Memorandum 2014, Department of Applied Mathematics, University of Twente, the Netherlands

Tactical Planning in Healthcare using Approximate Dynamic Programming

Peter J.H. Hulshof[⋆⋆] · Martijn R.K. Mes[⋆⋆] · Richard J. Boucherie · Erwin W. Hans

Abstract Tactical planning of resources in hospitals concerns elective patient admission planning and the intermediate term allocation of resource capacities. Its main objectives are to achieve equitable access for patients, to serve the strategically agreed number of patients, and to use resources efficiently. We propose a method to develop a tactical resource allocation and patient admission plan that takes stochastic elements into consideration, thereby providing robust plans. Our method is developed in an Approximate Dynamic Programming (ADP) framework and copes with multiple resources, multiple time periods and multiple patient groups with various uncertain treatment paths through the hospital and an uncertain number of arrivals in each time period, thereby integrating decision making for a chain of hospital resources. Computational results indicate that the ADP approach provides an accurate approximation of the value functions, and that it is suitable for large problem instances at hospitals, in which the ADP approach performs significantly better than two other heuristic approaches. Our ADP algorithm is generic, as various cost functions and basis functions can be used in various settings of tactical hospital management.

Keywords Healthcare · tactical planning · resource capacity planning · patient admission planning · Dynamic Programming (DP) · Approximate Dynamic Programming (ADP)

⋆ This research is supported by the Dutch Technology Foundation STW, applied science division of NWO and the Technology Program of the Ministry of Economic Affairs.

⋆⋆ Corresponding authors: University of Twente, Peter J.H. Hulshof and Martijn R.K. Mes, P.O. Box 217, 7500 AE Enschede, the Netherlands. E-mail: {p.j.h.hulshof,m.r.k.mes}@utwente.nl.

Center for Healthcare Operations Improvement and Research (CHOIR), University of Twente, Enschede, the Netherlands

1 Introduction

Tactical planning is a key element of hospital planning and control that concerns the intermediate term allocation of resource capacities and elective patient admission planning [9]. Its main objectives are to achieve equitable access and treatment duration for patient groups, to serve the strategically agreed target number of patients (i.e., production targets or quota), to maximize resource utilization and to balance workload [10].

From a clinician's perspective, tactical resource and admission plans break the clinician's time down into separate activities (e.g., consultation time and surgical time) and determine the number of patients to serve from a particular patient group at a particular stage of their care process (e.g, consultation or surgery). We use the term care process to identify a chain of care stages for a patient, for example a visit to an outpatient clinic, a surgery, and a revisit to the outpatient clinic. At each stage in the care process, patients incur access time, which is the time spent on the waiting list before being served. Controlled access times ensure quality of care for the patient and prevent patients from seeking treatment elsewhere [22]. The term care process is not to be confused with "clinical pathway", which is described in [5].

Care processes connect multiple departments and resources together as an integrated network. Fluctuations in both patient arrivals (e.g., seasonality) and resource availability (e.g., holidays) at one department may impact the entire care chain. For patients, this results in varying access times for each separate stage in a care process, and from a hospital's perspective, this results in varying resource utilizations and service levels. To cope with these fluctuations, intermediate-term re-allocation of hospital resources, taking into account a care chain perspective [3, 8, 15], seems necessary.

The tactical planning problem in healthcare is stochastic in nature. Randomness exists in for example the number of (emergency) patient arrivals and the number of patient transitions after being treated at a particular stage of their care process. Several papers have focused on tactical planning problems that span multiple departments and resources in healthcare [6, 11, 13] and other industries [7]. In [10], the literature and various applications is reviews and it is concluded that existing approaches to develop tactical resource and admission plans in the OR/MS literature are myopic, focus on developing long-term cyclical plans, or are not able to provide a solution for real-life instances. The authors develop a deterministic method for tactical planning over multiple departments and resources within a mathematical programming framework.

In this paper, we develop a stochastic approach for the tactical planning problem in healthcare by modeling it as a Dynamic Programming problem (DP). Due to the properties of the tactical planning problem, with discrete time periods and transitions that depend on the decision being made, DP is a suitable modeling approach. As problem sizes increase, solving a DP is typically intractable due to the 'curse of dimensionality'. To overcome this problem, an alternative solution approach for real-life sized instances of the tactical planning problem is needed. The field of Approximate Dynamic Pro-

gramming (ADP) provides a suitable framework to develop such an alternative approach, and we use this framework to develop an innovative solution approach. ADP uses approximations, simulations and decompositions to reduce the dimensions of a large problem, thereby significantly reducing the required calculation time. A comprehensive explanation and overview of the various techniques within the ADP framework are given in [16]. The application of ADP is relatively new in healthcare, it has been used in ambulance planning [12, 18] and patient scheduling [14]. Other applications in a wider spectrum of industries include resource capacity planning [4, 19], inventory control [20], and transportation [21].

