OR Spectrum
https://doi.org/10.1007/s00291-026-00858-x

ORIGINAL ARTICLE

Timeslot allocation for waiting list control

R. F. M. Vromans[1,2] · J. T. van Essen[3] · Y. M. van der Vlugt[3] · M. Carlier[2]

Received: 31 August 2023 / Accepted: 21 April 2026

Abstract

As pressure on the healthcare system increases, patients who require surgery experience longer access times to pre- and post-operative appointments and surgery. Hospitals can control their waiting lists by allocating timeslots to the different types of appointments they discern. To inform patients about their appointments in a timely manner, they need to make this decision several weeks in advance. However, the precise consequences of the timeslot allocation on the future waiting list are uncertain, as not all patients follow the same treatment pathway. Furthermore, as these planning decisions are made weeks in advance, they are based on an uncertain prediction of future waiting lists. In this paper, methods are developed with the aim to support hospitals in optimizing their timeslot allocation to reduce patient access times and utilize all available capacity in the outpatient department and operating room. The problem is modelled as a Markov decision process (MDP). However, as the state, decision and outcome spaces grow exponentially in size, even for a single surgeon, an exact solution cannot be determined. We thus compare four alternative solution methods to the static allocation method that is common in hospitals. Least-squares policy iteration is used to find an approximate solution, an (integer) linear program is formulated to solve a deterministic variant of the MDP, several heuristic decision rules are investigated, and a hybrid method is proposed that statically allocates a percentage of timeslots and then dynamically allocates the remaining timeslots with the linear program when sufficient information is available to effectively deal with variability. The solution methods are tested on a case study at the orthopedic care chain of the Sint Maartenskliniek hospital in the Netherlands. Simulation results indicate that in balanced capacity scenarios, static allocation achieves the highest performance when planning more than 4 weeks ahead. In contrast, in unbalanced systems, steering capacity toward patient groups incurring the highest costs yields better outcomes. The hybrid method offers flexibility as it can be adapted to both balanced and unbalanced situations. For the case study, we find that statically allocating 60% of timeslots and dynamically allocating the remainder 4 weeks in advance provides the best results in terms of meeting access time targets and efficient resource utilization.

Extended author information available on the last page of the article

Published online: 19 May 2026

Keywords Healthcare · Operations research · Integrated planning · Timeslot scheduling · Data-driven optimization

Abbreviations

ADP	Approximate dynamic programming
DA	Discharge appointment
EVI	Exact value iteration
FA	First appointment
FU	Follow-up appointment
ILP	Integer linear program
LP	Linear program
LSPI	Least-squares policy iteration
LSTD	Least-squares temporal differencing
MILP	Mixed integer linear program
MDP	Markov decision process
MSE	Mean squared error
OD	Outpatient department
OR	Operating room
SMK	Sint Maartenskliniek

1 Introduction

The COVID-19 pandemic highlighted the importance of sufficient healthcare capacity to society's well-being. Even outside pandemics, the Dutch healthcare system faces increasing pressure from an aging population and staff shortages (Nederlandse Zorgautoriteit 2020). This pressure causes longer access times to appointments and surgeries, lower quality of care, higher costs, and overworked staff. To prevent healthcare shortages, society can either increase resources or use current resources more efficiently. This paper focuses on using surgical specialists' time efficiently to treat as many patients as possible, on time.

Surgical specialists see patients in the outpatient department (OD) and in operating rooms (OR), also called operating theatres. Elective patients have multiple outpatient appointments of different types before and after surgery. Their care pathway begins with a relatively long first appointment (FA) to assess the nature of the complaint. In one or more follow-up appointments (FU) the surgeon decides on surgical eligibility and urgency, and determines the appropriate procedure. Post-surgery, a patient might have further follow-up appointments. The final step is a discharge appointment (DA). At every step in the care pathway, the surgeon assesses the urgency and the corresponding access time target.

Specialists typically work either full or half days at the OD and OR, commonly referred to as sessions. The number of OD and OR sessions per week constitutes the session plan, which determines how the access times for the OD and OR will develop. Surgeons can, for example, reduce their OR waiting list by temporarily scheduling more OR sessions at the cost of OD sessions. However, an increase in surgeries eventually generates higher demand for discharge appointments. When more

capacity is needed for discharge appointments, this reduces capacity and lengthens access times for other appointment types.

After the sessions are planned, the next step is appointment scheduling, typically based on timeslots. To control access for the different appointment types, hospitals can decide how many appointments of each type to include in each session. This is called timeslot allocation. Furthermore, hospitals can define timeslot schedules that also determine the order, start times, and durations of timeslots.

This paper explores how, given a session plan, the allocation of timeslots to appointment types can be used to control waiting lists per appointment type. We treat surgeries as appointments within this framework. We focus specifically on timeslot allocation, while excluding sequencing and duration, as these primarily affect performance aspects other than access times. We refer to the problem of determining how many of each appointment type to allocate as the *timeslot allocation problem*. Our aim is to develop a model for this problem and compare different solution methods.

We contribute to the research field of appointment scheduling, and specifically timeslot scheduling, in four ways. First, we build on a model for timeslot allocation, presented by Hulshof et al. (2013, 2016), and extend the objective function with rewards for appointments and surgeries to stimulate the optimal use of the available capacity. Second, we compare the performance of multiple solution methods. We evaluate four decision rules, formulate a similar linear program (LP) and integer linear program (ILP) to Hulshof et al. (2013), and investigate a form of approximate dynamic programming (ADP) suitable for infinite-time-horizon problems: least-squares policy iteration (LSPI). We evaluate these methods on real-life data from the Sint Maartenskliniek (SMK), and compare their performance against the static method currently used by the hospital. Third, we implement the concept of planning ahead. To accommodate timely booking of appointments, timeslot schedules need to be generated weeks in advance. Perfect information about the future state of the system for which we make a planning decision is then unavailable. In this paper, we develop a method to predict that future state. Furthermore, we compare the performance of the proposed solution methods as the planning horizon increases. Fourth, to the best of our knowledge, the variant of LSPI used in this paper has not been used previously in literature. Lagoudakis and Parr (2003) introduce an off-policy variant using an approximate Q-function and Powell (2019) presents an on-policy variant using an approximate value function. We introduce and implement an off-policy variant using an approximate value function: off-policy LSPI-V.

In Sect. 2, we provide an overview of related research in the field of healthcare planning. Section 3 presents our model for the timeslot allocation problem. Section 4 introduces the solution methods that are used. Section 5 describes the small test instance, large test instance, and the case study SMK instance we use to test the solution methods. We discuss the results of the solution methods per instance in Sect. 6 and present our conclusions and recommendations in Sect. 7.

2 Literature review

In this section, we discuss relevant research related to the timeslot allocation problem. Section 2.1 presents previous research at the SMK on session planning for surgeons. Section 2.2 discusses literature on outpatient appointment scheduling. Section 2.3 describes previous research on a similar problem to the timeslot allocation problem, patient admission planning, on which our approach is based.

2.1 OD and OR session planning

As discussed, the timeslot allocation problem allocates available capacity in OD and OR sessions to specific appointment types. Therefore, we first discuss research on OD and OR session planning as the OD and OR session planning will be input to our model. Tsai (2017) and Hattingh (2019) present work at the Sint Maartenskliniek (SMK) to determine the optimal planning of OD and OR sessions for orthopedic surgeons. They aim to stabilize internal access times to surgery, reduce unused OR time, and meet each surgeon's yearly production targets. Access times are stabilized by reducing fluctuations around a target OR waiting list length. Both model the patient pathway at SMK as a Markov decision process, but apply different methods to solve the problem.

Tsai (2017) develops a detailed model that includes sessions and appointments in the outpatient department, radiology, preoperative screening, and surgery. Stochastic dynamic programming is applied to determine an optimal allocation, but this is impractical due to high computation times for calculating transition probabilities between states. Even after significantly reducing the state space through simplifying assumptions, the model remains computationally expensive. Tsai (2017) recommends excluding the planning of the radiology department and mentions approximate dynamic programming as a possible solution method.

Hattingh (2019) formulate a similar Markov decision process (MDP) model to Tsai (2017), following the recommendation to exclude the planning of the radiology department. They compare test cases in which the effects of OD and OR sessions on waiting lists are modeled as either stochastic or deterministic. Replacing the stochastic parameters by deterministic averages results in similar solutions for the test cases and significantly reduced the runtime. This allows for the formulation of an integer linear program to optimize the model. The ILP successfully determines the number of OD and OR sessions per week for each surgeon over a 2-week planning period. These outcomes are input for our timeslot allocation problem, in which the available time in the OD and OR sessions is assigned to the different appointment types.

2.2 Outpatient appointment scheduling

Outpatient appointment scheduling has been intensively studied; see Ahmadi-Javid et al. (2017) and Ala and Chen (2022) for an overview. This section discusses the research related to our timeslot allocation problem.

Erdelyi and Topaloglu (2011) consider the case where there is a fixed amount of daily processing capacity. Each day, jobs with different priorities arrive randomly.

The decision to optimize is which jobs are scheduled on which days. The waiting cost of a job depends on its priority level. The objective is to minimize the total expected waiting cost over a finite planning horizon. This setting differs from our setting, because jobs arriving on different days are considered to be independent. Therefore, in Erdelyi and Topaloglu (2011), no transition probabilities between states need to be taken into account. The problem is solved using approximate dynamic programming, where the value function is approximated by decomposing the problem per day. Patrick et al. (2008) consider a similar problem for which approximate dynamic programming techniques are used to formulate an approximate linear program, which is solved using column generation. The resulting policy is tested through simulation.

Nunes et al. (2009) consider a problem similar to timeslot allocation, in which patients from different specialties are admitted during fixed planning periods. Each patient has a treatment pattern using resources during their admission. The problem is modelled as a Markov decision process, where the action is the number of patients to admit for each speciality in the next planning period. The state is the number of patients for each specialty with a specific treatment pattern during the last period. The transition probabilities indicate the probability that a patient receives a different treatment pattern. The objective is to be as close as possible to the target resource utilization. This objective is modelled using a cost function that includes costs for underutilization (below target), overutilization (above target), and overcapacity (demand exceeds capacity). Only small instances are solved using value iteration, and developing solution methods for larger instances is left for future work.

Nguyen et al. (2015) determines the capacity required to schedule first and follow-up appointments while respecting access time targets. For both appointment types, a constant discharge rate is assumed. This problem differs from the setting considered in our paper, as we consider the capacity to be given and the access time to be minimized. Aslani et al. (2021) extends upon the work of Nguyen et al. (2015) by considering the demand for first appointments to be uncertain. They develop a robust optimization model that minimizes the maximum capacity required for the worst-case realization of demand.

Bakker and Tsui (2017) dynamically allocate capacity to multiple specialists and activities. The activities consist of first appointments, follow-up appointments, and surgery. The capacity is not assigned optimally but is approximated by a greedy heuristic that aims to minimize variability in roster activities. The approach is evaluated using simulation.

Laan et al. (2018) optimise appointment scheduling with respect to access time, taking into account fluctuating patient arrivals and physicians' unavailability. Follow-up appointments are modeled as a patient type, which means they are treated independently, whereas in practice this is not the case. They formulate a stochastic mixed integer programming problem, and approximate its solution using two different approaches: (1) a mixed integer programming approach that results in a static appointment schedule, and (2) Markov decision theory, which results in a dynamic scheduling strategy. They apply the methodologies to a case study of the surgical outpatient clinic of the Jeroen Bosch Hospital.

Deglise-Hawkinson et al. (2018) develop an appointment template for an integrated care environment in which multiple specialties serve multiple patient types.

This template only considers the number of new patients to schedule. For follow-up downstream appointments, capacity is reserved rather than planned. The problem is formulated as a queuing network optimization problem. An approximate solution is determined by solving it as a deterministic linear optimization problem.

In summary, most of the discussed literature on appointment scheduling only considers independent appointments, first appointments, or determines the needed capacity based on demand. The research that does consider the full chain of appointments and surgeries uses a greedy heuristic. We contribute to literature by presenting exact and approximate methods for planning a full chain of appointments and by applying the developed methods for real-life sized settings.

2.3 Timeslot allocation

A very similar problem to the one under consideration in this paper was investigated by Hulshof et al. (2013, 2016). They describe a general model for tactical resource allocation and patient admission planning in health care, which is formulated as a mixed integer linear program (MILP) in Hulshof et al. (2013) and as an MDP in Hulshof et al. (2016).

Hulshof et al. (2013) formulate the problem as a mixed integer linear program. Since an MILP cannot solve stochastic problems, the transition probabilities are modeled as deterministic fractions q_{ij} of patients moving from one stage (or queue) to another. The objective is "to achieve equitable access and treatment duration for patient groups and to serve the strategically agreed number of patients". In the objective function, this is approximated by minimizing the weighted sum of patients waiting in each queue over all access times and time periods. The authors apply an iterative scheme to determine the weights in the objective function, thereby optimizing the hospital's true performance targets. Relatively large instances (fifty queues, two resources, and six time periods) took, on average, 4 min to compute, with an integrality gap of 0.01%. Results show that the model allocates resources more equitably over time and that the number of patients served is closer to the strategically set target. The authors recommend implementing this model using a rolling-horizon approach, i.e., only applying decisions for earlier time periods while considering expected effects on later time periods. The model is relatively easy to adapt, for example, with more constraints or a different objective function.

Hulshof et al. (2016) formulate the problem as an MDP. They allocate capacity to patient groups to provide equitable access to treatment and meet the hospital's strategic goals. Computing the exact solution to this MDP using dynamic programming is only possible for very small instances, as the state, decision, and outcome spaces grow exponentially in size [the three curses of dimensionality (Powell 2019)]. Therefore, approximate dynamic programming (ADP) is applied, a form of approximate, model-free, on-policy value iteration. The algorithm overcomes intractability by using a post-decision state and a linear parametrization of the value function using basis functions to approximate future costs. An ILP is used to efficiently find an optimal action given the current approximate value function and problem constraints. Various basis functions are proposed and compared using regression analysis. Recursive least squares is used to update the parameter vector used for the value function.

