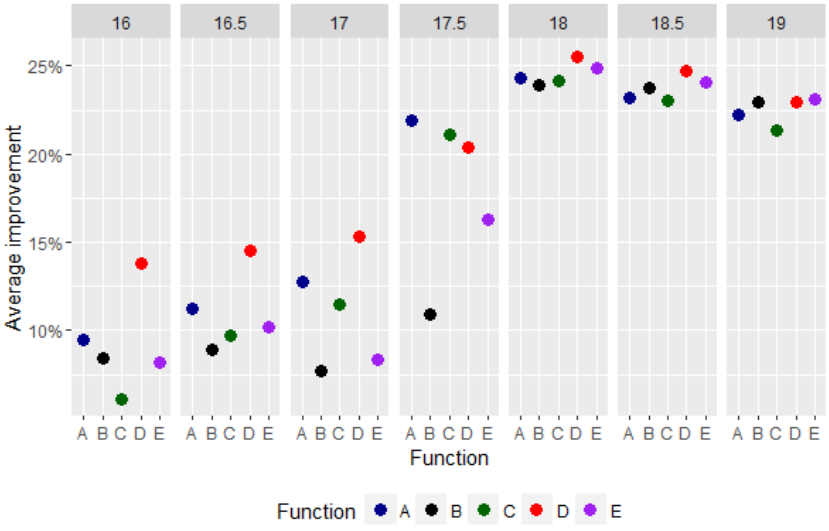


Figure 6.4: Average improvement by λ for different functions.

and the more important is our one-step policy improvement. But if λ reaches a certain value, ρ becomes that high that it does not matter what policy is applied, since many patients have to be rejected and the rejection costs are dominating g.

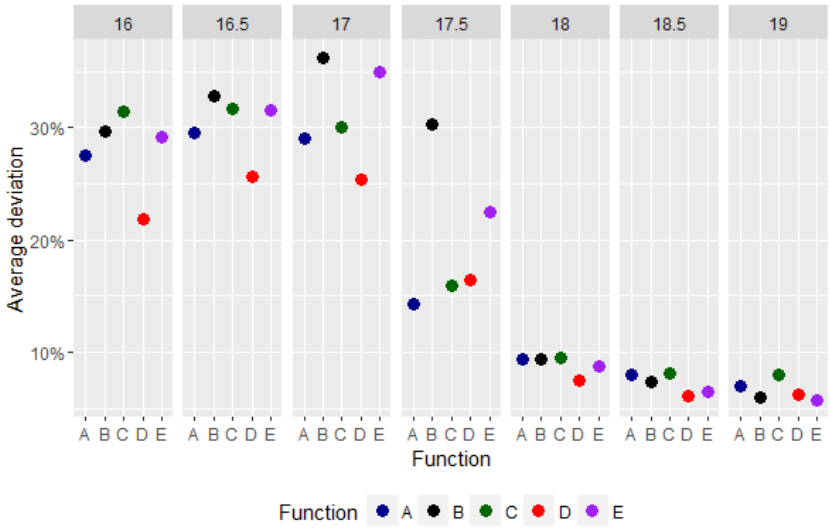


Figure 6.5: Average deviation by λ for different functions.

Figure 6.5 shows for each λ the average deviation by the different functions. The deviation is the difference between g obtained after the one-step policy improvement from the BEM method and g^* obtained from value iteration. In general, it applies that for the lower values of λ the average deviation is higher than for the higher values of λ. For the higher values of λ it holds that the average deviation decreases slightly. We just mentioned that if λ reaches a certain value, ρ becomes that high that it does not matter what policy is applied, and hence g of any policy approaches g^*.

Table 6.8: Average and variance of the improvement and deviation by function.

Function	Improvement		Deviation	
	Avg (%)	Var	Avg (%)	Var
A	17.9	0.37	17.9	0.96
B	15.2	0.55	21.7	1.57
C	16.7	0.48	19.2	1.13
D	19.6	0.23	15.6	0.71
E	16.5	0.52	19.9	1.4

Table 6.8 shows for each function the average and variance of the improvement of the BEM method relative to the initial policy and the average and variance of the deviation of the BEM method relative to g^* obtained from value iteration. Overall, function D, gives the best improvement and approaches on average g^* the best for the scheduling process over three working days.