We aim to contribute to the literature in two ways. First, we provide a theoretical contribution to the development of tactical resource and admission plans in healthcare in the field of Operations Research and Management Science (OR/MS). We develop an approach to develop tactical plans that take randomness in patient arrivals and patient transitions to other stages into account. These plans are developed for multiple resources and multiple patient groups with various care processes, thereby integrating decision making for a chain of hospital resources. The model is designed with a finite horizon, which allows all input to be time dependent. This enables us to incorporate anticipated or forecasted fluctuations between time periods in patient arrivals (e.g., due to seasonality) and resource capacities (e.g., due to vacation or conference visits) in developing the tactical plans. The model can also be used in 'realtime'. If during actual implementation of the tactical plan, deviations from forecasts make reallocation of resource capacity necessary, the developed model can be used to determine an adjusted tactical plan. The model can be extended to include different cost structures, constraints, and additional stochastic elements. Second, the solution approach is innovative as it combines various methods and techniques within the ADP-framework and the field of mathematical programming. Also, the application of ADP is new in tactical resource capacity and patient admission planning, and relatively new in healthcare in general, where it has mainly been applied in ambulance planning [12, 18] and patient scheduling [14].

This paper is organized as follows. Section 2 discusses the mathematical problem formulation, and Section 3 describes the exact Dynamic Programming (DP) solution approach for small instances. Section 4 introduces the ADP approaches necessary to develop tactical plans for real-life sized instances. Section 5 describes how the model can be used to develop or adjust tactical plans in healthcare. Section 6 discusses computational results and Section 7 concludes this paper.

2 Problem formulation

This section introduces the problem and the patient dynamics in care processes. We provide notation and present the stochastic model formulation of

the problem that captures randomness in patient arrivals and patient transitions between queues.

The planning horizon is discretized in consecutive time periods $\mathcal{T} = \{1, 2, \ldots, T\}$. This finite horizon allows all input to the model to be time dependent and enables incorporating anticipated or forecasted fluctuations between time periods in patient arrivals and resource capacities. We consider a set of resource types $\mathcal{R} = \{1, 2, \ldots, R\}$ and a set of patient care processes $\mathcal{G} = \{1, 2, \ldots, G\}$. Each of these care processes consists of a set of stages $\mathcal{K}_g = \{1, ..., e_g\}$, where e_g gives the number of stages in the care process $g \in \mathcal{G}$. To simplify notation, we denote each stage in $\cup_{g \in \mathcal{G}} \mathcal{K}_g$ by a queue j. We introduce the set $\mathcal{J}$ as the set of all queues and $\mathcal{J}^r$ as the set of queues that require capacity of resource $r \in \mathcal{R}$.

Each queue $j \in \mathcal{J}$ requires a given amount of time units from one or more resources, given by $s_{j,r}$, $r \in \mathcal{R}$, and different queues may require the same resource. The number of patients that can be served by resource $r \in \mathcal{R}$ is limited by the available resource capacity $\eta_{r,t}$ in time period $t \in \mathcal{T}$. Because we also allow for queues without resource requirement (dummy queues), $\cup_{r \in \mathcal{R}} \mathcal{J}^r$ is not necessarily equal to $\mathcal{J}$.

After being treated at a queue $j \in \mathcal{J}$, patients either leave the system or join another queue. To model these transitions, we introduce $q_{j,i}$ which denotes the fraction of patients that will join queue $i \in \mathcal{J}$ after being treated in queue $j \in \mathcal{J}$. The value $q_{j,0} = 1 - \sum_{i \in \mathcal{J}} q_{j,i}$ denotes the fraction of patients that leave the system after being treated at queue $j \in \mathcal{J}$. In general, $q_{j,i}$ is positive when $i \in \mathcal{J}$ is immediately succeeding $j \in \mathcal{J}$ in the same care process. However, our modeling framework allows for different types of transitions (for example transitions to any prior or future stage in the same care process, and transitions between queues of different care processes). In addition to demand originating from the treatment of patients at other queues within the system, there is also demand from outside the system. The number of patients arriving from outside the system to queue $j \in \mathcal{J}$ at time $t \in \mathcal{T}$ is given by $\lambda_{j,t}$. As can be observed, our model has a finite horizon.

Within the definition of $q_{j,i}$ lies the major assumption of our model:

Assumption 1 *Patients are transferred between the different queues according to transition probabilities $q_{j,i}, \forall j, i \in \mathcal{J}$ independent of their preceding stages, independent of the state of the network and independent of the other patients.*

For practical purposes in which Assumption 1 does not hold, we can adjust the various care processes to ensure it does hold. For example, if after some stage within a care process, the remainder of the patient's path depends on the current stage, we create a new care process for the remaining stages and patients flow with a certain probability to the first queue in that new care process.

For the arrival processes, we assume the following.

Assumption 2 *Patients arrive at each queue from outside the system according to a Poisson process with rate $\lambda_{j,t}$, $\forall j \in \mathcal{J}, t \in \mathcal{T}$. The external arrival*

process at each queue $j \in \mathcal{J}$ in time period $t \in \mathcal{T}$ is independent of the external arrival process at other queues and other time periods. Since all arrival processes are independent, we obtain $\lambda_{0,t} = \sum_{j=1}^{|\mathcal{J}|} \lambda_{j,t}$, $\forall t \in \mathcal{T}$.