The authors use small test instances to show that the ADP algorithm provides a close approximation to the outcome of exact dynamic programming in a reasonable time. For large instances (forty queues, four resources, eight time periods), the algorithm takes approximately 1 h to converge to a parameter vector. The resulting policy outperforms two proposed greedy heuristics on test instances. The authors conclude that ADP is an appropriate technique for tactical healthcare planning, but do not compare the performance with the approach developed in Hulshof et al. (2013).

We apply the model by Hulshof et al. (2016) to the timeslot allocation problem. For this purpose, we use an infinite rather than a finite time horizon, extend the objective function to maximize the number of patients treated, and employ a different approximation method. Furthermore, we apply it to a case study, compare the approximate method with several exact and heuristic methods, and evaluate the impact of planning ahead over a number of time periods.

3 Problem formulation

In this section, we describe the model for the timeslot allocation problem. Section 3.1 describes the problem and discusses our assumptions. Section 3.2 describes the sets, parameters, functions, states, and actions defining the model. Section 3.3 proposes the objective function, and Sect. 3.4 describes how we incorporate the concept of planning ahead. All notation used in this section is summarized in Table 1.

3.1 Problem definition

We aim to model how allocating timeslots to different appointment types affects the waiting lists for a single surgeon. The inputs are the number of OD and OR sessions in the planning period as derived from the activity plan. Their capacity is expressed in fixed time units (e.g., 5 or 10 min). The model further requires surgeon-specific parameters and the current state of the waiting lists. It then allocates a number of OD and OR timeslots per appointment type. Timeslots require an integer number of time units. For example, with 5-min time units, a 15-min timeslot occupies 3 units and a 20-min timeslot occupies 4 units.

Since treating patients not only affects the current waiting lists but also future states, as patients progress through their treatment, it is important to take future consequences of current decisions into account. Additionally, since no two patients are the same, the consequences are uncertain, making this a sequential decision problem under uncertainty. For this reason, we model the problem as a Markov decision process (MDP).

We use the following assumptions:

1. The total number of available OD and OR time units is given for each time period. There is sufficient capacity of other resources (locations, materials) to utilize all timeslots.
2. The duration of an OD and OR timeslot is fixed per surgeon. In practice, this is not necessarily the case for OR appointments, as the time required depends on the

Table 1 Sets, variables, parameters, and functions used

Sets	
$\mathcal{T}$	Time periods
$\mathcal{J}$	Queues
$\mathcal{W}_j$	Waiting times for queue $j \in \mathcal{J}$
$\mathcal{R}$	Resource types
$\mathcal{S}$	State space
$\mathcal{A}$	Action space
Variables	
$s_{j,w}$	Sub-state: number of patients in queue $j \in \mathcal{J}$ who have been waiting for $w \in \mathcal{W}_j$ time periods
$a_{j,w}$	Sub-action: number of patients to treat from queue $j \in \mathcal{J}$ who have been waiting for $w \in \mathcal{W}_j$ time periods
Parameters	
u_j	Access time target of patients in queue $j \in \mathcal{J}$
m_j	Minimum access time of patients in queue $j \in \mathcal{J}$
$\eta_{r,t}$	Available number of time units for resource $r \in \mathcal{R}$ at time $t \in \mathcal{T}$
$\zeta_{j,r}$	Number of time units of resource $r \in \mathcal{R}$ required by patient in queue $j \in \mathcal{J}$
$q_{i,j}$	Probability that a patient transfers from queue $i \in \mathcal{J}$ to queue $j \in \mathcal{J}$
$\lambda_{j,t}$	Number of new patients to enter queue $j \in \mathcal{J}$ at time $t \in \mathcal{T}$
r_j	Reward for treating a patient from queue $j \in \mathcal{J}$
$c_{j,w}$	Cost of a patient in queue $j \in \mathcal{J}$ who has been waiting for $w \in \mathcal{W}_j$ time periods
d_j	Allowed maximum access time of patients in queue $j \in \mathcal{J}$ before costs are incurred
ω_j	Relative weight of queue $j \in \mathcal{J}$
γ	Discount factor
p	Planning horizon: the number of time periods to plan ahead
Functions	
$f_{i,j,v}\,(a_{i,v})$	Transition function of patients from queue $i \in J$ to queue $j \in J$ after waiting $v \in \mathcal{W}_j$ time periods depending on action $a_{i,v}$
$C(s, a)$	Contribution function for taking action a in state s
$\pi(s)$	Policy function defining which action to take in state s
$V(s)$	Value function of state s

complexity of the surgery. We use the rounded average of surgery durations per specific surgeon.

3. When patients are transferred to a queue, they can be treated as soon as the minimal access time for that queue is met. This means that intermediate steps like diagnostics and pre-operative screening (POS) have to be completed in time for the next appointment or surgery. Diagnostic and screening capacities are outside the scope of this research.

4. The considered surgeon is not affected by, or dependent on, the activities of other surgeons. In the SMK, the majority of patients stay with the same surgeon for all consultations and surgery.
5. The selected action is assumed to be implemented exactly as planned. In practice, however, the patient with the highest contribution may not be available at the required time, appointments can be canceled, and patients with higher urgency may arrive later when no time slots are available. Nevertheless, this assumption is sufficient for comparing the quality of the solution methods.
6. All appointments are realized (zero no-shows), and appointment slots for which there are no patients on the waiting list remain unused. These assumptions also suffice when we use the model to compare the quality of solution methods.

3.2 States and actions

Our model is based on the model by Hulshof et al. (2016). Different from Hulshof et al. (2016), we consider an **infinite time horizon**. This allows us to take into account all future consequences of actions. Unless stated otherwise, we set the set of **decision epochs** $\mathcal{T}$ to $\mathbb{Z}^+$. The patient queues are given by the set $\mathcal{J}$. Another addition to the work of Hulshof et al. (2016) is that each queue $j \in \mathcal{J}$ has a given access time target denoted by u_j. The number of time periods that a patient has been waiting in a queue is indicated by the value $w \in \mathbb{Z}^+$. For some solution methods, it may be necessary to upper bound the waiting time for queue $j \in \mathcal{J}$ with some value W_j. Therefore, we introduce the set $\mathcal{W}_j$, which is equal to either $\mathbb{Z}^+$ or $\{0, \ldots, W_j\}$ depending on the solution method. Let sub-state $s_{j,w}$ denote the number of patients waiting in queue $j \in \mathcal{J}$ for $w \in \mathcal{W}_j$ time periods. **State** s is defined as the vector of all such sub-states,

$$s = (s_{j,w})_{j \in \mathcal{J},\, w \in \mathcal{W}_j}.$$

The size of a state $|s|$ is the sum of its sub-states, and represents the total number of patients in the system at a certain moment in time. The state space of all possible states is denoted by $\mathcal{S}$.

The sub-action $a_{j,w} \in \mathbb{Z}^+$ denotes the number of patients selected for treatment from sub-state $s_{j,w}$. **Action** a is defined as the vector of all such sub-actions,

$$a = (a_{j,w})_{j \in \mathcal{J},\, w \in \mathcal{W}_j}.$$

Note that we cannot treat more patients than available in queue $j \in \mathcal{J}$, waiting for $w \in \mathcal{W}_j$ time periods, i.e. $a_{j,w} \leq s_{j,w}$. Also, each queue $j \in \mathcal{J}$ has a minimal access time denoted by m_j, which means that patients from queue $j \in \mathcal{J}$ cannot be treated when their waiting time w is less than m_J, i.e., $a_{j,w} = 0$ for $w < m_j$. Furthermore, the required capacity of resource $r \in \mathcal{R}$ to treat patients at time $t \in \mathcal{T}$, can not exceed the available capacity $\eta_{r,t}$ at this time. We consider a set of resources $\mathcal{R} = \{\text{OD}, \text{OR}\}$ consisting of the outpatient department and operating room. The resource capacity $\eta_{r,t}$ indicates the number of time units available for resource $r \in \mathcal{R}$ at time $t \in \mathcal{T}$. Parameter $\zeta_{j,r}$ indicates how many time units of resource $r \in \mathcal{R}$ are required to treat a patient in queue $j \in \mathcal{J}$. Given the capacity restriction, the action space of all

feasible actions $\mathcal{A}(t)$ is dependent on time $t \in \mathcal{T}$. The set of feasible actions at time $t \in \mathcal{T}$ given a state s is denoted by $\mathcal{A}(s,t)$ and is given by:

$$\mathcal{A}(s,t) = \left\{ a : \begin{array}{ll} a_{j,w} \leq s_{j,w}, & \forall j \in \mathcal{J}, w \in \mathcal{W}_j, \\ a_{j,w} = 0, & \forall j \in \mathcal{J}, w \in \mathcal{W}_j, w < m_j, \\ \sum\limits_{j \in \mathcal{J}} \sum\limits_{w \in \mathcal{W}_j} \zeta_{j,r} a_{j,w} \leq \eta_{r,t}, & \forall r \in \mathcal{R}. \end{array} \right\}. \quad (1)$$

When treated, a patient in queue $i \in \mathcal{J}$ either transfers to queue $j \in \mathcal{J}$ with **transition probability** $q_{i,j}$, or leaves the system with probability $1 - \sum_{j \in \mathcal{J}} q_{i,j}$. Let 0 denote the exit queue. When a patient is not treated, the waiting time $w \in \mathcal{W}_j$ of this patient is incremented by one. The number of new patients arriving from outside the system to queue $j \in \mathcal{J}$ at time $t \in \mathcal{T}$ is given by **exogenous information** $\lambda_{j,t}$. Given a state s and action a, the resulting next state s' is given by:

$$s'_{j,w} = \begin{cases} \lambda_{j,t+1} + \sum\limits_{i \in \mathcal{J}} \sum\limits_{v \in \mathcal{W}_j} f_{i,j,v}(a_{i,v}), & w = 0, \forall j \in \mathcal{J}, \\ s_{j,w-1} - a_{j,w-1}, & 1 \leq w \leq W_j - 1, \forall j \in \mathcal{J}, \\ \sum\limits_{v=W_j-1}^{W_j} s_{j,v} - a_{j,v}, & w = W_j, \forall j \in \mathcal{J}, \end{cases} \quad (2)$$

where $f_{i,j,v}(a_{i,v})$ denotes the number of patients transitioning from queue $i \in \mathcal{J}$ to queue $j \in \mathcal{J} \cup \{0\}$ after waiting $v \in \mathcal{W}_j$ time periods, given action $a_{i,v}$. For fixed *i* and *v*, the transitions are defined as

$$\left(f_{i,j,v}\left(a_{i,v}\right)\right)_{j \in \mathcal{J} \cup \{0\}} \sim \text{Multinomial}\left(a_{i,v}, \left(q_{i,j}\right)_{j \in \mathcal{J} \cup \{0\}}\right).$$

In implementation, this multinomial draw is obtained via inverse transform sampling: for each of the $a_{i,v}$ patients, a uniform random variable $u \sim \mathcal{U}(0,1)$ is drawn and assigned to the smallest index *j* such that $\sum_{k \leq j} q_{i,k} \geq u$.

3.3 Contribution function

The objective function of our MDP model balances rewards obtained from treating patients with costs incurred when patients are not treated on time. The model, therefore, maximizes the difference between rewards and costs. Following Powell (2019), we refer to this quantity as the contribution. Accordingly, we refer to the objective function as the **contribution function**.

The first part of our contribution function, which is an addition to the work of Hulshof et al. (2016), is to maximize the number of patients treated, as this increases both the patients' health benefits and resource utilization. Therefore, a reward r_j is assigned for every patient treated from queue $j \in \mathcal{J}$. The total reward for an action a is given by:

$$\sum_{j\in\mathcal{J}}\sum_{w\in\mathcal{W}_j} r_j \cdot a_{j,w}.$$

The second part of our contribution function is to minimize the weighted sum of patients who are not treated on time. The number of patients that is not selected for treatment and has been waiting $w \in \mathcal{W}_j$ time periods in queue $j \in \mathcal{J}$ is given by $s_{j,w} - a_{j,w}$. The allowed maximum access time d_j determines whether patients are treated on time. We define the cost of letting a patient in queue $j \in \mathcal{J}$ wait for $w \in \mathcal{W}_j$ time periods by $c_{j,w}$. If the waiting time $w \in \mathcal{W}_j$ has not yet exceeded the allowed maximum access time d_j, the patient from queue $j \in \mathcal{J}$ is not late yet, and the cost should be zero: $c_{j,w} = 0$ for $w < d_j$. Furthermore, $c_{j,w}$ should be non-decreasing in $w \in \mathcal{W}_j$ (as higher access times should be penalized more heavily), and non-increasing in u_j (as a higher value of the allowed access time u_j indicates lower urgency). Accordingly, we define the cost function for a patient in queue $j \in \mathcal{J}$ waiting for $w \in \mathcal{W}_j$ as $c_{j,w} = \omega_j \cdot \frac{w}{u_j}$ for $w \geq d_j$. Here, ω_j is a parameter determining the weight of each queue relative to the other queues. The total cost of an action *a* applied in state *s* is given by:

$$\sum_{j\in\mathcal{J}}\sum_{w\in\mathcal{W}_j} c_{j,w} \cdot (s_{j,w} - a_{j,w}).$$

This means that a patient from queue $j \in \mathcal{J}$ who is chosen to be treated after waiting for $w \in \mathcal{W}_j$ time periods incurs a total cost of $\sum_{v=0}^{w} c_{j,v}$, which is equal to $\sum_{v=u_j}^{w} c_{j,v}$ since $c_{j,v} = 0$ when $v < d_j$.

Combining the two parts gives the following contribution function that we aim to maximize:

$$C(s,a) = \sum_{j\in\mathcal{J}}\sum_{w\in\mathcal{W}_j} r_j \cdot a_{j,w} - c_{j,w} \cdot (s_{j,w} - a_{j,w}). \tag{3}$$

Note that a low reward relative to the cost of waiting may result in the optimal solution accepting only a small number of new patients to ensure that waiting times do not exceed the allowed maximum access time; a high reward may result in longer waiting lists to ensure full use of available resources.