6.2 Five, Ten, Twenty working days

In this section the results of the approximation functions {A,B,C,D,E} of the scheduling process over five, ten and twenty days are given. The parameters S and K needed for the BEM method are fixed to 3.500.000 and 5.000, respectively. As in the scheduling process over three working days, we perform the BEM method 10 times for each λ and each function in order to handle the fluctuations in the simulation and hence, to obtain a more reliable result for the BEM method.

Figure 6.6 shows for each λ the average improvement of the BEM method by the different functions for the scheduling process over five working days. It shows more or less the same pattern as the results of the scheduling process over three working days. The threshold in the model over three working days where the average improvement is increasing in λ is at a value of 18. In the scheduling process over five working days this threshold is at $\lambda = 17.5$. Notice that when $\lambda = 16$, functions {A,C} do not give an improvement. Function D gives an improvement of over 19% even for low values of λ.

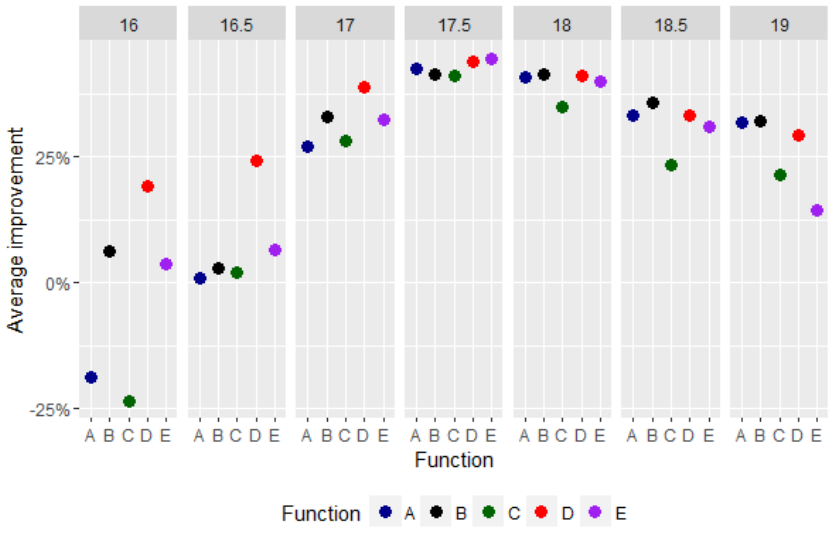


Figure 6.6: Average improvement by λ for different functions for the scheduling process over five working days.

Figure 6.7 shows for each λ the average improvement of the BEM method by the different functions for the scheduling process over ten working days. Functions {A, B, D, E} show the same pattern here as the results of the scheduling process over three and five working days. For functions {A,D} the threshold is at $\lambda = 17$. For functions {B,E} the threshold is at $\lambda = 17.5$. Only function C shows a different pattern than we have seen before. Up to $\lambda = 17$, the average improvement is increasing and it performs similar to the other functions. However, with $\lambda = 17.5$ the average improvement decreases from 44% to 11% after which the average improvement is again increasing in λ, but has the lowest average

improvement of all functions. Notice that when $\lambda = 16$, no function gives an improvement compared to the initial policy. Apparently, the initial policy performs good with this load of the system.