We introduce $\mathcal{U} = \{0, 1, 2, ..., U\}$ to represent the set of time periods patients can be waiting. Given Assumption 1, patients are characterized by the queue in which they are waiting and the amount of time they have been waiting at this queue. We introduce

$$S_{t,j,u} = \text{Number of patients in queue } j \in \mathcal{J} \text{ at time } t \in \mathcal{T} \text{ with a waiting time of } u \in \mathcal{U}.$$

The state of the system at time period t can be written as S_t, which is a matrix made up of the elements $(S_{t,j,u})$, for all $t \in \mathcal{T}, j \in \mathcal{J}$, and $u \in \mathcal{U}$. We define decisions as actions that can change the state of the system. The decisions are given by

$$x_{t,j,u} = \text{Number of patients to treat in queue } j \in \mathcal{J} \text{ at time } t \in \mathcal{T}, \text{ with a waiting time of } u \in \mathcal{U}.$$

The decision at time period t can be written as $x_t = (x_{t,j,u})_{j \in \mathcal{J}, u \in \mathcal{U}}$, in the same way as we write the state description S_t. The cost function $C_t(S_t, x_t)$ related to our current state S_t and decision x_t can be modeled in various ways. The main objectives of tactical planning are *to achieve equitable access and treatment duration for patient groups* and *to serve the strategically agreed number of patients* [10]. The focus in developing this model is on the patient's waiting time (equitable access and treatment duration), and we assume that the strategically agreed number of patients is set in accordance with patient demand (as the model accepts all patients that arrive). The cost function in our model is set-up to control the waiting time per stage in the care process, so per individual queue ($j \in \mathcal{J}$). It is also possible to adapt the cost function for other tactical planning settings, for example to control the total waiting time per individual care process $g \in \mathcal{G}$ or for all queues that use a particular resource $r \in \mathcal{R}$. We choose the following cost function, which is based on the number of patients for which we decide to wait at least one time unit longer

$$C_t(S_t, x_t) = \sum_{j \in \mathcal{J}} \sum_{u \in \mathcal{U}} c_{j,u} \left(S_{t,j,u} - x_{t,j,u}\right), \qquad \forall t \in \mathcal{T}. \tag{1}$$

The cost component $c_{j,u}$ in (1) is set by the hospital to distinguish between queues $j \in J$ and waiting times $u \in \mathcal{U}$. In general, higher $u \in \mathcal{U}$ will have higher costs as it means a patient has a longer total waiting time. This could be modeled in various ways, for example the cost $c_{j,u}$ could be incrementally increasing with $u \in \mathcal{U}$ or the hospital has some target/threshold waiting time after which waiting costs increase significantly.

The Integer Linear Programming (ILP) version of the introduced problem can be written as

$$\min \sum_{t \in \mathcal{T}} C_t(S_t, x_t) = \sum_{t \in \mathcal{T}} \sum_{j \in \mathcal{J}} \sum_{u \in \mathcal{U}} c_{j,u} \left(S_{t,j,u} - x_{t,j,u}\right), \tag{2}$$

subject to

$$S_{t,j,0} = \lambda_{j,t} + \sum_{i \in \mathcal{J}} \sum_{u \in \mathcal{U}} q_{i,j} x_{t-1,i,u}, \qquad \forall j \in \mathcal{J}, t \in \mathcal{T}, \tag{3}$$

$$S_{t,j,U} = \sum_{u=U-1}^{U} \left(S_{t-1,j,u} - x_{t-1,j,u} \right), \qquad \forall j \in \mathcal{J}, t \in \mathcal{T}, \tag{4}$$

$$S_{t,j,u} = S_{t-1,j,u-1} - x_{t-1,j,u-1}, \qquad \forall j \in \mathcal{J}, t \in \mathcal{T}, u \in \mathcal{U} \backslash \{0, U\}, \tag{5}$$

$$x_{t,j,u} \leq S_{t,j,u}, \qquad \forall j \in \mathcal{J}, t \in \mathcal{T}, u \in \mathcal{U}, \tag{6}$$

$$\sum_{j \in \mathcal{J}^r} s_{j,r} \sum_{u \in \mathcal{U}} x_{t,j,u} \leq \eta_{r,t}, \qquad \forall r \in \mathcal{R}, t \in \mathcal{T}, \tag{7}$$

$$x_{t,j,u} \in \mathbb{Z}_+, \qquad \forall j \in \mathcal{J}, t \in \mathcal{T}, u \in \mathcal{U}. \tag{8}$$

Constraints (3) to (5) stipulate that the waiting list variables are consistent. Constraint (3) determines the number of patients newly entering a queue. Constraint (4) updates the waiting list for the longest waiting patients per queue (which is bounded by U). Constraint (5) updates the waiting list variables at each time period for all $u \in \mathcal{U}$ that are not covered by the first two constraints. Constraint (6) stipulates that not more patients are served than the number of patients on the waiting list. Constraint (7) assures that the resource capacity of each resource type $r \in \mathcal{R}$ is sufficient to serve all patients. Constraint (8) is an integrality constraint for the number of patients to serve of each type at each time period.