The aim is, given initial state $s \in \mathcal{S}$, to find an optimal policy π^* maximizing the value function. Here, policy π defines for each state *s* which action *a* to take, i.e., $\pi(s) = a$. The discount factor is given by γ.

$$V^*(s) = \max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t\in\mathcal{T}} \gamma^t C\left(s_t, \pi\left(s_t\right)\right) | s_0 = s\right].$$

3.4 Planning ahead

Until this point, we have assumed that an action *a* is selected at time $t \in \mathcal{T}$ with perfect knowledge of state *s* at time $t \in \mathcal{T}$. However, planning decisions are made in

advance to timely inform surgeons of their schedules and patients of their appointments. The number of time periods to plan ahead is defined as the *planning horizon* and is denoted by the parameter p. This implies that at time $t \in \mathcal{T}$, we must determine an action a for time $t + p$ based on incomplete knowledge of state s at time $t + p$. To obtain an estimate of state s at time $t + p$, we can use the state s at time t and the actions for time $t, t + 1, \ldots, t + p - 1$, as these actions have already been determined in previous time periods. When introducing the various solution methods in the next section, we do not explicitly account for the planning horizon p. Planning ahead can be applied to any of the introduced solution methods by not deciding on an action for the next time period, but for p time periods ahead. See Sect. 6.4 for the effect of planning ahead on the solution quality.

4 Solution methods

As we have modelled the problem as a Markov decision process, the solution takes the form of a policy: a function that specifies which action to take for each state. An optimal policy maximizes the expected contribution gained by following that policy over time. Such an optimal policy can be found through exact value iteration (EVI), which we discuss in Sect. 4.1. However, as the state, decision, and outcome spaces grow exponentially in size, we investigate several approximation methods and heuristics. These are least-squares policy iteration (Sect. 4.2), integer linear programming (Sect. 4.3), and decision rules (Sect. 4.4). Section 4.5 describes the current (static) allocation method used by SMK that we use as a benchmark. Finally, Sect. 4.6 discusses a hybrid method that combines the static allocation with one of the dynamic policies.

4.1 Exact value iteration

The exact solution to an MDP can be found using dynamic programming. For realistic instances of the timeslot allocation problem, this is infeasible as the state, decision, and outcome spaces grow exponentially in size. Still, finding the exact solution for small test instances is useful for comparing the performance of the other proposed solution methods. We also use this method to compare different basis functions for least-squares policy iteration.

Finding an exact solution requires solving the Bellman equation:

$$V^*(s) = \max_{a \in \mathcal{A}(s)} \left(C(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V^*(s') \right), \quad \forall s \in \mathcal{S}. \tag{4}$$

This equation can be solved using the exact value iteration algorithm for infinite horizon optimization introduced by Powell (2019). Computing the transition probability $P(s'|s,a)$ from state s to state s' under action a is intractable for our problem due to the large number of possible transitions. Instead, a computationally simpler but

equivalent approach is to sum over all possible realizations of these transitions rather than over all possible next states.

Let $I = \{(i, j) : q_{ij} > 0, i, j \in \mathcal{J} \cup \{0\}\}$ be the pairs of indices of queues between which patients can transition, including the 'exit' queue 0. Now let $X = (X_{i,j})_{(i,j)\in I}$ be a random variable denoting the number of patients transitioning from queue i to queue j. Let a_i be the action indicating the number of patients to treat from queue $i \in \mathcal{J}$. Since a_i patients are treated from queue $i \in \mathcal{J}$ and each patient independently transitions to queue $j \in J$ with probability q_{ij}, the number of patients transitioning from queue i to j follows a binomial distribution: $X_{ij} \sim B(a_i, q_{ij})$. Now, let $\mathcal{X}(a)$ be the set of possible realizations of transitions resulting from action a, i.e.,

$$X(a) = \left\{ x \in \mathbb{Z}^{|I|} \ : x_{ij} \geq 0, \forall (i,j) \in I \text{ and } \sum_{j\in\mathcal{J}} x_{ij} = a_i, \forall i \in \mathcal{J} \right\}.$$

Using the transition function $T(s, a, x)$, which provides the new state given previous state s, action a, and (a realization of) exogenous information x, Equation (4) can now be written as:

$$\begin{aligned} V^*(s) &= \max_{a\in\mathcal{A}(s)} \left(C(s,a) + \gamma \sum_{x\in\mathcal{X}(a)} P(X = x|a) V^*(T(s,a,x)) \right) \\ &= \max_{a\in\mathcal{A}(s)} \left(C(s,a) + \gamma \sum_{x\in\mathcal{X}(a)} \left(\prod_{(i,j)\in I} P\left(X_{ij} = x_{ij}|a_i\right) \right) V^*\left(T(s,a,x)\right) \right) \\ &= \max_{a\in\mathcal{A}(s)} \left(C(s,a) + \gamma \sum_{x\in\mathcal{X}(a)} \left(\prod_{(i,j)\in I} \binom{a_i}{x_{ij}} q_{ij}^{x_{ij}} (1-q_{ij})^{a_i - x_{ij}} \right) V^*\left(T(s,a,x)\right) \right), \end{aligned}$$

which can be solved using the exact value iteration algorithm.

4.2 Least-squares policy iteration

Real-life sized instances of the timeslot allocation problem are intractable to solve exactly. Therefore, we apply least-squares policy iteration (LSPI) as an approximation method. LSPI scales to large state spaces by approximating the value function with a set of basis functions. As a result, the value of many states can be estimated simultaneously rather than computed individually. Combining the off-policy LSPI-Q by Lagoudakis and Parr (2003) and on-policy LSPI-V by Powell (2019), we propose an off-policy LSPI-V algorithm. The value function (V) version of LSPI is required because the action space is prohibitively large to approximate the Q-function.

The value of a state is approximated as a linear combination of basis functions $\phi(s)$, with a parameter vector θ weighting their contributions. Each basis function encodes a feature of the current state that is predictive of the system's future evolution and potential value. The algorithm determines a policy by learning a parameter vector θ such that the weighted sum of these features approximates the future contri-

bution of the current state. By combining multiple basis functions, the approximation can account for different drivers of future value and their interactions. We use least-squares temporal differencing (LSTD), see Bradtke and Barto (1996), as opposed to recursive LSTD, to reduce the computational load.

Algorithm 1 presents the off-policy LSPI-V algorithm. The procedure starts by initializing the parameter vector $\theta_0 \in \mathbb{R}^F$ and setting the iteration counter $n = 0$ (lines 2–3). In addition, small positive constants ϵ and δ are chosen to ensure numerical stability and define the convergence criterion (line 3). A dataset of sampled states $D = \{s_1, \ldots, s_M\}$ is then generated (line 4). Based on the current parameter vector, a policy is defined (line 5). The algorithm proceeds iteratively (lines 7–18) until convergence, i.e., until the change in θ between iterations falls below a predefined threshold δ (line 7). At the start of each iteration, the auxiliary quantities are initialized: the vector $b_0 = 0$ (line 9) and the matrix $A_0 = \epsilon I$ (line 10), where I is the $F \times F$ identity matrix. Within each iteration, all M sampled states are processed sequentially (lines 11–16). For a given sample s_m, the current policy selects an action a_m (line 12). Next, exogenous information x_m is sampled (line 13), and the transition function is used to obtain the next state s'_m (line 14). Rather than updating the parameter vector θ after each transition, the algorithm accumulates information in vector b and the matrix A using LSTD updates (line 15). Specifically, the vector b aggregates the immediate contributions, while the matrix A captures the relationship between basis functions of consecutive states. After processing all M samples, the parameter vector is computed as $\theta = A_M^{-1} b_M$ (line 17). Because the number of basis functions F is limited, the matrix inversion is computationally tractable. Finally, the updated parameter vector defines a new policy, since the policy (line 5) depends on the value function approximation. This process repeats until convergence, after which the algorithm returns the approximate policy π (line 20).

1 Initialization:
2 $\theta_0 = \mathbf{0} \in \mathbb{R}^F$, $n = 0$
3 Choose small constants $\epsilon, \delta > 0$
4 Generate a set of sampled states $D = \{s_1, \ldots, s_M\}$
5 Define the policy: $\pi(s|\theta) = \arg\max_{a \in \mathcal{A}(s)} \left(C(s,a) + \gamma \theta^\top \phi(\overline{T}(s,a))\right)$
6 Learn parameters θ_n:
7 **while** $||\theta_n - \theta_{n-1}|| \geq \delta$ *or* $n = 0$ **do**
8 Increment n: $n = n + 1$
9 Initialize $b_0 = \mathbf{0} \in \mathbb{R}^F$
10 Initialize $A_0 = \epsilon I$, using $F \times F$ identity matrix I
11 **for** $m = 1, \ldots, M$ **do**
12 Compute action $a_m = \pi(s_m|\theta_{n-1})$
13 Obtain a sample of exogenous information x_m from the simulation
14 Determine the next state $s'_m = T(s_m, a_m, x_m)$
15 Perform the LSTD update: $b_m = b_{m-1} + C(s_m, a_m)\phi(s_m)$, $A_m = A_{m-1} + \phi(s_m) \cdot (\phi(s_m) - \gamma\phi(s'_m))^\top$
16 **end**
17 Update θ_n: $\theta_n = A_M^{-1} b_M$
18 **end**
19 $N = n$
20 **return** *the approximate policy* $\pi(s|\theta_N)$

Algorithm 1 Off-policy LSPI-V for value function approximation

The following sections discuss the main components of LSPI in more detail. Section 4.2.1 describes how in line 4, states and exogenous information are sampled in order to construct the dataset used to approximate the value function. Section 4.2.2

discusses the linear program that is used in line 12 to determine the corresponding action for each sampled state. Finally, Sect. 4.2.3 discusses several possible basis functions ϕ to represent the state space.

4.2.1 Sampling of states and exogenous information

LSPI requires generating a set of M sampled states $D = \{s_1, \ldots, s_M\} \subseteq \mathcal{S}$. A sampled state is obtained by first randomly generating the total number of patients in the system. For each patient in the system, a random patient care pathway is selected from a dataset containing realised care pathways. A patient care pathway contains the ordered appointments/queues of a patient. We then randomly draw the patient's current queue from their pathway and the current waiting time. The sampled state is the number of patients in each queue $j \in \mathcal{J}$ waiting for $w \in \mathcal{W}_j$ time periods.

In addition, to determine the next state s'_m, we need to sample exogenous information x_m, which contains the number of newly arriving patients for each queue. Similar to the sampled state, we randomly generate a number of new patients and for each randomly select a care pathway from the set of realised care pathways. For these patients, we determine the first queue in their pathway and set the waiting time to 0.

4.2.2 Action selection

Using the policy function to choose an action requires solving:

$$\pi\left(s|\theta\right) = \arg\max_{a \in \mathcal{A}(s)} \left(C(s,a) + \gamma\theta^\top \phi(\overline{T}(s,a))\right),$$

where $\overline{T}(s,a)$ is an approximation of the next state. When an MDP can be solved exactly, given optimal value function V^*, the optimal policy can be described as:

$$\begin{aligned}\pi(s) &= \arg\max_{a \in \mathcal{A}(s)} \left(C(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a)V^*(s')\right) \\ &= \arg\max_{a \in \mathcal{A}(s)} \left(C(s,a) + \gamma \mathbb{E}[V^*(s')|s,a]\right).\end{aligned}$$

Now, rather than an optimal value function, we have an approximate value function $\overline{V}$. Since we are using a linear parametrisation and linear basis functions, this function is linear in the state vector. Therefore, we can rewrite $\mathbb{E}[\overline{V}(s')|s,a]$ to $\overline{V}(\mathbb{E}[s'|s,a])$. This gives:

$$\pi\left(s|\theta\right) = \arg\max_{a \in \mathcal{A}(s)} \left(C(s,a) + \gamma\theta^\top \phi(\mathbb{E}[s'|s,a])\right).$$

For this problem, the expected next state can be calculated using the transition probabilities $q_{i,j}$ from queue $i \in \mathcal{J}$ to queue $j \in \mathcal{J}$ and expected number of newly arriving patients λ_j to queue $j \in \mathcal{J}$. Given an action a, the expected number of patients

transitioning from queue $i \in \mathcal{J}$ to queue $j \in \mathcal{J}$ is given by $\sum_{w \in \mathcal{W}_j} q_{i,j} \cdot a_{i,w}$. The rest of the state transitions is determined by deterministic events as patients who are not treated wait an extra time period.

Using the above, we can formulate the following integer linear program to determine the next action resulting in the expected next state $\overline{s}$, which we refer to as the *policy-LP*:

$$\max_{a} \quad C(s,a) + \gamma \sum_{f=1}^{F} \theta_f \cdot \phi_f(\overline{s}) \tag{5a}$$

$$\text{s.t.} \quad \overline{s}_{j,0} = \lambda_j + \sum_{i \in \mathcal{J}} \sum_{w \in \mathcal{W}_i} q_{i,j} \cdot a_{i,w}, \quad \forall j \in \mathcal{J}, \tag{5b}$$

$$\overline{s}_{j,w} = s_{j,w-1} - a_{j,w-1}, \quad \forall j \in \mathcal{J}, 1 \leq w \leq W_j - 1, \tag{5c}$$

$$\overline{s}_{j,W_j} = \sum_{w=W_j-1}^{W_j} s_{j,w} - a_{j,w}, \quad \forall j \in \mathcal{J}, \tag{5d}$$

$$a_{j,w} \leq s_{j,w}, \quad \forall j \in \mathcal{J}, w \in \mathcal{W}_j, \tag{5e}$$

$$\sum_{j \in \mathcal{J}} \sum_{w \in \mathcal{W}_j} \zeta_{j,r} a_{j,w} \leq \eta_r, \quad \forall r \in \mathcal{R}, \tag{5f}$$

$$a_{j,w} = 0, \quad \forall j \in \mathcal{J}, w \in \mathcal{W}_j w < m_j, \tag{5g}$$

$$a_{j,w} \geq 0, \quad \forall j \in \mathcal{J}, w \in \mathcal{W}_j, \tag{5h}$$

$$\overline{s}_{j,w} \geq 0, \quad \forall j \in \mathcal{J}, w \in \mathcal{W}_j. \tag{5i}$$

The contribution function $C(s, a)$ is as defined in Eq. (3). Constraints (5b)–(5d) determine the expected next state following Eq. (2). Constraints (5b) define for each queue $j \in \mathcal{J}$ the expected number of patients to transition to queue j given actions $a_{i,w}$ for all $i \in \mathcal{J}$ and $w \in \mathcal{W}_i$ and the expected number of new patients λ_j arriving at queue j. Constraints (5c) ensure for each queue $j \in \mathcal{J}$ and waiting time $w \in \mathcal{W}_j$ with $w < W_j - 1$ that patients who are not treated stay in the same queue and their waiting time is incremented by 1. Constraints (5d) ensure for each queue $j \in \mathcal{J}$ that the waiting time is held constant when the upper bound is reached. Constraints (5e)–(5f) define all feasible actions following Eq. (1). Constraints (5e) ensure that not more patients are treated than waiting for $w \in \mathcal{W}_j$ in queue $j \in \mathcal{J}$. Constraints (5f) ensure for each resource $r \in \mathcal{R}$ that the resource capacity η_r is not exceeded. Constraints (5g) make sure that patients from queue $j \in \mathcal{J}$ cannot be treated before having met their minimal access time m_j. Note that for λ_j and η_r, we take the expected value of

$\lambda_{j,t}$ and $\eta_{r,t}$ over $t \in \mathcal{T}$, respectively. Finally, Constraints (5h)–(5i) ensure that the action and state variables represent non-negative numbers of patients.