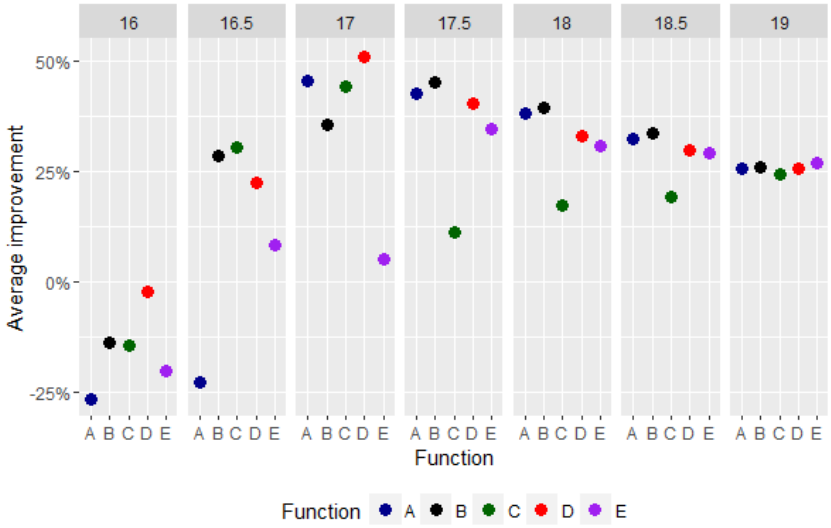
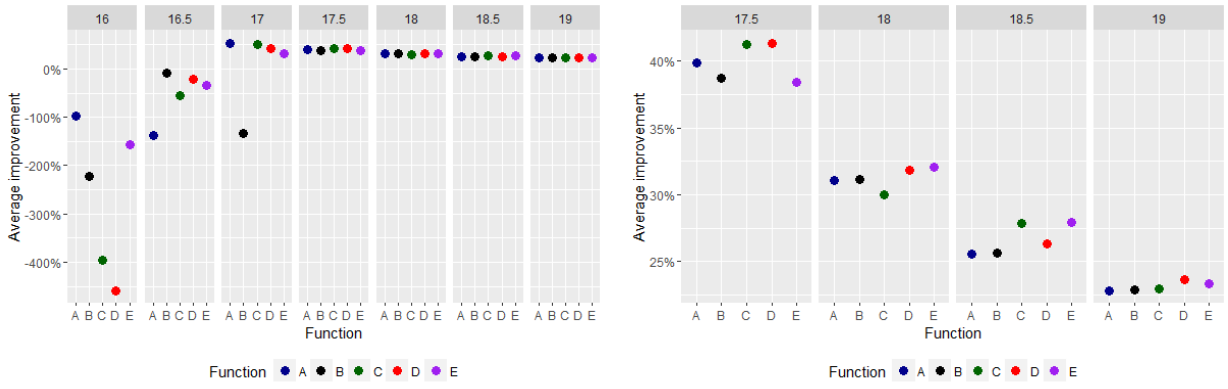


Figure 6.7: Average improvement by λ for different functions for the scheduling process over ten working days.

Figure 6.8 shows for each λ the average improvement of the BEM method by the different functions for the scheduling process over twenty working days. Remarkable are the results for $\lambda = 16$ in Figure 6.8a. The average improvement has a range of -100% for function A to -450% for function D. Also, for $\lambda = 16.5$ no function gives an improvement compared to the initial policy. Because of the large ranges in Figure 6.8a for $\lambda \leq 17$, the results for $\lambda > 17$ are hard to distinguish. Therefore, we show these results again in Figure 6.8b



(a) All values of λ

(b) $\lambda \in \{17.5, 18, 18.5, 19\}$

Figure 6.8: Average improvement by λ for different functions for the scheduling process over twenty working days.

without $\lambda \in \{16, 16.5, 17\}$. Again, for the higher values of λ it shows in general the same pattern as the results of the scheduling process over three, five and ten working days; as λ increases, the average improvement decreases.

Table 6.9: Average and variance of the improvement by function for five, ten and twenty working days.

Function	$N = 5$		$N = 10$		$N = 20$	
	Avg (%)	Var	Avg (%)	Var	Avg (%)	Var
A	22.5	4.53	19.4	8.36	29.8	0.44
B	27.5	2.29	27.8	3.35	29.6	0.38
C	18.2	4.27	18.8	2.92	30.5	0.47
D	32.8	0.74	28.7	2.50	30.8	0.48
E	24.7	2.35	16.4	5.62	30.4	0.32

Table 6.9 shows for the scheduling process over five, ten and twenty working days for each function the average and variance of the improvement of the BEM method relative to the initial policy, where we removed the results of $\lambda \in \{16, 16.5, 17\}$ from the scheduling process over twenty working days. Overall, function D, gives, just like the scheduling process over three days the best improvement, although all functions in the scheduling process over twenty working days perform almost equally well.