The above ILP version does not incorporate the different forms of randomness that are apparent in the actual system, such as random patient arrivals and uncertainty in patient transitions to other queues. The ILP uses approximations in the form of the expectation for those processes. More specifically, $\lambda_{i,t}$ and $q_{j,i}$, $i, j \in \mathcal{J}, t \in \mathcal{T}$ in Constraint 3 actually are parameters for stochastic processes. To capture all sources of random information, we introduce

$$W_t = \text{ The vector of random variables representing all the new information that becomes available between time } t-1 \text{ and } t.$$

The vector W_t contains all the new information, which consists of new patient arrivals and outcomes for transitions between queues. We distinguish between *exogeneous* and *endogeneous* information in

$$W_t = \left(\widehat{S}_t^e, \widehat{S}_t^o \left(x_{t-1} \right) \right), \qquad \forall t \in \mathcal{T},$$

where the exogeneous $\widehat{S}_t^e = \left(\widehat{S}_{t,j}^e \right)_{\forall j \in \mathcal{J}}$ represents the patient arrivals from outside the system, and the endogeneous $\widehat{S}_t^o \left(x_{t-1} \right) = \left(\widehat{S}_{t,j,i}^o \left(x_{t-1} \right) \right)_{\forall i,j \in \mathcal{J}}$ represents the patient transitions to other queues as a function of the decision vector x_{t-1}. $\widehat{S}_{t,j,i}^o \left(x_{t-1} \right)$ gives the number of patients transferring from queue $j \in \mathcal{J}$ to queue $i \in \mathcal{J}$ at time $t \in \mathcal{T}$, depending on the decision vector x_{t-1}.

Assumptions 1 and 2 imply that the probability distribution (conditional on the decision) of future states only depends on the current state, and is independent of preceding states in preceding time periods. This means that the described process has the Markov property. We use this property in defining a transition function, S^M, to capture the evolution of the system over time as a result of the decisions and the random information.

$$S_t = S^M\left(S_{t-1}, x_{t-1}, W_t\right), \tag{9}$$

where

$$S_{t,j,0} = \widehat{S}^e_{t,j} + \sum_{i\in\mathcal{J}} \widehat{S}^o_{t,i,j}\left(x_{t-1,i}\right), \quad \forall j \in \mathcal{J}, t \in \mathcal{T}, \tag{10}$$

$$S_{t,j,U} = \sum_{u=U-1}^{U} \left(S_{t-1,j,u} - x_{t-1,j,u}\right), \quad \forall j \in \mathcal{J}, t \in \mathcal{T}, \tag{11}$$

$$S_{t,j,u} = S_{t-1,j,u-1} - x_{t-1,j,u-1}, \quad \forall j \in \mathcal{J}, t \in \mathcal{T}, u \in \mathcal{U}\backslash\{0, U\}, \tag{12}$$

are the stochastic counterparts of the first constraints (3) to (5) in the ILP formulation. The stochastic information is captured in (10). All arrivals in time period $t \in \mathcal{T}$ to queue $j \in \mathcal{J}$ from outside the system ($\widehat{S}^e_{t,j}$) and from internal transitions ($\sum_{i\in\mathcal{J}} \widehat{S}^o_{t,i,j}\left(x_{t-1,i}\right)$) are combined in (10).

We aim to find a policy (a decision function) to make decisions about the number of patients to serve at each queue. We represent the decision function by

$$X^\pi_t\left(S_t\right) = \text{ A function that returns a decision } x_t \in \mathcal{X}_t\left(S_t\right), \text{under the policy } \pi \in \Pi.$$

The set Π refers to the set of potential decision functions or policies. $\mathcal{X}_t$ denotes the set of feasible decisions at time t, which is given by

$$\begin{aligned} \mathcal{X}_t\left(S_t\right) = \{\, x_t | \\ & x_{t,i,u} \le S_{t,i,u}, && \forall i \in \mathcal{J}, t \in \mathcal{T}, u \in \mathcal{U} \\ & \textstyle\sum_{j\in\mathcal{J}^r} s_{j,r} \sum_{u\in\mathcal{U}} x_{t,j,u} \le \eta_{r,t}, && \forall r \in \mathcal{R}, t \in \mathcal{T} \\ & x_{t,j,u} \in \mathbb{Z}_+ && \forall i \in \mathcal{J}, t \in \mathcal{T}, u \in \mathcal{U}\}. \end{aligned} \tag{13}$$

As given in (13), the set of feasible decisions in time period t is constrained by the state space S_t and the available resource capacity $\eta_{r,t}$ for each resource type $r \in \mathcal{R}$. Our goal is to find a policy π, among the set of policies Π, that minimizes the expected costs over all time periods given initial state S_0. This goal is given in

$$\min_{\pi\in\Pi} \mathbb{E}\left\{\sum_{t\in\mathcal{T}} C_t\left(S_t, X^\pi_t\left(S_t\right)\right) | S_0\right\}, \tag{14}$$

where $S_{t+1} = S^M\left(S_t, x_t, W_{t+1}\right)$. Note that the description in (14) is in line with the ILP's objective function in (2). The challenge is to find the best policy $X^\pi_t\left(S_t\right)$.