Even though it is only possible to treat an integer number of patients, we have relaxed the restriction on action variable $a_{j,w}$. Since this program must be solved for every state in dataset D, in every iteration of the algorithm, we have relaxed this integer constraint to a non-negativity constraint in order to reduce the solving time. When we use the resulting actions, we check whether rounding to the nearest integer violates the capacity. If not, we use the rounded actions. If so, we round down to the nearest integer. Rounding down results in loss of optimality, as some available capacity goes unused. When the policy is actually used for planning, the integer constraint can be used, as in that case the policy-LP is solved once per planning period.

4.2.3 Basis functions

Well-chosen basis functions are essential for the convergence and performance of LSPI. In this section, various basis functions are proposed. We also introduce a linear regression method to determine which basis function best approximates the exact value function V^*.

Basis functions may be non-linear (for example, Gaussian or quadratic functions of features), but we focus on linear functions to ensure that an action selection can be done by solving the policy-LP, and such that $\mathbb{E}[\phi(s)] = \phi(\mathbb{E}[s])$. When designing basis functions, it is important to take the dimension F of a basis function into account. A larger dimension F will increase the computation time, as the algorithm requires computing the inverse of an $F \times F$ matrix A. A high-dimensional parametrisation also increases the risk of overfitting. Conversely, decreasing F may result in a worse value function approximation if the features do not accurately represent the state space.

Table 2 shows the four basis functions we selected, and their dimension. Basis functions 1 and 4 are similar to those used in Hulshof et al. (2016). It is also possible to use a combination of multiple basis functions, for example, the cost of each queue *and* the total number of patients in each queue. Since the number of queues $|\mathcal{J}|$ is limited, the dimension of basis functions 2–4 is manageable. As the dimension of basis function 1 is the sum of the cardinalities of sets $\mathcal{W}_j$ of patient waiting times, it may become intractable. This would force us to upper-bound the waiting times, whereas this is not strictly necessary for the other basis functions. However, an upper bound on waiting times is necessary in any case for the policy-LP, because it becomes slower to solve as the upper bound increases.

A term of 1 should be added to each basis function, increasing the dimension by one. If we do not do this, the all-zero state will always have value $V(0) = \theta^\top \phi(0) = 0$, which will not necessarily correspond with the exact value for that state. This term will improve the ability to approximate the exact value function. However, it does not change the resulting policy, as it only adds a constant term to the approximate value function and therefore does not affect which action results in the maximum value.

If the exact value function V^* is known, we can use it to determine which basis functions should be used for LSPI. Given a basis function ϕ, linear regression can be used to calculate the θ minimizing the mean squared error (MSE) between the exact

value function V^* and approximate value function $\overline{V} = \theta^T \phi$ for a training set of n states $\{s_1, \ldots, s_n\} \subset \mathcal{S}$ Powell (2019):

$$\theta^* = \arg\min_{\theta \in \mathbb{R}^F} \left(\sum_{i=1}^{n} \left(V^*(s_i) - \theta^\top \phi(s_i) \right)^2 \right). \tag{6}$$

After using a training set of n states to find θ^* for a given basis function, the performance of that basis function can be measured by calculating the MSE on a test set of m states:

$$\text{MSE}(\phi, \{s_1, \ldots, s_m\}) = \sum_{i=1}^{m} \left(V^*(s_i) - \theta^{*\top} \phi(s_i) \right)^2.$$

The training and test sets should be disjoint, as the test set measures how well the found parameterisation generalises to unseen samples. Furthermore, we use k-fold cross-validation to obtain an accurate estimation of the true MSE for the whole dataset (Koutroumbas and Theodoridis 2008). This process is repeated for different basis functions. We then use the basis function resulting in the lowest MSE since it provides the closest approximation to the exact value function.

4.3 Integer linear programming

This section presents the integer linear program that we use to solve the deterministic, finite-time-horizon variant of the MDP presented in Sect. 3. We use the same sets, parameters, and variables; see Table 1. We modify the model because an ILP is deterministic and cannot handle an infinite time horizon. To remove all stochasticity from the problem, we let $q_{i,j}$ represent transition fractions instead of transition probabilities. In addition, we take for $\lambda_{j,t}$ the expected number of patients arriving in queue $j \in \mathcal{J}$ at time $t \in \mathcal{T}$. Instead of $\mathcal{T} = \mathbb{Z}^+$, we use a finite time horizon $\mathcal{T} = \{0, 1, \ldots, T\}$ for some $T \in \mathbb{Z}^+$.

Table 2 Selected basis functions for least-squares policy iteration

	Feature	Basis function	F
1	Number of patients in queue $j \in \mathcal{J}$ waiting for $w \in \mathcal{W}_j$ time periods	$s_{j,w}$, $\forall j \in \mathcal{J}, w \in \mathcal{W}_j$	$\sum_{j \in \mathcal{J}} \lvert\mathcal{W}_j\rvert$
2	Number of patients in queue $j \in \mathcal{J}$ who are waiting shorter, equal to, and longer than the access time target for treatment	$\sum_{w \leq u_j - 1} s_{j,w}$, s_{j,u_j}, $\sum_{w > u_j} s_{j,w}$, $\forall j \in \mathcal{J}$	$3 \cdot \lvert\mathcal{J}\rvert$
3	Cost of patients in queue $j \in \mathcal{J}$	$\sum_{w > d_j} c_{j,w} \cdot s_{j,w}$, $\forall j \in \mathcal{J}$	$\lvert\mathcal{J}\rvert$
4	Total number of patients in queue $j \in \mathcal{J}$	$\sum_{w \in \mathcal{W}_j} s_{j,w}$, $\forall j \in \mathcal{J}$	$\lvert\mathcal{J}\rvert$

Note that the ILP is similar to the policy-LP (5a), except that we take multiple time periods into account. To account for less precision in the later time periods, we use discount factor γ.

$$\max_{a \in \mathcal{A}^T} \sum_{t \in \mathcal{T}} \gamma^t C(s_t, a_t) \tag{7a}$$

$$s_{j,w,0} = s_{j,w}^{current}, \quad \forall j \in \mathcal{J}, w \in \mathcal{W}_j, \tag{7b}$$

$$0 s_{j,0,t} = \lambda_{j,t} + \sum_{i \in \mathcal{J}} \sum_{w \in \mathcal{W}_j} q_{i,j} a_{i,w,t-1}, \quad \forall j \in \mathcal{J}, t \in \mathcal{T}, t > 0, \tag{7c}$$

$$s_{j,w,t} = s_{j,w-1,t-1} - a_{j,w-1,t-1}, \quad \forall j \in \mathcal{J}, 1 \leq w \leq W_j, t \in \mathcal{T}, t > 0, \tag{7d}$$

$$s_{j,W_j,t} = \sum_{w=W_j-1}^{W_j} s_{j,w,t} - a_{j,w,t}, \quad \forall j \in \mathcal{J}, t \in \mathcal{T}, \tag{7e}$$

$$a_{j,w,t} \leq s_{j,w,t}, \quad \forall j \in \mathcal{J}, w \in \mathcal{W}_j, t \in \mathcal{T}, \tag{7f}$$

$$\sum_{j \in \mathcal{J}} \sum_{w \in \mathcal{W}_j} \zeta_{j,r} a_{j,w,t} \leq \eta_{r,t}, \quad \forall r \in \mathcal{R}, t \in \mathcal{T}, \tag{7g}$$

$$a_{j,w} = 0, \quad \forall j \in \mathcal{J}, w < m_j, \tag{7h}$$

$$a_{j,w,t} \in \mathbb{Z}^+, \quad \forall j \in \mathcal{J}, w \in \mathcal{W}_j, t \in \mathcal{T}. \tag{7i}$$

Constraints (7b) fix for each queue $j \in \mathcal{J}$ and waiting time $w \in \mathcal{W}_j$ the initial state $s_{j,w,0}$ to the current state $s_{j,w}^{current}$ for which the ILP is solved. Constraints (7c) define the expected number of new patients in queue $j \in \mathcal{J}$ at time $t \in \mathcal{T}$ by summing the expected number of new patients $\lambda_{j,t}$ and the expected number of patient transitioning to queue j from other queues. Constraints (7d) ensure for queue $j \in \mathcal{J}$ that patients who are not treated stay in the same queue and their waiting time is incremented by 1. Constraints (7e) ensure for each queue $j \in \mathcal{J}$ that the waiting time is held constant when the upper bound W_j is reached. Constraints (7f) ensure that not more patients are treated from queue $j \in \mathcal{J}$ with waiting time $w \in \mathcal{W}_j$ than there are in the queue. Constraints (7g) ensure for each resource $r \in \mathcal{R}$ that the resource capacity η_r is not exceeded. Constraints (7h) make sure that only patients from queue $j \in \mathcal{J}$ that have met their minimal access time m_j are treated. Finally, Constraints (7i) ensure that the action variables represent non-negative integer numbers of patients. To make the problem faster to solve, integer constraints (7i) could be relaxed to $a_{j,w,t} \geq 0$. In that case, we check whether the actions can be rounded to the next integer without violating capacity. If this is not possible, the actions in the optimal solution must be

rounded down to the nearest integer to ensure feasibility, which could result in a loss of optimality.

The solution to the ILP is a vector $a = (a_0, \ldots, a_T)$ of actions to perform over a time horizon of length *T*. However, actually performing these actions may not lead to good results in the real world for two reasons. First, since the solution is fixed and transition fractions are used rather than probabilities, the solution is not optimal for real-life realizations. Second, the finite time horizon distorts the solution towards the end, as future consequences of those actions are not taken into account. It is more realistic that a capacity planning department making use of this ILP would solve it every time a planning decision is required, with s_0 as the (predicted) state for which an allocation must be determined, and only implement the first action a_0. We refer to this as a rolling time horizon. *T* should still be set to a longer period of time, such as one year, so that future consequences of the first action are taken into account.

4.4 Heuristic dynamic decision rules

In this section, we formulate four decision rules that can provide a feasible solution to the timeslot allocation problem at a certain time $t \in \mathcal{T}$. The following decision rules are considered. Note that the index *t* is omitted as we consider the state and resource capacity at a certain fixed time $t \in \mathcal{T}$.

- *Highest contribution*: Treat the patient that would result in the highest increase in contribution if treated now, taking only patients into account that have waited the minimal access time. The contribution is expressed as the sum of the reward and the cost of the patient; by treating that patient, we gain the associated reward and avoid the associated cost. Find the

$$(j^*, w^*) = \arg\max \{c_{j,w} + r_j : \ j \in \mathcal{J}, w \in \mathcal{W}_j, w \geq m_j, s_{j,w} \geq 1\}.$$

  Increment sub-action a_{j^*,w^*} with one and subtract one from sub-state s_{j^*,w^*} and the corresponding remaining resource capacity. Continue until all resource capacity has been used or no eligible patients remain. The resulting action $a = (a_{j,w})_{j \in \mathcal{J}, w \in \mathcal{W}_j}$ denotes the number of patients treated from each sub-state $s_{j,w}$ according to the Highest Contribution rule.
- *Highest cost*: Treat the patient that would add the highest cost when not treated, taking only patients into account that have waited the minimal access time:

$$(j^*, w^*) = \arg\max \{c_{j,w} : \ j \in \mathcal{J}, w \in \mathcal{W}_j, w \geq m_j, s_{j,w} \geq 1\}.$$

  Increment sub-action a_{j^*,w^*} with one and subtract one from sub-state s_{j^*,w^*} and from the corresponding remaining resource capacity. Continue until all resource capacity has been used or no eligible patients remain. The resulting action $a = (a_{j,w})_{j \in \mathcal{J}, w \in \mathcal{W}_j}$ denotes the number of patients treated from each sub-state $s_{j,w}$ according to the Highest Cost rule.

- *Longest queue*: Treat the patient with the longest waiting time from the longest queue, taking only patients into account that have waited the minimal access time, i.e.,

$$j^* = \arg\max \left\{ \sum_{w \in \mathcal{W}_j, w \geq m_j} s_{j,w} : \ j \in \mathcal{J} \right\},$$

$$w^* = \max \left\{ w \in \mathcal{W}_j : s_{j^*,w} \geq 1 \right\}.$$

  Increment sub-action a_{j^*,w^*} with one and subtract one from sub-state s_{j^*,w^*} and from the corresponding remaining resource capacity. Continue until all resource capacity has been used or no eligible patients remain. The resulting action $a = (a_{j,w})_{j \in \mathcal{J}, w \in \mathcal{W}_j}$ denotes the number of patients treated from each sub-state $s_{j,w}$ according to the Longest Queue rule.