7 Conclusion

The goal of this research was to develop a model that prescribes the (near) optimal appointment date for a patient at the moment this patient makes his request. We modeled the scheduling process as an MDP and and we solved this MDP to optimality using value iteration. This resulted in an optimal value function of which the optimal policy (appointment date) can be derived. However, due to the curse of dimensionality, we were only able to solve our MDP to optimality for the scheduling process over three working days. It is computationally infeasible to solve our MDP to optimality for more than three working days and a reasonable available capacity on a day. Therefore, we employed an ADP technique, in order to derive an estimate of the optimal value function of our MDP. We explored two techniques, the BEM method and AVI and compared these techniques on the basis of an $M/M/s$ queue. Since AVI has a long running time if the reduced state space is large, we choose to apply the BEM method to our MDP. We simulated our initial policy to determine g and we keep track of the last S states that are visited. These states are added to what we call the list of important states with probability 0.1. We apply the k-means clustering algorithm to the list of important states to determine the set of representative states. From the scheduling process over three working days, we see no substantial difference between the number of last states (S) but K should not be chosen too small. From all combinations of the set of basis functions, the following combination outperforms all other combinations:

$$t + \sum_{p=1}^{3}\sum_{n=1}^{N} x_n^p + \sum_{n=1}^{N} x_n^2 * t + \sum_{n=1}^{N-1} x_n * x_{n+1} * t + \sum_{n=1}^{N-2} x_n * x_{n+1} * x_{n+2}$$

On average the long-run expected average cost obtained from this function deviates with 15.6% from g^*, with a variance of 0.71. The average improvement compared to the initial policy is 19.6% with variance 0.23. This function also outperforms all other combination in the scheduling processes over five, ten and twenty working days. In general it holds that the lower λ, the lower the low load of the system, the better our initial policy performs and hence, less improvement is obtained. The higher the load of the system, the worse our initial policy performs and the more important is our one-step policy improvement. But if λ reaches a certain value the load of the system becomes that high that it does not matter what policy is applied, since many patients have to be rejected.

7.1 Extensions to our model

As mentioned in the Introduction, HOTflo aims to develop an online application in which patients can schedule their own appointment at the outpatient clinic. The underlying algorithm of this application must show various date and time options of which patients can choose their own appointment date and time. The model we developed prescribes the

appointment date for a patient at the moment this patient makes his request, but this model needs some extensions, some of them are necessary, and others are desirable, in order to use this model in practice. These extensions are:

1. Appointment time (necessary);
2. Multiple appointment options (highly desirable);
3. Long term distribution (highly desirable);
4. Cancellations and no shows;
5. Follow-up appointments;
6. Dependency between outpatient clinics;
7. ...

Appointment time

Our model only prescribes the appointment date and not the appointment time. This is, of course, a necessary extension. There is plenty of literature available on intra-day scheduling, which is about the best timing and sequence of appointments on a given day, to gain inspiration about how to implement this into our model.

Multiple appointment options

Our model only prescribes one appointment date, but patients might have preferences for a certain date and/or time for the appointment. Therefore, the model should not return one (near) optimal appointment date, but m (near) optimal appointment dates. This extension is highly desirable. Patients should be able to choose from several options when they want to make an appointment.

Long term distribution

When our model reaches the end of the day, in other words if $t = T$, there is a shift in the schedule. The first day disappears from the schedule, the second day becomes the first day, the third day becomes the second day and so on, and finally a new empty day enters the schedule. By adding an empty day, we ignore the long term planning. It would be more realistic if not an empty day enters the schedule, but some kind of random number from, for instance, a Poisson distribution. This number represents the number of blocks that is already scheduled in the long-term. This extension is highly desirable in order to have a more realistic schedule.