Remark 3 Incorporated in the formulation of the model is the assumption that after a treatment decision x_t at the beginning of time t, patients immediately generate waiting costs in the following queue (if they move on to a following care stage, and do not exit the system) after entering that queue in time period $t+1$. In practice, after a treatment, a patient may require to wait a minimum time lag before a follow-up treatment can be initiated. The model can be extended to cover cases with time lags $d_{i,j}$ (time lag in the transition from queue i to queue j) by allowing u to be negative in $S_{t,j,u}$. For example, $S_{t,j,-2}$ then indicates the number of patients that will enter queue j two time periods from now. Incorporating this time lag changes the system dynamics: patients with $u < 0$ cannot be served and we set $C_{t,j,u}\left(S_{t,j,u}, x_{t,j,u}\right) = 0$ for $u < 0$, $\forall i \in \mathcal{J}, t \in \mathcal{T}, u \in \mathcal{U}$.

The various sets, indices, state descriptions and parameters that will be used in following sections are given in Table 1

Sets		**Indices**	
$\mathcal{G}$	Care processes	$g \in \mathcal{G}$	Care process
$\mathcal{J}$	Queues	$i, j \in \mathcal{J}$	Queue
$\mathcal{T}$	Time periods	$t \in \mathcal{T}$	Time period
$\mathcal{R}$	Resource types	$r \in \mathcal{R}$	Resource type
$\mathcal{U}$	Time periods (to indicate waiting time)	$u \in \mathcal{U}$	Waiting time period
$\mathcal{J}^r$	Queues for resource type r	$i, j \in \mathcal{J}^r$	Queue
Decision variables			
$x_{t,j,u}$	Number of patients to treat in queue j in time period t, who have been waiting u time periods		
State description			
S_t	State of the system at time period t		
$S_{t,j,u}$	Number of patients in queue j in time period t with a waiting time of u		
Parameters			
$c_{j,u}$	Costs for queue j and u time periods waiting		
$\lambda_{j,t}$	New demand in queue j in time period t		
$\eta_{r,t}$	Capacity of resource type r in time period t in time units		
$q_{i,j}$	Probability that a patient moves from queue i to queue j		
$s_{j,r}$	Expected capacity requirements from resource type r for a patient in queue j in time units		

Table 1 The sets, indices, variables, state description and parameters used.

3 Dynamic Programming

To solve the problem formulated in Section 2, we propose an exact DP approach. The DP approach is suitable due to the finite horizon of the problem, the decision dependent transitions of patients, and the randomness in patient arrivals and patient transitions. In the following, the DP approach is explained in detail. First, we state the optimality equation, and second, we elaborate the expectation in that optimality equation.

By the principal of optimality [1], we can find the optimal policy by solving

$$V_t\left(S_t\right) = \min_{x_t \in \mathcal{X}_t(S_t)} \left(C_t\left(S_t, x_t\right) + \mathbb{E}\left\{V_{t+1}\left(S_{t+1}\right) | S_t, x_t, W_{t+1}\right\}\right), \tag{15}$$

where $S_{t+1} = S^M\left(S_t, x_t, W_{t+1}\right)$ gives the state S_{t+1} as a function of the current state S_t, the decisions x_t, and the new information W_{t+1}.

The optimal decision minimizes the value that is calculated with the value function $V_t\left(S_t\right)$. In the value function, 'direct' costs are incurred for the decision in the current time period ($C_t\left(S_t, x_t\right)$), and 'future' costs reflect the expected costs in future time periods, *as a result* of the decision in the current time period ($\mathbb{E}\left\{V_{t+1}\left(S_{t+1}\right) | S_t, x_t, W_{t+1}\right\}$). The expectation of the 'future' costs is based on the probability distribution for the arrival of new patients and the transitions of all treated patients in the decision vector x_t of the current time period.

As a next step, we specifiy the expectation in (15). We introduce the vector w, consisting of elements w_j representing the number of patients leaving queue j, for all j in $\mathcal{J}$ and arriving from outside the system (w_0). To administer all possible transitions, we introduce the elements, w_{ij} representing a realization for the number of patients that are transferred from queue i to queue j after service at queue i. w_{0j} represents the realization of the number of external arrivals at queue j. w_{j0} represents the number of patients leaving the hospital after treatment at queue j. In addition, we introduce the vector w', which represents a realization of the number of patients arriving at each queue. The element w'_j represents the number of patients arriving at queue j.