Split cost: For this decision rule, we assume that each queue $j \in \mathcal{J}$ only requires one resource $r_j \in \mathcal{R}$. To determine the number of patients a_j to treat from queue $j \in \mathcal{J}$, we distribute the available resource capacity of resource $r_j \in \mathcal{R}$ over all queues $i \in \mathcal{J}_{r_j}$ that require resource r_j, proportional to the total cost of the queue.

$$a_j = \min \left(\sum_{w \geq m_j} s_{j,w}, \left\lfloor \eta_{r_{j,t}} \cdot \frac{\sum_{w \geq m_j} c_{j,w} \cdot s_{j,w}}{\sum_{i \in \mathcal{J}_{r_j}} \sum_{w \geq m_j} c_{i,w} \cdot s_{i,w}} \right\rfloor \right), \qquad \forall j \in \mathcal{J}.$$

  Note that we cannot treat more than $\sum_{w \in \mathcal{W}_j, w \geq m_j} s_{j,w}$ patients for queue $j \in \mathcal{J}$. Within each queue, the patients with the highest waiting time are treated.

These rules are simple and easy to implement and understand. For hospitals, the use of decision rules is beneficial as they do not require complex optimization software and a solution can be found quickly. Also, it is much easier to explain to hospital staff why certain decisions should be made. The rule itself provides an explanation, for example, "we should always treat our most urgent patients first". This is in contrast with the other approaches, which are essentially black-boxes, as mathematical knowledge is required to explain why the solution provided is optimal. However, we expect that this increase in simplicity and explainability comes with a loss of performance.

4.5 Static allocation

To compare the new solution methods with the current practice, we consider the method currently used by SMK for timeslot allocation within OD sessions. The hospital generally does not make adaptations to the number of first appointments (FA), follow-up appointments (FU), and discharge appointments (DA) that are planned within an OD session. Rather, a static allocation is used with a fixed number of appointments for each appointment type, as this can be done quickly and easily. Since

manually adapting a surgeon's timeslots is a lot of work, it only happens temporarily when an intervention is necessary to prevent overly long or short waiting lists.

For most surgeons, there are three different static allocations used to plan an OD session, depending on whether the surgeon is aided during that session by zero, one, or two assistants. Schedules for sessions with more assistants allow for more OD appointment slots. However, the proportion of FA, FU, and DA appointments within one session remains roughly the same. The hospital determines these proportions by calculating the ratios between the appointment types for previous patients of the surgeon, and total OD time. Under the assumption that future patients and care pathways are similar to those in the past, this should ensure that waiting lists remain relatively balanced. We evaluate this method to compare between the new solution methods discussed in this paper and the current allocation method of SMK.

4.6 Hybrid method

Finally, we investigate a hybrid method that combines the static allocation method that plans as far ahead as hospitals want, with a dynamic method that plans a few weeks in advance. This allows us to make some appointments far enough in advance, while exploiting the improved knowledge we have of the predicted state when planning less far ahead. In this section, we describe a hybrid method that uses the rolling-horizon LP for the dynamic decisions, but we can also combine the static allocation with the other solution methods.

The static allocation method is used to determine the allocation of a fraction $0 \leq \alpha \leq 1$ of the timeslots, the *static* allocation fraction. Then, we use the rolling-horizon LP p time periods prior to the week to be scheduled, to determine the allocation of the remaining $(1-\alpha)$ of the timeslots, making up the *dynamic* schedule fraction.

Constraints (8)–(12) should be added to the rolling-horizon LP in Sect. 4.3 to ensure that the correct action is taken. Let $\overline{a}_{j,t}$ be the number of patients to treat from queue $j \in \mathcal{J}$, as determined by the static allocation. Let K be some large number representing an upper bound on the queue length over all queues $j \in \mathcal{J}$. Recall that $s_{j,w,t}$ denotes the sub-state size and $a_{j,w,t}$ the number of patients to treat from that sub-state. Constraints (8) and (10) determine the minimum of the number of patients available for treatment ($\sum_{m_j \leq w \leq W_j} s_{j,w,t}$ in queue $j \in \mathcal{J}$ at time period $t \in \mathcal{T}$) and the number of patients to treat according to the static allocation fraction ($\overline{a}_{j,t} \cdot \alpha$). For this, we introduce binary variables $y_{j,t}$ for each queue $j \in \mathcal{J}$ and time period $t \in \mathcal{T}$, which are 1 when the minimum is equal to $\sum_{m_j \leq w \leq W_j} s_{j,w,t}$ and 0 when the minimum is equal to $\overline{a}_{j,t} \cdot \alpha$. Then, Constraints (9) and (11) ensure that the action $\sum_{w \in \mathcal{W}_j} a_{j,w,t}$ chosen for queue $j \in \mathcal{J}$ at time period $t \in \mathcal{T}$, is greater than or equal to this minimum. This is necessary because simply forcing the action to be larger than or equal to the static fraction of the allocation can lead to infeasibility, if the size of the queue is smaller in that time period.

$$\overline{a}_{j,t} \cdot \alpha - \sum_{w=m_j}^{W_j} s_{j,w,t} \leq K \cdot y_{j,t}, \qquad \forall j \in \mathcal{J}, t \in \mathcal{T}, \tag{8}$$

$$\sum_{w \in \mathcal{W}_j} a_{j,w,t} \geq \overline{a}_{j,t} \cdot \alpha - K \cdot y_{j,t} \qquad \forall j \in \mathcal{J}, t \in \mathcal{T}, \tag{9}$$

$$\sum_{w=m_j}^{W_j} s_{j,w,t} - \overline{a}_{j,t} \cdot \alpha \leq K \cdot (1 - y_{j,t}), \qquad \forall j \in \mathcal{J}, t \in \mathcal{T}, \tag{10}$$

$$\sum_{w \in \mathcal{W}_j} a_{j,w,t} \geq \sum_{w=m_j}^{W_j} s_{j,w,t} - K \cdot (1 - y_{j,t}), \qquad \forall j \in \mathcal{J}, t \in \mathcal{T}, \tag{11}$$

$$y_{j,t} \in \{0, 1\}, \qquad \forall j \in \mathcal{J}, t \in \mathcal{T}. \tag{12}$$

Note that when $\alpha = 0$ the hybrid method is equal to the (I)LP and when $\alpha = 1$ it is equal to the static method.

5 Test instances

To test how well the solution methods from Sect. 4 perform on the model constructed in Sect. 3, we create three test instances. Section 5.1 discusses a small test instance and Sect. 5.2 a large test instance. These artificial instances are used to tune the parameters of the solution methods. Section 5.3 discusses the full-scale instance based on actual SMK data.

The small test instance is required because it can be solved using exact value iteration (EVI). This instance is used to assess how well a policy obtained through LSPI approximates the optimal policy found via EVI. To enable an exact solution, the state space must remain sufficiently small. However, this strict size constraint makes the instance less representative of the real-life problem, which motivates the definition of a second, larger test instance. This larger dataset is used to tune the LSPI parameters. For this purpose, it should resemble the real-life setting while remaining small enough to allow for efficient solution and experimentation. For both test instances and the complete SMK instance, each time period represents a planning period of 2 weeks.

5.1 Small test instance

The following test instance with a bounded state space enables comparison of the policy obtained through LSPI with the optimal policy found through EVI. There are three queues $\mathcal{J} = \{FU_1, OR_0, OR_1\}$, of which the FU queue has an access time target u_{FU_1} of 1 and the OR queues have an access time target u_{OR_0} and u_{OR_1} of 0 and 1, respectively. The resource capacities of the resources $\mathcal{R} = \{OD, OR\}$

are set to $\eta_{OD,t} = 4$ and $\eta_{OR,t} = 2$ for all time periods $t \in \mathcal{T}$. The FU queue requires $\zeta_{FU_1,OD} = 1$ unit of OD time per patient, and the OR queues require $\zeta_{OR_0,OR} = \zeta_{OR_1,OR} = 1$ unit of OR time per patient. The other resource requirements $\zeta_{FU_1,OR}$, $\zeta_{OR_0,OD}$, and $\zeta_{OR_1,OD}$ are 0. The upper bound on the waiting time, W_j, is set to 2 for each queue $j \in \mathcal{J}$. To guarantee a bounded state space when applying EVI, the queue lengths are bounded from above by 7 and 2 for the FU queue and OR queues, respectively. The transition probabilities between queues are shown in Fig. 1. Three new patients enter the FU queue in every time period, so $\lambda_{FU_1,t} = 3$ for all $t \in \mathcal{T}$, and $\lambda_{OR_0,t}$ and $\lambda_{OR_1,t}$ are 0 for all $t \in \mathcal{T}$. The cost for keeping a patient waiting in queue $j \in \mathcal{J}$ for $w \in \mathcal{W}_j$ time periods is set to $c_{j,w} = \omega_j \cdot \frac{w}{u_j+1}$ where $\omega_{FU_1} = 1$ and $\omega_{OR_0} = \omega_{OR_1} = 4$. Adding 1 to u_j in the denominator is necessary as one of the access time targets is 0. The rewards for treating patients are $r_{FU_1} = 1$ and $r_{OR_0} = r_{OR_1} = 4$.

5.2 Large test instance

The following test instance, a scaled-down but representative version of the full problem, enables efficient testing of different parameter settings and algorithm variations for the solution methods. There are five queues: $\mathcal{J} = \{FA_2, FU_4, OR_2, OR_4, DA_3\}$, for which the access time targets are given by $u_{FA_2} = 2$, $u_{FU_4} = 4$, $u_{OR_2} = 2$, $u_{OR_4} = 4$, and $u_{DA_3} = 3$. As for the small test instance, there are two resources given by $\mathcal{R} = \{OD, OR\}$. The FA, FU, and DA queues require one unit of OD time, i.e., $\zeta_{FA_2,OD} = \zeta_{FU_4,OD} = \zeta_{DA_3,OD} = 1$ and the OR queues require one unit of OR time per patient, i.e., $\zeta_{OR_2,OR} = \zeta_{OR_4,OR} = 1$. The other resource requirements $\zeta_{FA_2,OR}$, $\zeta_{FU_4,OR}$, $\zeta_{OR_2,OD}$, $\zeta_{OR_4,OD}$, $\zeta_{DA_3,OR}$ are 0. Resource capacities are set to $\eta_{OD,t} = 16$ and $\eta_{OR,t} = 2$ for all time periods $t \in \mathcal{T}$. Transition probabilities are shown in Table 3. The rows show the probability that a patient will transfer from the appointment type in that row to the appointment type indicated in each column. The 'Exit' column is the probability that the patient will leave the system. 8 new patients enter the FA queue in every time period, so $\lambda_{FA_2,t} = 8$ for all $t \in \mathcal{T}$ and $\lambda_{FU_4,t}$, $\lambda_{OR_2,t}$, $\lambda_{OR_4,t}$, and $\lambda_{DA_3,t}$ are 0 for all $t \in \mathcal{T}$.

The upper bound on access times is $W_j = 3 \cdot u_j$ for all $j \in \mathcal{J}$. This allows higher waiting times for queues with higher access time targets, as patients generally wait longer in those queues.

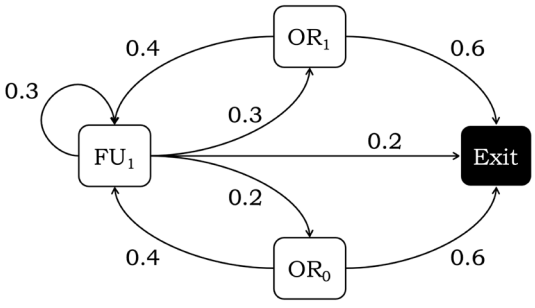

Fig. 1 Queues and transition probabilities for the small test instance

The cost for keeping a patient waiting in queue $j \in \mathcal{J}$ for $w \in \mathcal{W}_j$ time periods is set to $c_{j,w} = \omega_j \cdot \frac{w}{u_j}$ where $\omega_{FA_2} = 2, \omega_{FU_4} = 1, \omega_{OR_2} = \omega_{OR_4} = 4$ and $\omega_{DA_3} = 1$. The rewards for treating patients are $r_{FA_2} = r_{FU_4} = 2, r_{OR_2} = r_{OR_4} = 10$, and $r_{DA_3} = 1$.

5.3 Case study instances

The real-life instance is based on data collected at the SMK and represents the allocation problem for one surgeon. This section describes the queues, urgencies, and parameters that specify the real-life instance.

There are nine queues: $\mathcal{J} = \{FA_2, FU_3, FU_6, FU_{12}, OR_1, OR_2, OR_4, OR_6, DA_3\}$, for which the access time targets are given by $u_{FA_2} = 2, u_{FU_3} = 3, u_{FU_6} = 6$, $u_{FU_{12}} = 12$, $u_{OR_1} = 1$, $u_{OR_2} = 2$, $u_{OR_4} = 4$, $u_{OR_6} = 6$, and $u_{DA_3} = 3$. Minimal access times are $m_{FA_2} = 0$, $m_{FU_3} = 2$, $m_{FU_6} = 2$, $m_{FU_{12}} = 2$, $m_{OR_1} = 0$, $m_{OR_2} = 1$, $m_{OR_4} = 1$, $m_{OR_6} = 1$, and $m_{DA_3} = 2$. Again, there are two resources given by set $\mathcal{R} = \{OD, OR\}$.

The SMK yearly budget for the considered surgeon defines that a capacity of 3155 outpatient time units, 300 surgeries per year, and 1196 external arrivals per year result in balanced OD and OR waiting lists. Due to e.g. holidays, the capacity is not divided equally over the year. This is reflected in the capacity per time period $\eta_{r,t}$. The number of external arrivals is drawn from a normal distribution with $\mu = 46$ and $\sigma = 40$. To investigate sensitivity, an unbalanced setting is also simulated in which OR capacity is reduced to 220 surgeries per year.

First appointments require two OD time units, while follow-up and discharge appointments require only one OD time unit. Surgeries require one OR time unit. This gives $\zeta_{FA_2,OD} = 2$, and $\zeta_{FU_3,OD} = \zeta_{FU_6,OD} = \zeta_{FU_{12},OD} = \zeta_{DA_3,OD} = \zeta_{OR_1,OR} = \zeta_{OR_2,OR} = \zeta_{OR_4,OR} = \zeta_{OR_6,OR} = 1$. The other resource requirements are 0. As for the large test instance, the upper bounds on the waiting times are set to $W_j = 3 \cdot u_j$ for all $j \in \mathcal{J}$.