Cancellations and no shows

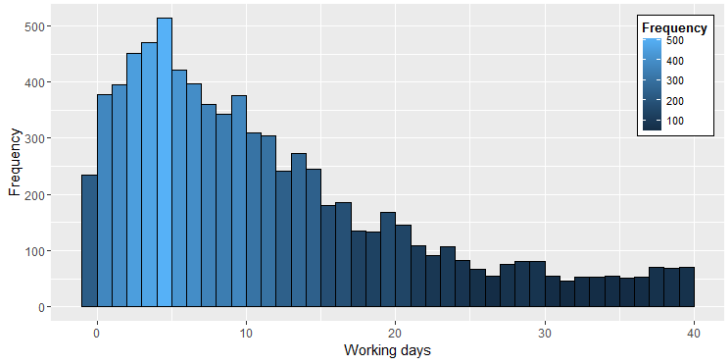
Patients can, after they are scheduled, do not show up or cancel their appointment. If this is not included in the model, than in the end more capacity remains available than necessary.

Follow-up appointments and dependency between outpatient clinics

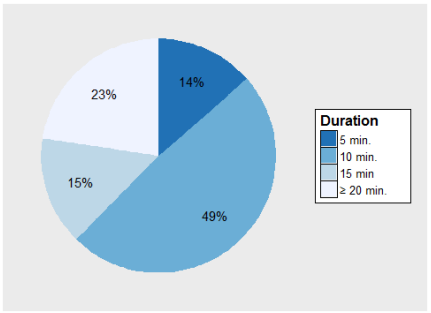
Patients may require multiple appointments on one or more days. Instead of a single appointment, combination appointments and appointment series are needed. Combination appointments imply that multiple appointments are planned on the same day, so that patients require fewer hospital visits. Appointment series imply that patients need to be scheduled for a number of treatment sessions. For example, patients being treated with radiation therapy for cancer.

A Patient characteristics

New Patients



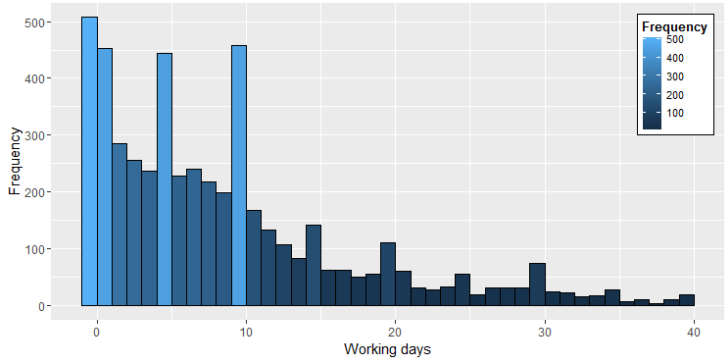
(a) Distribution of 'Working days'



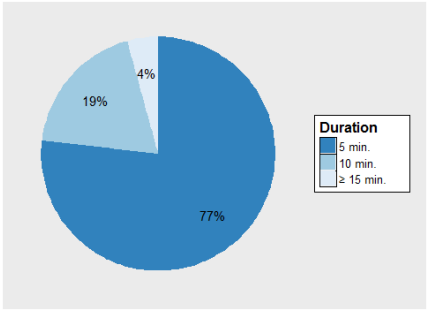
(b) Distribution of 'Duration'.

Figure A.1: New patients

Consults by telephone



(a) Distribution of 'Working days'



(b) Distribution of 'Duration'.

Figure A.2: Consults by telephone.