Note that under Assumption 1, the transition process follows a multinomial distribution with the parameters $q_{j,i}$ with $i, j \in \mathcal{J}$, and $q_{j,0} = 1 - \sum_{i \in \mathcal{J}} q_{j,i}$ for patients leaving the hospital. Enumerating the product of the probability and value associated with all possible outcomes of w', establishes the expectation in (15):

$$V_t\left(S_t\right) = \min_{x_t \in \mathcal{X}(S_t)} \left(C_t\left(S_t, x_t\right) + \sum_{w'} P(w'|x_t) V_{t+1}\left(S_{t+1} | S_t, x_t, w'\right)\right),$$

and from [2], we obtain

$$P(w'|x_t) = \sum_{w_0=0}^{\infty} P(w_0) \times \sum_{\left\{\begin{array}{c} w_{ij}, i=0,...,|\mathcal{J}|, j=0,...,|\mathcal{J}| : \\ w_{ij} \geq 0, w_{ij} = 0 \text{ if } p_{i,j} = 0, w_{00} = 0, \\ w_j = \sum_{u \in \mathcal{U}} x_{t,j,u}, j = 1,...,|\mathcal{J}|, \\ \sum_{j=0}^{|\mathcal{J}|} w_{ij} = w_j, i = 0,...,|\mathcal{J}|, \\ \sum_{i=0}^{|\mathcal{J}|} w_{ij} = w'_j, j = 0,...,|\mathcal{J}| \end{array}\right\}} \prod_{i=0}^{|\mathcal{J}|} \binom{w_i}{w_{i0},...,w_{i|\mathcal{J}|}} \prod_{j=0}^{|\mathcal{J}|} p_{i,j}^{w_{ij}}.$$

With Assumption 2, we obtain

$$P(w_0) = \frac{\lambda_{0,t}^{w_0}}{w_0!} e^{-\lambda_{0,t}}, \quad \text{with } \lambda_{0,t} = \sum_{j=1}^{|\mathcal{J}|} \lambda_{j,t},$$

and

$$p_{i,j} = \begin{cases} q_{i,j}, & \text{when } i = 1, ..., |\mathcal{J}|, j = 0, ..., |\mathcal{J}|, \\ \frac{\lambda_{j,t}}{\lambda_{0,t}}, & \text{when } i = 0, j = 1, ..., |\mathcal{J}|, \\ 0, & \text{when } i = 0, j = 0. \end{cases}$$

Using existing techniques, such as value iteration and backward dynamic programming, (15) can be solved to optimality. Proof of the existence of optimal solutions is given in [17].

The exact DP solution method can be used to calculate small instances. These instances particularly do not reflect the complexity and size of a real-life sized instance in a hospital. Computing the exact DP solution is generally difficult and possibly intractable for large problems due to three reasons: (i) The state space $S(t)$ for the problem may be too large to evaluate the value function $V_t(S_t)$ for all states within reasonable time, (ii) the decision space $\mathcal{X}(S_t)$ may be too large to find a good decision for all states within reasonable time, and (iii) computing the expectation of 'future' costs may be intractable when the outcome space is large. The outcome space is the set of possible states in time period $t+1$, given the state and decision in time period t. Its size is driven by the random information on the transitions of patients between queues and the external arrivals.

Using dynamic programming to solve real-life size instances of the tactical planning problem seems intractable. To illustrate this, suppose that $\hat{M}$ gives the max number of patients per queue and per number of time periods waiting. The number of states is then given by

$$\hat{M}^{(|\mathcal{J}|\cdot|\mathcal{U}|)}.$$

Consider a system of 8 care processes with an average of 5 stages, resulting in 40 queues, and a maximum number of time periods waiting set to 4. For such a system, the resulting number of states is $\hat{M}^{40\cdot 4} = S_{\max}^{160}$, which is intractable for DP for any $\hat{M} > 1$. Note that in a practical instance at a hospital, $\hat{M}$ may be very large, e.g., $\hat{M} \geq 20$. Additional complexity is added by a large decision space. Assume that in the same system, there is resource capacity available to treat 30 patients in total in one time period. If we assume that they can be treated at 40 queues in the system and that all available resource capacity must be used, this is the same as dividing 30 items over 40 bins. Hence, there are already $\binom{40+32-1}{40-1} = \binom{71}{39} = 1.6 \cdot 10^{20}$ different decisions to evaluate when we are only looking at all decisions that use up maximum resource capacity. In addition, the outcome space may be large, caused by the large number of possible outcomes for the stochastic processes of patient transitions between queues and patient arrivals at each queue.

4 Approximate Dynamic Programming

Various alternatives exist to overcome the intractability problems with DP like mentioned in Section 3. The problem size can for example be reduced by aggregating information on resource capacities, patients, and/or time periods. We propose an innovative solution approach within the frameworks of ADP and mathematical programming, which can be used to overcome all three mentioned reasons for intractability of DP for large instances. Our solution approach is based on value iteration with an approximation for the value functions. In this section, we explain this approach in more detail.

First, we discuss the use of a 'post-decision' state as a single approximation for the outcome state. Second, we introduce the method to approximate the value of a state and decision, and third, we explain how we use a 'basis functions' approach in the algorithm to approximate that value. This combination uses an approximation for the expectation of the outcome space, thereby reducing complexity significantly. It also enables calculating the value state by state, making the necessity to calculate the entire state space at once, which was the primary reason of intractability of the exact DP approach, obsolete. Fourth, we explain how we overcome the large decision space for large problem instances with an ILP.

4.1 Post-decision state

To avoid the problem of a large outcome space and the intractable calculation of the expectation of the 'future' costs, we use the concept of a post-decision state S_t^x [16]. The post-decision state is the state that is reached, directly after a decision has been made, but before any new information W_t has arrived. It is used as a single representation for all the different states the system can be in the following time period, and it is based on the current pre-decision state S_t and the decision x_t. This simplifies the calculation or approximation of the 'future' costs.