Patient pathways extracted from the hospital information system are used to generate vectors of appointment types, for example $(FA_2, FU_3, FU_6, OR_2, DA_3)$. These vectors represent the chronological sequence of appointments for each patient. Transition probabilities between appointment types are estimated from these pathways by counting the observed transitions between consecutive appointments. For each appointment type, the transition probability to a subsequent appointment type is calculated as the relative frequency of that transition among all observed transitions originating from that appointment type. Table 4 summarizes the transition probabilities. Each row provides the probability of a patient transferring to the appointment

Table 3 Transition probabilities $q_{i,j}$ for the large test instance

	FA_2	FU_4	OR_2	OR_4	DA_3	Exit
FA_2	0	0.5	0.01	0.1	0	0.39
FU_4	0	0.4	0.02	0.15	0	0.43
OR_2	0	0.2	0	0	0.75	0.05
OR_4	0	0.25	0	0	0.7	0.05
DA_3	0	0.6	0	0	0	0.4

type indicated in each column. The 'Start' row shows the probability that an appointment type is the patient's first interaction with the considered surgeon. The 'Exit' column shows the probability that the treatment is finished.

The costs and rewards in the contribution function should reflect SMK's priorities and the relative importance of different appointment types. SMK has indicated that it is most important to meet the access time targets for the OR queue and to use all available OR capacity, as OR time is more expensive to waste than OD time. The hospital also expressed that meeting access time targets for follow-up and discharge appointments is more important than for first appointments. Next, for elective appointments and surgeries, the target access time is less strict than for acute appointments. We therefore only incur costs after $d_{FU_3} = 4$, $d_{FU_6} = 7$, $d_{FU_{12}} = 14$, $d_{OR_4} = 5$, $d_{OR_6} = 8$, and $d_{DA_3} = 4$. To determine appropriate costs and rewards, we repeatedly simulated the system using the rolling-horizon LP as the solution method, and adjusted costs and rewards until the model showed the desired behavior (as described above). For each queue $j \in \mathcal{J}$ and waiting time $w \in \mathcal{W}_j$ we used the cost function $c_{j,w} = \omega_j \cdot \frac{w}{u_j}$ for $w \geq d_j$ and 0 elsewhere. From the iterative simulations, weights were obtained as $\omega_{FA_2} = 0.5$, $\omega_{FU_3} = \omega_{FU_6} = \omega_{FU_{12}} = 3$, $\omega_{OR_1} = \omega_{OR_2} = \omega_{OR_4} = \omega_{OR_6} = 10$, and $\omega_{DA_3} = 3$. Rewards were obtained as $r_{FA_2} = 5$, $r_{FU_3} = r_{FU_6} = r_{FU_{12}} = 3$, $r_{OR_1} = r_{OR_2} = r_{OR_4} = r_{OR_6} = 50$, and $r_{DA} = 3$.

6 Computational results

This section shows the results of applying the different solution methods to the three considered instances. Section 6.1, describes the use of simulation to evaluate the expected performance of the various solution methods in practice. Section 6.2 discusses the use of the small test instance to determine how well the approximate value function found using LSPI approximates the true value function found using EVI, and which basis functions lead to the best approximation. Section 6.3, describes the results for different LSPI parameters, the effects of changing γ on the rolling-horizon LP, and the comparison of the decision rules introduced in Sect. 4.4, using the large

Table 4 Transition probabilities $q_{i,j}$ for the case study instance

	FA_2	FU_3	FU_6	FU_{12}	OR_1	OR_2	OR_4	OR_6	DA_3	Exit
Start	0.7116	0	0.2138	0	0.0185	0.0026	0.0049	0.0264	0.0220	0
FA_2	0.0037	0.2400	0.1298	0.1010	0.0012	0.0049	0.0055	0.0753	0.0147	0.4238
FU_3	0.0028	0.1951	0.1127	0.0919	0.0038	0.0104	0.0189	0.0994	0.0170	0.4479
FU_6	0.0085	0.1575	0.0840	0.0953	0.0028	0.0028	0.0038	0.0660	0.0151	0.5642
FU_{12}	0	0.1535	0.0930	0.0605	0.0023	0.0023	0	0.0628	0.0070	0.6186
OR_1	0	0.2	0.0333	0.0167	0.0667	0	0	0	0.3833	0.3
OR_2	0	0.1	0	0	0.0333	0	0.0333	0.0333	0.6667	0.1333
OR_4	0	0.1522	0.0435	0.0435	0.0217	0	0	0	0.5870	0.1522
OR_6	0	0.1421	0.0299	0.0175	0.0050	0	0	0.0099	0.7182	0.0773
DA_3	0.0021	0.3080	0.2089	0.0654	0	0.0021	0.0021	0.0232	0.0105	0.3776

test instance. In these sections, perfect information about the state is assumed when selecting an action, i.e., no planning ahead is performed. In Sect. 6.4, the solution methods are applied to the case study instance when planning ahead is required.

6.1 Evaluation method

To compare the performance of the different solution methods, we simulate the application of the solution methods for a number of trials over a number of subsequent time periods. To generate an initial state with patients in queues that are at a realistic point in their treatment, we require patient pathways. For the SMK instance, we randomly draw patient pathways out of a historical dataset of realized patient pathways. Since there is no historical data for the small and large test instances, we generate patient pathways by starting with a first appointment (FA) and then randomly drawing the next stage or exit based on the transition probabilities.

We initialize the simulation by drawing pathways for the selected number of initial patients and sampling their current stage and waiting time. Patients are modeled as tuples $\langle v, i, w \rangle$ where v is the patient pathway, i is a counter indicating the stage of treatment the patient is in (starting at 1), and w is the current waiting time. For example, patient $\langle (FA_4, FU_6, FU_3, OR_4, DA_3), 4, 2 \rangle$ indicates a patient waiting for surgery with an access time target of 4 who has been waiting for two time periods after completing one first appointment and two follow-up appointments.

The current waiting time for the initial state is sampled from the typical distribution of waiting times for each appointment type at any moment in time. We have fitted four probability distributions (beta, gamma, exponential, and truncated normal) to the SMK data on follow-up appointments, and the exponential distribution resulted in the best fit. Based on this, we estimate that the waiting times of patients are exponentially distributed with scale parameter $\beta = u_j$, where u_j is the access time target of queue $j \in \mathcal{J}$.

An exception is made for first appointments. The waiting list for first appointments can contain a large number of patients relative to the weekly arrival rate. When sampling the current waiting time of these patients from an exponential distribution, a disproportionately large number of patients may be assigned a very short waiting time. To avoid this unrealistic concentration of short waiting times, we instead sample waiting times for first appointments from a uniform distribution. The current waiting time for individual patients is drawn from the distributions and rounded to the nearest integer less than or equal to W_j.

The simulated patient pathways are unknown to the solution methods; they are used solely to determine the next state after an action. When the patient is treated, i is incremented by 1 and w is set to 0. If a patient has to wait for another time period, w is incremented by 1 when w was less than W_j or remains W_j when w was already equal to W_j. When $i > |v|$, the patient has completed their treatment and is removed from the system. After each time period, a fixed number of new patients arrive, for which we sample a patient pathway. These patients are initialized at stage 1 with a waiting time of 0, i.e., $i = 1$ and $w = 0$.

We measure the performance of the solution methods by whether patients are seen within the access times targets and capacity is used efficiently, see Sect. 6.4.2. For

efficiency, we estimate this performance by the average contribution per time period over a number of trials. After applying an action to the current state, we add the corresponding contribution to the total contribution of that trial. Within each trial, the same settings are used for each solution method, meaning that the initial patients and newly arriving patients in each time period are the same. Note that the average contribution performance measure is not related to the "average reward optimality criterion" for undiscounted infinite horizon MDPs.

6.2 Results small test instance

The small test instance allows for finding the exact solution using exact value iteration (EVI). To calculate the exact value function we use $\gamma = 0.9$ and tolerance parameter $\epsilon = 0.1$. The convergence criterion was met after seven iterations, taking over 12 h. We use the resulting value function throughout the remainder of this section.

We now use the exact value function V^* for the small test instance to measure which basis functions result in the closest approximation of V^*, through linear regression. The exact solution provides us with a full dataset of 373, 248 state-value pairs. We compare each of the proposed basis functions through 10-fold cross-validation on this dataset. The 10 subsets are selected randomly. We also compare pairwise combinations of basis functions. For several of these combinations, we found no optimal θ^* , and these combinations are left out of the results. Table 5 shows the average MSE for the (combinations of) basis functions.

It is clear that basis functions 1 and 2 result in the lowest MSE. It should be noted that those functions are very similar for the small test instance. For queues FA_1 and OR_1, having an access time target of 1, the number of early patients is equal to $s_{j,0}$, the number of on time patients is equal to $s_{j,1}$ and the number of late patients is equal to $s_{j,2}$. The only difference between the basis functions arises for queue OR_0 with an access time target of 0, as it is impossible to be early in that queue, and patients with a waiting time of 1 and 2 time periods are late. This explains why their MSE is almost exactly the same. It is interesting to see that, while basis functions 3 and 4 are of equal dimension, basis function 3 results in a much better approximation of the exact value function. The cost of a patient $c_{j,w}$ in queue $j \in \mathcal{J}$ waiting for $w \in \mathcal{W}_j$ time periods provides information that is important to be able to approximate the value of the state, as it gives an indication of the number of patients and their combined waiting times. Combining functions 3 and 4 provides better results, but not better than basis functions 1 and 2.

We now compare the performance of the approximate policy found through LSPI, the exact policy found through EVI, and the rolling-horizon LP on the small test instance with varying numbers of initial patients. We evaluate the rolling-horizon LP for the remainder of this paper, and thus relax the integrality constraint, as we perform thousands of experiments, and this reduces the computation time by a factor of approximately 100. This does come with a loss of optimality, and therefore,

Table 5 Mean squared errors of basis functions for 10-fold cross-validation on small test instance

Basis function	1	2	3	4	$3+4$
Average MSE	2.3055	2.3054	4.0474	12.0450	3.5708

we expect results to improve compared to the results shown in this paper when the integer constraint is applied.

We perform 50 trials per number of initial patients with 30 subsequent time periods each, and discount factor $\gamma = 0.9$ for all methods. We implement the rolling-horizon LP in Gurobi, and we set the time horizon to $T = 30$. For LSPI, we use basis function 1 and dataset size $M = 5000$. Figure 2 shows the results of this experiment. Recall that the contribution should be maximized.

We find that the performance of the policies resulting from LSPI and EVI is very similar. This implies that LSPI is able to approximate the exact value function well. Moreover, the rolling-horizon LP produces quite similar and sometimes better results than LSPI and EVI. This can be explained by the fact that we bound the length of the queues when determining the optimal policy using EVI (resulting in an approximation) and that LSPI uses an approximation of the value function. In the next section, we investigate various parameter settings to improve the LSPI algorithm.

6.3 Results large test instance

The large test instance, as described in Sect. 5.2, reflects a scaled-down version of the full instance at SMK, thus enabling us to perform experiments within reasonable time. This section describes the results of those experiments. The static allocation method is not considered here, as the optimal allocation for the large test instance does not provide us with useful information about the optimal allocation for the full instance. In Sect. 6.3.1, we determine the values of γ, ϕ, and M which result in the best performance of LSPI. Section 6.3.2 shows the value of γ that results in the best performance for the rolling-horizon LP. Section 6.3.3 discusses which decision rule performs best.

6.3.1 Settings for LSPI

For LSPI, only γ, ϕ, and M need to be chosen by the user (ignoring the constants ϵ and δ, as these do not have a large influence on the algorithm as long as they are chosen small enough). To improve the solution quality and/or speed up convergence, various methods have been proposed in literature, or can be devised by considering characteristics of the problem.

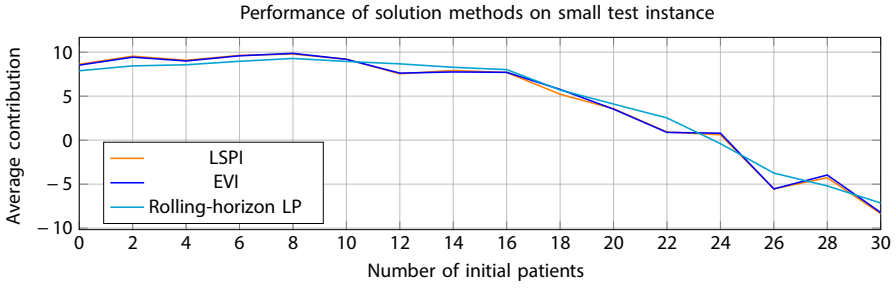

Fig. 2 Performance of LSPI, EVI, and rolling-horizon LP on the small test instance for varying numbers of initial patients

We evaluate the performance of different settings for LSPI on the large test instance with varying numbers of initial patients. We perform 50 trials per number of initial patients (drawn uniformly from the range [50, 70]) with the time horizon set to 30 and, unless stated otherwise, discount factor $\gamma = 0.75$, basis function 1, and a dataset size $M = 5000$. We use the same dataset for all experiments, except for the experiments on the size of the dataset. Since experiments have shown that the algorithm usually converges within approximately 4 to 6 iterations on the large test instance for similar parameter settings, we set an upper bound on the number of iterations of $N = 20$. If the convergence criterion has not been met by this point, we assume that the algorithm will never converge for the settings used. Figure 3 shows the average contribution of the LSPI for different settings.

Figure 3a shows the performance of the four different basis functions. LSPI using basis function 3 (the cost of patients in each queue) does not converge. Despite this, the combination of basis functions 3 and 4 does converge, and results in higher average contribution than basis function 4 alone. LSPI converged for all other basis functions in 4–6 iterations, with basis function 1 resulting in the best performance, and 4 in the worst. For this reason, basis function 1 (essentially the full state vector) is used for LSPI on the case study instance.

Figure 3b shows the performance for different discount factors. The discount factor γ determines to what extent we take into account future costs and rewards. If $\gamma = 0$, the policy is only based on the contribution gained by the current state and action, while for $\gamma \to 1$ we look infinitely far into the future. Intuitively, this tells us that increasing γ will improve the policy, but also makes the value function much more difficult to approximate and can result in no convergence. This intuition is confirmed by the experiments testing γ, where we see that the average contribution increases as γ increases, and LSPI stops converging for $\gamma \geq 0.8$. Choosing $\gamma \approx 0.6$ results in the best policies.