B List of Abbreviations and Symbols

Abbreviations

ADP Approximate Dynamic Programming
AVI Approximate Value Iteration
BEM Bellman Error Minimisation
MDP Markov Decision Process

Symbols

$\mathcal{A}_x$	The action space, the set of actions when the system is in state $x \in \mathcal{X}$
C	The total capacity on a working day
$c(x, a)$	The costs when taking action a in state x
D	The Bellman error for state $x \in \mathcal{X}$
g	The long run expected average cost
g^*	The optimal long run expected average cost
K	The number of clusters
k_{1j}	Costs for exceeding the maximum recommended waiting time for a patient of type $(1, j)$
N	The number of working days
$p(x, a, y)$	The probability of going from state x to state y when $a \in \mathcal{A}_x$ is chosen
p_{ij}	The probability that a patient of type (i, j) will arrive during a time interval
r	A vector of parameters used in the Bellman Error Minimisation
r_{ij}	The penalty cost of rejecting a patient of type (i, j)
S	The number of last states
T	The number of intervals of the current working day
V	The value function
V^*	The optimal value function
V_π	The value function for a fixed policy
$V_\pi(x)$	The value of a state x under policy π
$\tilde{V}$	The approximate value function
$w(x)$	The weight of the Bellman error made in state $x \in \tilde{\mathcal{X}}$.
w_{1j}	The maximum recommended waiting time for a patient of type $(1, j)$
w_{2j}	The specific appointment time for a patient of type $(2, j)$
$\mathcal{X}$	The state space, a set of states
$\tilde{\mathcal{X}}$	Subset of $\mathcal{X}$, representing the set of representative states
β_{ij}	The required scheduled service time for a patient of type (i, j)
γ	Perturbation parameter to overcome the problem of aperiodicity
λ	The arrival rate of patients

Π	The set of all policies
π	A policy
π^*	The optimal policy
$\pi(x)$	The action chosen in state x under policy π
ρ	The load of the system
ϕ	The set of basis functions used in the Bellman Error Minimisation

Bibliography

[1] O. Alagoz, H. Hsu, A. J. Schaefer, and M. S. Roberts. Markov decision processes: a tool for sequential decision making under uncertainty. *Medical Decision Making*, 30 (4):474–483, 2009.

[2] T. Cayirli and E. Veral. Outpatient scheduling in health care: a review of literature. *Production and operations management*, 12(4):519–549, 2003.

[3] T. Cayirli, K. K. Yang, and S. A. Quek. A universal appointment rule in the presence of no-shows and walk-ins. *Production and Operations Management*, 21(4):682–697, 2012.

[4] A. Erdelyi and H. Topaloglu. Approximate dynamic programming for dynamic capacity allocation with multiple priority levels. *IIE Transactions*, 43(2):129–142, 2010.

[5] S. A. Erdogan, A. Gose, and B. T. Denton. Online appointment sequencing and scheduling. *IIE Transactions*, 47(11):1267–1286, 2015.

[6] J. Feldman, N. Liu, H. Topaloglu, and S. Ziya. Appointment scheduling under patient preference and no-show behavior. *Operations Research*, 62(4):794–811, 2014.

[7] D. Gupta and B. Denton. Appointment scheduling in health care: Challenges and opportunities. *IIE transactions*, 40(9):800–819, 2008.

[8] D. Gupta and L. Wang. Revenue management for a primary-care clinic in the presence of patient choice. *Operations Research*, 56(3):576–592, 2008.

[9] G. C. Kaandorp and G. Koole. Optimal outpatient appointment scheduling. *Health Care Management Science*, 10(3):217–229, 2007.

[10] P. M. Koeleman and G. M. Koole. Optimal outpatient appointment scheduling with emergency arrivals and general service times. *IIE Transactions on Healthcare Systems Engineering*, 2(1):14–30, 2012.

[11] S. Li. *K-groups: A generalization of K-means by energy distance*. PhD thesis, Bowling Green State University, 2015.

[12] N. Liu, S. Ziya, and V. G. Kulkarni. Dynamic scheduling of outpatient appointments under patient no-shows and cancellations. *Manufacturing & Service Operations Management*, 12(2):347–364, 2010.

[13] J. Patrick, M. L. Puterman, and M. Queyranne. Dynamic multipriority patient scheduling for a diagnostic resource. *Operations research*, 56(6):1507–1525, 2008.

[14] W. B. Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

[15] M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2005.

[16] D. Roubos. The application of approximate dynamic programming techniques. PhD thesis, VU Amsterdam, the Netherlands, 2010.

[17] H. C. Tijms. *A first course in stochastic models*. John Wiley & Sons, 2003.