The stochastic transitions and external arrivals, captured in W_{t+1}, follow after the post-decision state S_t^x in time period t and before the pre-decision state S_{t+1} of time period $t+1$. The transitions take place as follows. In addition to the transition function (9), which gives the transition from pre-decision state S_t to pre-decision state S_{t+1}, we introduce a transition function $S^{M,x}(S_t, x_t)$, which gives the transition from the pre-decision state S_t to the post-decision state S_t^x. This function is given by:

$$S_t^x = S^{M,x}(S_t, x_t), \tag{16}$$

with

$$S^x_{t,j,0} = \sum_{i \in \mathcal{J}} \sum_{u \in \mathcal{U}} q_{i,j} x_{t,i,u} \qquad \forall j \in \mathcal{J}, t \in \mathcal{T} \tag{17}$$

$$S^x_{t,j,U} = \sum_{u=U-1}^{U} (S_{t,j,u} - x_{t,j,u}) \qquad \forall j \in \mathcal{J}, t \in \mathcal{T} \tag{18}$$

$$S^x_{t,j,u} = S_{t,j,u-1} - x_{t,j,u-1} \qquad \forall j \in \mathcal{J}, t \in \mathcal{T}, u \in \mathcal{U} \backslash \{0, U\}. \tag{19}$$

The above constraints are in line with the stochastic transitions between two states in (9) to (12). The transition in (16) to (19) is based on the path that patients follow after treatment. There are two differences with the ILP formulation in (2) to (8). The post-decision state is in the same time-period t as the pre-decision state, and the external arrivals to the system are not included in this formulation, as they are not a result of the decision that is taken. Note that the post-decision state is a direct image of the pre-decision state S_t and the decision x_t.

The actual realizations of new patient arrivals and patient transitions will occur in the transition from the post-decision state in the current time period to the pre-decision state in the next time period. Note that (16) can be adapted to include pre-defined priority rules like always treating patients with longest waiting times before selecting others within the same queue. This rule is used in our computational experiments as well. For the remainder of this chapter, whenever we use the word 'state', we are referring to the pre-decision state.

We rewrite the DP formulation in (15) as

$$V_t(S_t) = \min_{x_t \in \mathcal{X}_t(S_t)} \left(C_t(S_t, x_t) + V^x_t(S^x_t)\right),$$

where the value function of the post-decision state is given by

$$V^x_t(S^x_t) = \mathbb{E}\left\{V_{t+1}(S_{t+1}) \,|S^x_t\right\}. \tag{20}$$

To reduce the outcome space for a particular state and decision, we replace the value function for the 'future costs' of the post-decision state $V^x_t(S^x_t)$ with an approximation based on the post-decision state. We denote this approximation by $\overline{V}^n_t(S^x_t)$, which we are going to learn iteratively, with n being the iteration counter.

We now have to solve

$$\tilde{x}^n_t = \arg \min_{x_t \in \mathcal{X}_t(S_t)} \left(C_t(S_t, x_t) + \overline{V}^{n-1}_t(S^x_t)\right), \tag{21}$$

which gives us the decision that minimizes the value $\widehat{v}^n_t$ for state S_t in the n-th iteration. The function $\widehat{v}^n_t$ is given by

$$\widehat{v}^n_t = \min_{x_t \in \mathcal{X}_t(S_t)} \left(C_t(S_t, x_t) + \overline{V}^{n-1}_t(S^x_t)\right). \tag{22}$$

Note that $V^{n-1}_t(S^x_t) = 0$ is equivalent to having a standard myopic strategy where the impact of decisions on the future is ignored.

After making the decision $\tilde{x}_t^n$ and finding an approximation for the value in time period t (denoted by $\widehat{v}_t^n$), the value function approximation $\overline{V}_{t-1}^{n-1}\left(S_{t-1}^x\right)$ can be updated. We denote this by

$$\overline{V}_{t-1}^{n}\left(S_{t-1}^x\right) \longleftarrow U\left(\overline{V}_{t-1}^{n-1}\left(S_{t-1}^x\right), S_{t-1}^x, \widehat{v}_t^n\right). \tag{23}$$

In (23), we update the value function approximation for time period $t-1$ in the n-th iteration with the 'future' cost approximation for time period $t-1$ in the $n-1$-th iteration, the post-state of time period $t-1$, and the value approximation for time period t. The objective is to minimize the difference between the 'future' cost approximation for time period $t-1$ and the approximation $\widehat{v}_t^n$ for time period t with the updating function, as n increases. This is done by using the algorithm presented in the following section.

4.2 The ADP algorithm

We solve (21) recursively. Starting with a set of value function approximations and an initial state vector in each iteration, we sequentially solve a subproblem for each $t \in \mathcal{T}$, using sample realizations of W_t, which makes it a Monte Carlo simulation. In each iteration, we update and improve the approximation of 'future' costs with (23). Consecutively, the subproblems are solved using the updated value function approximations in the next iteration. This is presented in Algorithm 4.