Figure 3c shows the performance for different dataset sizes. Increasing the size of the dataset that θ is trained on, M, should result in better policies as the algorithm can explore a larger part of the state space. The trade-off here is that this will increase the runtime of the algorithm (linearly). The results show that LSPI will converge reliably for $M \geq 2500$; the algorithm did not converge for $M = 500$ and $M = 1000$. The average contribution increases as M increases until $M = 10{,}000$. For values of M higher than 10, 000, the average contribution is lower. This could be a sign of overfit-

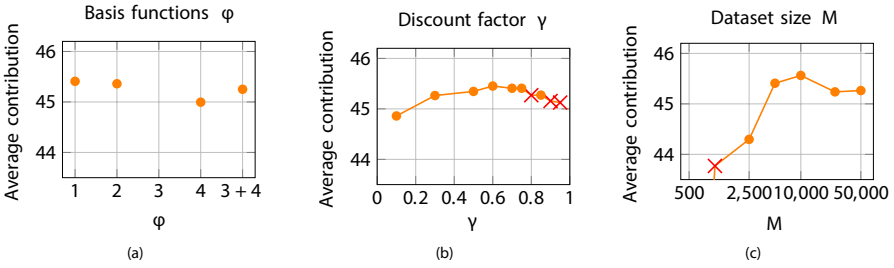

Fig. 3 Performance of LSPI on the large test instance for different parameter settings. Settings for which there was no convergence are marked with a red cross

ting, or it could indicate that, above a certain dataset size, the performance of LSPI is more sensitive to the contents of the dataset than to the size.

6.3.2 Setting γ for the rolling-horizon LP

The discount factor γ is, next to LSPI, also part of the objective function of the rolling-horizon LP. We evaluate the average contribution per time period and runtime of the rolling-horizon LP for the large test instance, where the average is taken over 50 trials for different values of γ. The time horizon, number of time periods, and number of initial patients are all set to 30. Results are shown in Fig. 4, which shows that initially, the average contribution and runtime increase as γ increases. Although there are minor fluctuations, the region $\gamma \in [0.3, 0.95]$ shows a stable performance, while there is a drop in performance from $\gamma = 0.95$ to 1. Runtime is stable for $\gamma \geq 0.1$.

6.3.3 Comparison of decision rules

We test the four decision rules described in Sect. 4.4 on the large test instance. Figure 5 shows the average contribution per time period gained over 50 trials when executing each of the decision rules for 30 subsequent time periods. The results show that the Highest Cost rule performs best, followed closely by the Highest Contribution rule. The Split Cost rule performs worst.

The Highest Contribution rule is a direct, greedy maximization of the contribution function within each time period, without taking future effects of the decisions into account. Technically, the rule is equivalent to the rolling-horizon LP with $\gamma = 0$, as in that case the action is selected which maximizes the contribution in only the current time period. Because of this, we expect the performance of the decision rule to be slightly worse than that of the rolling-horizon LP.

We expected the Highest Contribution rule to perform better than the Highest Cost rule as the former better reflects the contribution function. However, it seems that the high cost accumulated for untreated patients has a greater effect on the total contribution over the entire planning horizon.

6.4 Results case study

Now that we have determined which parameters to use for LSPI and the rolling-horizon LP, and the best performing decision rule, we apply the solution methods to the

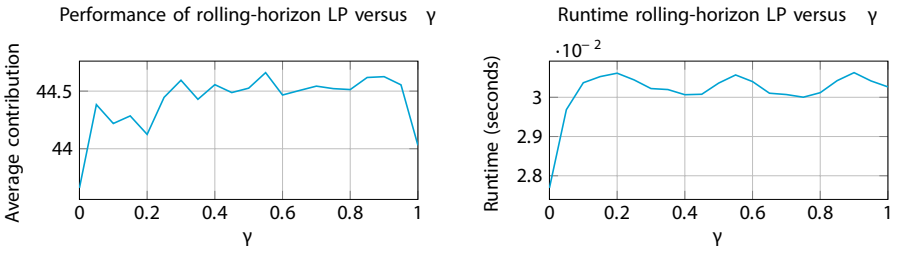

Fig. 4 Performance and runtime of the rolling-horizon LP for different values of γ

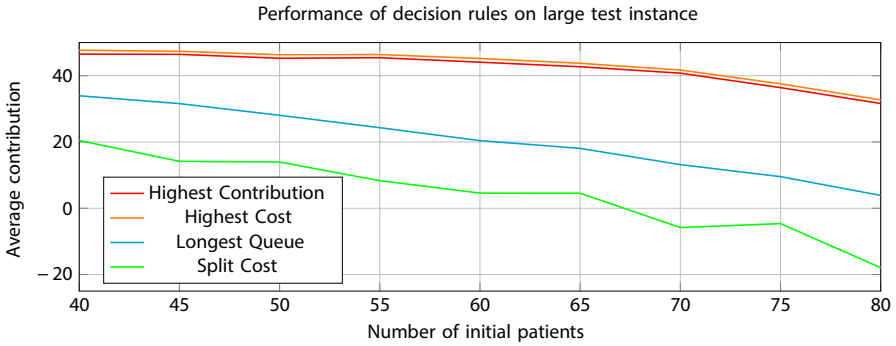

Fig. 5 Comparison of the decision rules for different numbers of initial patients

full timeslot allocation problem at SMK as described in Sect. 5.3. In Sect. 6.4.1, we compare the performance of the solution methods with and without planning ahead. Then, in Sect. 6.4.2, we stop using the estimation of performance and compare how well the LP, static allocation, and hybrid method perform on the actual access time targets and the number of treatments.

6.4.1 Comparison of solution methods

We compare the performance of the LSPI, the rolling-horizon LP, the Highest Cost decision rule, the static allocation used by SMK, and the hybrid method. We compare the average contribution over 100 trials, consisting of 30 subsequent time periods each, and the following settings for the solution methods:

- *LSPI*: Basis function 1, $\gamma = 0.6$, and dataset size $M = 50{,}000$. Because of the increased size of the problem, we expect a larger dataset than for the large test instance is necessary to achieve good results. The LSPI algorithm took 3 iterations, lasting 4 h, to converge to a parameter vector.
- *Rolling-horizon LP*: Time horizon $T = 13$ (a half year) and $\gamma = 0.75$ as we found that this discount factor performs better when planning ahead.
- *Heuristic decision rule*: Highest Cost, see Sect. 4.4.
- *Static allocation*: Through experiments with different allocations we found that approximately 50% of OD time should go to first appointments, 42.5% to follow-up appointments, and 7.5% to discharge appointments. The follow-up appointment slots are further allocated to the different urgencies: 40% to FU_3, 45% to FU_6, and 15% to FU_{12}. Given the low number of OR slots, we allocate 1 slot per period to OR_1 and OR_2, 2 to OR_4, and the remaining capacity to OR_6. If fewer than 10 OR-slots are available, OR_1 receives only 1 slot.
- *Hybrid allocation*: Settings for the static allocation and rolling-horizon LP as described above. SMK management decided that $\alpha = 0.6$ results in the best trade-off between performance and booking patients far in advance and that the dynamic fraction would be allocated only $p = 2$ periods in advance.

First, we assume perfect information about the state for which we must choose an action, so we do not plan ahead. Figure 6 shows the results for varying numbers of initial patients and two OR capacity settings. In the balanced setting, the OR capacity is 300 surgeries per year (or 365 in the first 30 periods), which enables stable queues. In the unbalanced setting, there is a shortage because the OR capacity is decreased to 220 surgeries per year (or 266 in the first 30 periods). Reducing the OR capacity in the unbalanced setting forces a choice between increasing OR access times, which would delay patient treatment, or decreasing the number of OD appointments to maintain a stable OR waiting list, limiting patient access to care.

Figure 6 shows that in the balanced setting, the LSPI, the Highest Cost decision rule, and the rolling-horizon LP result in a similar performance. In this simulation, the LSPI performed marginally better for 400 and 500 initial patients, while the LP performed best for over 500 patients. The static allocation resulted in the lowest contribution. In the unbalanced setting, overall performance declines since the reduced OR capacity reduces the achievable rewards and increases the expected costs. The differences between LSPI, the Highest Cost decision rule, and the rolling-horizon LP become slightly more pronounced, though their relative ranking is unchanged.

Second, we investigate what happens in the more realistic situation that timeslots must be allocated to appointment types several weeks in advance, based on a prediction of the future state, as explained in Sect. 3.4. Since our prediction of the future state will differ from the actual state, we expect to lose some performance quality.

Figure 7 compares the average contribution of the different solution methods, now including the hybrid method, for different values of p. The number of initial patients in each trial of the simulation is drawn from $\mathcal{N}(700, 200)$. When planning p weeks ahead, in each time period, we calculate the predicted state for the current time period, $\widetilde{s}_t$, using the true state p time periods ago, s_{t-p}, and the planned allocations in the meantime. The first $p+1$ time periods are ignored in the average contribution, as this is a warmup period where s_{t-p} does not yet exist. In addition to the average contribution, we include the 95% confidence intervals.

As anticipated, the performance of the rolling-horizon LP, LSPI, and the decision rule declines as the planning horizon increases, whereas the results of the static allocation and hybrid method are unaffected. The static allocation performs compara-

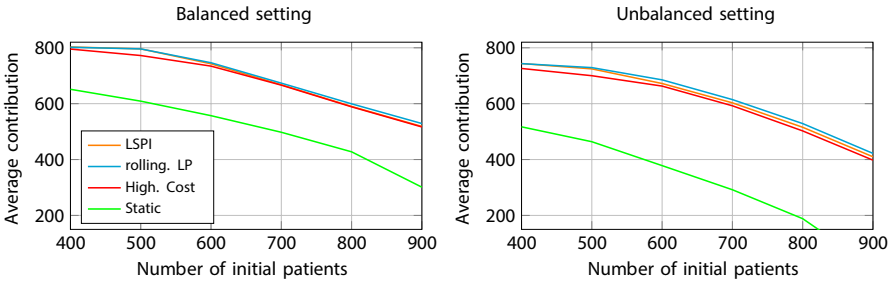

Fig. 6 Comparison of policies resulting from LSPI, the rolling-horizon LP, the highest cost decision rule, and static allocation on the case study instance when the number of initial patients is varied, for two settings

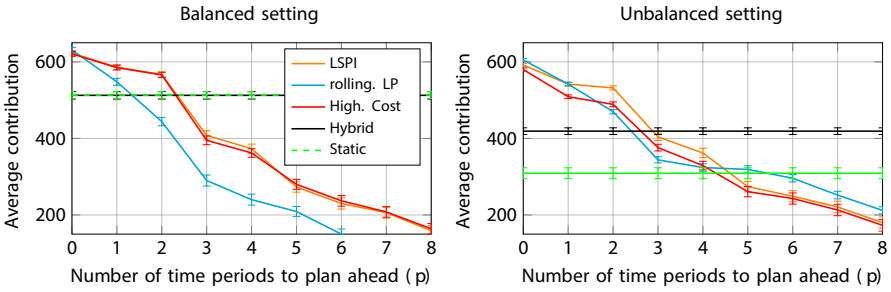

Fig. 7 Comparison of policies resulting from LSPI, the rolling-horizon LP, the highest cost decision rule, static allocation, and hybrid method on the case study instance when planning p time periods ahead for two settings

tively worse in the unbalanced setting than in the balanced setting, since it cannot adapt to imbalances; over time, waiting lists may become excessively long or short.

In the balanced setting, we conclude that the hybrid method and static allocation outperform the other methods. There is no significant difference in performance between them. Based on the confidence intervals, the Highest Cost decision rule and LSPI methods also perform equally well. The LP results in the worst performance for $p \geq 1$. This means that the fully dynamic methods for $p \geq 3$ overreact to the variability in the waiting lists, resulting in a decrease in performance instead of the intended improvement. This is in line with the practical experience at SMK, where interventions aimed at maintaining stable waiting lists have sometimes resulted in unexpected additional variability. Since the performance of the LP decreases as p increases, we expect the performance of the hybrid method also to decrease if it had to plan further ahead. In that case, the static allocation could become the best-performing method.

In the unbalanced setting, the rolling-horizon LP, LSPI, and the decision rule outperform the hybrid method for $p \leq 2$ and the static allocation for $p \leq 4$. For $p \geq 4$, the hybrid method performs significantly better than all other methods, as at $p = 3$, the difference between the hybrid method and the LSPI is not yet significant. The performance of the LP with respect to the other fully dynamic methods fluctuates, as it performs worse at $p = 2$ and $p = 3$, and better for $p > 4$.

Finally, we observe that for both settings, the LSPI and the decision rule significantly outperform the LP at $p = 2$. We expect that a hybrid method combining static allocation with LSPI or a decision rule will perform better than the current hybrid method at $p = 2$, both in the balanced setting and in the unbalanced setting.

6.4.2 Performance on treatments and access times

Until now, we have used the average contribution as an aggregate estimate of performance. We now compare how well the methods enable the case study hospital to treat as many patients as possible and to do so on time, as that is the aim of this paper (see Sect. 1). We present the results for the unbalanced setting and compare the static allocation method with the hybrid method for $\alpha = 0.6$ at $p = 2$ (4 weeks), and the rolling-horizon LP with $p = 6$ (12 weeks).

Table 6 Performance achieved by hybrid method ($p = 2, \alpha = 60\%$), rolling-horizon LP ($p = 6$), and static allocation

Queue	Treatments			Appointments on-time (%)		
	Hybrid	LP	Static	Hybrid	LP	Static
FA_2	1094 ± 2.9	1087 ± 6.8	986 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
FU_3	510 ± 4.2	502 ± 4.0	500 ± 4.1	95.0 ± 0.3	96.5 ± 0.2	94.4 ± 0.5
FU_6	576 ± 4.0	580 ± 5.5	594 ± 3.8	90.4 ± 0.3	85.6 ± 1.9	86.6 ± 1.1
FU_{12}	96 ± 1.9	81 ± 5.9	150 ± 2.1	40.8 ± 0.9	33.5 ± 6.5	92.1 ± 0.5
DA_3	204 ± 2.5	205 ± 2.8	210 ± 2.5	91.8 ± 0.4	87.6 ± 1.5	87.0 ± 1.3
Subtotal OD	2480 ± 3.5	2455 ± 11.3	2439 ± 6.2	49.7 ± 0.2	48.8 ± 0.6	53.5 ± 0.3
OR_1	39 ± 1.1	42 ± 1.1	34 ± 0.4	57.7 ± 3.7	65.3 ± 2.0	11.1 ± 3.3
OR_2	16 ± 0.9	13 ± 0.7	16 ± 0.8	87.8 ± 2.4	52.7 ± 4.8	82.5 ± 3.7
OR_4	25 ± 1.0	23 ± 0.7	25 ± 1.0	94.4 ± 1.3	85.7 ± 3.3	95.6 ± 1.1
OR_6	138 ± 1.6	151 ± 2.3	141 ± 0.0	55.2 ± 3.9	81.3 ± 2.2	77.5 ± 3.0
Subtotal OR	218 ± 2.0	230 ± 2.2	216 ± 1.4	62.2 ± 2.9	77.2 ± 1.5	69.3 ± 2.1
Total	2698 ± 4.1	2685 ± 12.3	2655 ± 6.4	50.7 ± 0.3	51.2 ± 0.6	54.8 ± 0.3

Table 6 shows, for each queue, the average number of treatments and the percentage of appointments that respect the maximum allowed access time of that queue. The averages and 95% confidence intervals are determined over 100 trials. The totals show that the hybrid method treats significantly more patients, but also the fewest patients within the maximum allowed access time. Although the rolling-horizon LP must allocate timeslots 12 weeks in advance, which appears to be a significant disadvantage, its performance approaches that of the hybrid method and includes more surgeries.

The data per queue show that, compared with the static allocation, the hybrid method significantly treats more urgent patients on time (FU_6, DA_3, and OR_1), and more FA_2 patients at the expense of elective patients (FU_{12} and OR_6). Although the hybrid method and LP treat approximately 100 FA_2 patients more than the static allocation, none of the methods succeed in treating any FA_2 patients on time. This indicates that the size of the initial FA_2 waiting list makes it infeasible to treat patients from both the external FA_2 waiting list and the internal waiting lists within the maximum allowed access time. Note that these results depend on the cost and reward parameters for the LP and hybrid method, and on the proportions used in the static allocation method. For example, increasing the proportion of first appointments in the static allocation, or assigning higher costs to FA patients, will improve external access times to first appointments but worsen internal access times to follow-up and discharge appointments. These results reflect the trade-offs made by the SMK. Other hospitals that want to use these methods will have to make their own trade-offs to achieve the desired behavior.

An important assumption underlying the results in Table 6 is that timeslots with no patients on the waiting list remain unused. For example, if the waiting list for OR_4 is empty when a timeslot is allocated, that timeslot goes unused. In practice, hospitals will book patients from different queues, but by not allowing this, we gain insight into how well the solution methods allocate timeslots to queues the first time right. We compare the OD and OR capacities with the number of time units used by the hybrid method and find that it utilizes 3574 of the 3839 available OD time units, or

93%. The OR department utilizes 218 of the 266 available time units, or 82%. For the remaining time units, the hospital can contact patients from other queues. The downside, however, is that this can lead to unexpected behavior and increase the forecast error. The simulation model used for this paper can be utilized to assess these effects.

It is evident that capacities allocated to surgeons and their waiting lists will not remain balanced forever, as, for instance, surgeons start or stop working at a hospital at some point. It is thus important to have a method that is at least partly dynamic. Given that the hybrid method can be set to static when required, we expect it to enable hospitals to achieve the best overall performance in situations where OD and OR capacities are balanced, as well as unbalanced. To ensure the hybrid method does not oversteer, we recommend investigating how to determine the appropriate level of α for different situations.

7 Conclusions and recommendations

In this paper, we allocate timeslots to patient groups with different appointment types and urgencies in outpatient and operating room sessions. Our goal is to increase the number of patients seen within their access time target, positively impacting their health and satisfaction. Furthermore, we aim to maximize the use of available resource capacity (i.e., OD and OR time) to treat as many patients as possible and minimize idle time. We classify the timeslot allocation problem as a sequential decision problem under uncertainty, and model it as a Markov Decision Process. We approached the timeslot allocation problem by developing several solution methods as alternatives to the current static allocation, namely Least-squares policy iteration (LSPI), a rolling-horizon Linear Program, several decision rules, and a hybrid method that statically allocates the fraction α of timeslots and dynamically allocates the remaining timeslots using the rolling-horizon LP. Since hospitals require timeslot schedules well in advance, we predict the future state for which to determine the timeslot allocation.

In comparison, for a case study of one surgeon at the orthopedic department of the Sint Maartenskliniek (SMK), the hybrid method resulted in the highest average contribution when planning ahead in a situation with an OR capacity shortage. When OD and OR capacities were balanced, the static allocation and hybrid method performed equally well, significantly outperforming the other methods. This suggests that when prior decisions, such as the OD and OR session planning, ensure balanced OD and OR waiting lists, it may be optimal to use a static allocation. This is an important insight, as static allocation is a common practice in hospitals.

We then compared the performance per queue for the hybrid method, the static allocation, and rolling-horizon LP with a planning horizon of 12 weeks. We calculated the total number of treatments and the percentage of appointments realized within the maximum allowed access time for the unbalanced setting. Given the configuration of cost and reward parameters used and in comparison with the static allocation, the hybrid method resulted in a significant increase in total OD treatments and on-time appointments for several urgent patient groups, at the cost of a decrease in the percentage of on-time appointments for the least urgent patients. This is desirable

behavior for the case study hospital. Adjusting the cost and reward parameters would alter the prioritization of patients by the LP, resulting in different outcomes.

Our recommendation for implementation depends on the planning-horizon requirements and available software and technical expertise. In cases where all timeslots need to be allocated more than 12 weeks in advance, we recommend using the static allocation method, while making an effort to align the inflow of patients and the OD and OR capacities as much as possible. When it is possible to allocate a fraction of the timeslots only 4 weeks in advance, we recommend using a hybrid method, as it benefits from the accurate and up-to-date information available when planning closer to the actual date. The hybrid method, using a rolling-horizon LP, outperformed all dynamic methods in the case study. However, it requires specialised software and access to mathematical expertise for implementation and maintenance. If specialized solvers, such as Gurobi, are available and the output can be uploaded to the Hospital Information System, we recommend using the rolling-horizon ILP. Relaxing the integer constraints would allow the use of an open-source solver. Suppose the use of an (I)LP is not feasible. In that case, we recommend evaluating a hybrid method with the Highest Cost decision rule for the dynamic fraction, since this approach requires less complex software, is likely faster, and could potentially perform even better.

In line with these recommendations and considerations, SMK decided to implement hybrid timeslot allocation for the outpatient clinic. Depending on the variability per surgeon, up to 20% of OD sessions are allocated dynamically 4 weeks in advance. Experienced planners determine the dynamic allocation in the weekly Operational Planning Meeting. Since the planning department ensures that the appointment schedules for the coming 3 weeks are fully booked, they can accurately allocate the timeslots for the fourth week based on the remaining waiting lists. This way, the hospital was able to improve performance.

For hospitals considering the hybrid method, we recommend confirming $\alpha = 0.6$ per surgeon and determining whether it is possible to plan the dynamic fraction at $p = 1$ or $p = 2$. Furthermore, carefully select costs and rewards to reflect the required performance. To determine the correct values, hospitals can, as in this paper, manually simulate the performance of allocations derived using different cost and reward settings, use the automatic iterative method as proposed by Hulshof et al. (2013), or derive the costs and rewards from actual financial costs associated with wasting resource capacity and increasing access times. We focused on comparing solution methods, not on tuning these values.

Our results indicate that the static method performs best in a balanced system when timeslots must be allocated more than 4 weeks in advance. This seems natural, as steering is unnecessary in a balanced system. In the unbalanced setting, where the OR or OD capacities do not match patients' capacity requirements, steering capacity towards patient groups with the highest costs seems natural, as also shown by our results. The point at which the system switches between balanced and unbalanced is, both in practice and in theory, hard to determine and is an open question for further research.

Since the LSPI and Highest Cost decision rule significantly outperform the rolling horizon LP at $p = 2$ in both settings, a hybrid method incorporating one of these methods for the dynamic fraction could potentially outperform the current hybrid

method. Although the LSPI performs better than the decision rule in the unbalanced setting, we expect that the three major drawbacks of LSPI relative to a decision rule will persist: complexity of implementation, runtime, and explainability. First, the algorithm requires more data collection to program a simulation. Convergence is very sensitive to the settings of parameters and problem size. Second, on realistic problem sizes, the algorithm takes hours to converge, making experimentation with parameters time-consuming. For practical use, the algorithm would need to run every few months to ensure that model parameters are up-to-date. The high runtime also implies that we cannot hope to expand the model to optimize for multiple surgeons simultaneously. The third and perhaps most significant drawback for practical implementation is that the algorithm is a black box: we cannot reasonably explain why it chooses one action over another. In healthcare applications, this means that we cannot explain why we accept one patient but reject another. A hybrid method that uses the Highest Cost decision rule does not have these drawbacks. It is easier to implement, runs faster, and is fully explainable.

A natural extension of this work is to expand the model to incorporate interactions between surgeons. For example, at SMK, new patients waiting for a first appointment are not assigned to a single surgeon, but to a subspecialty consisting of multiple surgeons. We expect that it is possible to expand the ILP formulated in this paper to model a subspecialty, or even the entire orthopedic department. Expansion would make the program larger, and integer constraints on variables would be necessary to allocate whole patients to surgeons, significantly increasing solving time. On the other hand, calculating timeslot allocations for all surgeons simultaneously would be less work than calculating them for each surgeon separately. Furthermore, taking interaction effects into account could significantly improve performance for the group of surgeons as a whole.

References

Ahmadi-Javid A, Jalali Z, Klassen KJ (2017) Outpatient appointment systems in healthcare: a review of optimization studies. Eur J Oper Res 258(1):3–34. https://doi.org/10.1016/j.ejor.2016.06.064

Ala A, Chen F (2022) Appointment scheduling problem in complexity systems of the healthcare services: a comprehensive review. J Healthc Eng. https://doi.org/10.1155/2022/5819813

Aslani N, Kuzgunkaya O, Vidyarthi N, Terekhov D (2021) A robust optimization model for tactical capacity planning in an outpatient setting. Health Care Manag Sci 24(1):26–40. https://doi.org/10.1007/s10729-020-09528-y

Bakker M, Tsui K-L (2017) Dynamic resource allocation for efficient patient scheduling: a data-driven approach. J Syst Sci Syst Eng 26(4):448–462. https://doi.org/10.1007/s11518-017-5347-3
Bradtke SJ, Barto AG (1996) Linear least-squares algorithms for temporal difference learning. Mach Learn 22:33–57. https://doi.org/10.1007/BF00114723
Deglise-Hawkinson J, Helm JE, Huschka T, Kaufman DL, Van Oyen MP (2018) A capacity allocation planning model for integrated care and access management. Prod Oper Manag 27(12):2270–2290. https://doi.org/10.1111/poms.12941
Erdelyi A, Topaloglu H (2011) Approximate dynamic programming for dynamic capacity allocation with multiple priority levels. IIE Trans (Inst Ind Eng) 43(2):129–142. https://doi.org/10.1080/0740817X.2010.504690
Hattingh M (2019) Decision support for planning and scheduling orthopaedic surgeons. Technical report, University of Twente
Hulshof PJH, Boucherie RJ, Hans EW, Hurink JL (2013) Tactical resource allocation and elective patient admission planning in care processes. Health Care Manag Sci 16(2):152–166. https://doi.org/10.1007/s10729-012-9219-6
Hulshof PJH, Mes MRK, Boucherie RJ, Hans EW (2016) Patient admission planning using approximate dynamic programming. Flex Serv Manuf J 28(1–2):30–61. https://doi.org/10.1007/s10696-015-9219-1
Koutroumbas K, Theodoridis S (2008) Pattern recognition. Academic Press. ISBN 9780080949123
Laan C, van de Vrugt M, Olsman J, Boucherie RJ (2018) Static and dynamic appointment scheduling to improve patient access time. Health Syst 7(2):148–159. https://doi.org/10.1080/20476965.2017.1403675
Lagoudakis MG, Parr R (2003) Least-squares policy iteration. J Mach Learn Res 4(6):1107–1149. https://doi.org/10.1162/1532443041827907
Nederlandse Zorgautoriteit (2020) Zorgplicht zorgkantoren—beeld 2019 en uitdagingen (deel 3 bij Onderzoeksrapporten toezicht op langdurige zorg). Technical report, Nederlandse Zorgautoriteit. http://puc.overheid.nl/doc/PUC_624861_22
Nguyen TBT, Sivakumar AI, Graves SC (2015) A network flow approach for tactical resource planning in outpatient clinics. Health Care Manag Sci 18(2):124–136. https://doi.org/10.1007/s10729-014-9284-0
Nunes LGN, de Carvalho SV, Rodrigues RCM (2009) Markov decision process applied to the control of hospital elective admissions. Artif Intell Med 47(2):159–171. https://doi.org/10.1016/j.artmed.2009.07.003
Patrick J, Puterman ML, Queyranne M (2008) Dynamic multipriority patient scheduling for a diagnostic resource. Oper Res 56(6):1507–1525. https://doi.org/10.1287/opre.1080.0590
Powell WB (2019) Reinforcement learning and stochastic optimization problem: a unified framework for sequential decisions. Wiley, Hoboken
Tsai ER (2017) Optimal time allocation of an orthopedic surgeon. Technical report, University of Twente. https://purl.utwente.nl/essays/73487

Authors and Affiliations

R. F. M. Vromans[1,2] · J. T. van Essen[3] · Y. M. van der Vlugt[3] · M. Carlier[2]

✉ J. T. van Essen
j.t.vanessen@tudelft.nl

[1] University of Twente, Enschede, The Netherlands

[2] Rhythm BV, Amsterdam, The Netherlands

[3] Delft University of Technology, Delft, The Netherlands