Algorithm 4 *The Approximate Dynamic Programming algorithm*

Step 0. Initialization

- Step 0a. Choose an initial approximation $\overline{V}_t^0\left(S_t\right)$ for all $t \in \mathcal{T}$ and S_t.
- Step 0b. Set the iteration counter, $n = 1$, and set the maximum number of iterations N.
- Step 0c. Set the initial state to S_1.

Step 1. Do for $t = 1, ..., |\mathcal{T}|$:

- Step 1a. Solve (21) to get $\tilde{x}_t$.
- Step 1b. If $t > 1$, then update the approximation $\overline{V}_{t-1}^{n}\left(S_{t-1}^x\right)$ for the previous post-decision S_{t-1}^x state using

$$\overline{V}_{t-1}^{n}\left(S_{t-1}^x\right) \longleftarrow U^V\left(\overline{V}_{t-1}^{n-1}\left(S_{t-1}^x\right), S_{t-1}^x, \widehat{v}_t^n\right)$$

  where $\widehat{v}_t^n$ is the resulting value of solving (22).
- Step 1c. Find the post-decision state S_t^x with (16) to (19).
- Step 1d. Obtain a sample realization W_{t+1} and compute the new pre-decision state with (9).

Step 2. Increment n. If $n \leq N$ go to Step 1.

Step 3. Return $\overline{V}_t^N\left(S_t^x\right), \forall t \in \mathcal{T}$.

Using the approximation $\overline{V}_t^N(S_t^x)$, for all $t \in \mathcal{T}$, we can approximate the value of a post-decision state for each time period. With these approximations, we can find the best decision for each time period and each state, and thus develop a tactical resource capacity and patient admission plan for any given state in any given time period. The difference with the exact DP approach is not only that we now use an value function approximation for the 'future costs', but also that we do not have to calculate the values for the entire state space.

The current set-up of the ADP algorithm is single pass. This means that at each step forward in time in the algorithm, the value function approximations are updated. As the algorithm steps forward in time, it may take many iterations, before the costs incurred in later time periods are correctly transferred to the earlier time periods. To overcome this, the ADP algorithm can also be used with a double pass approach [16], where the algorithm first simulates observations and computes decisions for *all* time periods in one iteration, before updating the value function approximations. This may lead to a faster convergence of the ADP algorithm. We test the use of double pass versus single pass in Section 6. More details on double pass can be found in [16].

4.3 Basis function approach

The main challenge is to design a proper approximation for the 'future' costs $\overline{V}_t^n(S_t^x)$ that is computationally tractable and provides a good approximation of the actual value to be able to find a suitable solution for the optimization problem of (21). There are various strategies available. A general approximation strategy that works well when the state space and outcome space are large, which generally will be the case in our formulated problem as discussed earlier in this section, is the use of basis functions. We explain the strategy in more detail below.

An underlying assumption in using basis functions is that particular features of a state vector can be identified, that have a significant impact on the value function. Basis functions are then created for each individual feature that reflect the impact of the feature on the value function. For example, we could use the total number of patients waiting in a queue and the waiting time of the longest waiting patient as two features to convert a post-state description to an approximation of the 'future' costs. We introduce

$$\begin{aligned} \mathcal{F} &= \text{set of features,} \\ \phi_f(S_t) &= \text{basis function for the feature } f \in \mathcal{F} \text{ for the state } S_t. \end{aligned}$$

We now define the value function approximations as

$$\overline{V}_t^n(S_t^x) = \sum_{f \in \mathcal{F}} \theta_f^n \phi_f(S_t^x), \qquad \forall t \in \mathcal{T}, \tag{24}$$

where θ_f^n is a weight for each feature $f \in \mathcal{F}$, and $\phi_f(S_t^x)$ is the value of the particular feature $f \in \mathcal{F}$ given the post-decision state S_t^x. The weight θ_f^n is updated recursively and the iteration counter is indicated with n. Note that (24) is a linear approximation, as it is linear in its parameters. The basis functions themselves can be nonlinear [16].

Features are chosen that are independently separable. In other words, each basis function is independent of the other basis functions. For our application, we make the assumption that the properties of each queue are independent from the properties of the other queues, so that we can define basis functions for each individual queue that describe important properties of that queue. Example features and basis functions are given in Table 3, and we will discuss our selection of basis functions, based on a regression analysis, in Section 6.

In each iteration, the value function approximations are updated, as given in (23). In the features and basis functions approach, this occurs through the recursive updating of θ_f^n. Several methods are available to update θ_f^n after each iteration. An effective approach is the recursive least squares method, which is a technique to compute the solution to a linear least squares problem [16]. Two types of recursive least squares methods are available. The least squares method for *nonstationary* data provides the opportunity to put increased weight on more recent observations, whereas the least squares method for *stationary* data puts equal weight on each observation.

The method for updating the value function approximations with the recursive least squares method for nonstationary data follows from [16] and is given in Appendix 8.2. In this method, the parameter α^n determines the weight on prior observations of the value. Setting α^n equal to 1 for each n would set equal weight on each observation, and implies that the least squares method for stationary data is being used. Setting α^n to values between 0 and 1 decreases the weight on prior observations (lower α^n means lower weight). We define the parameter α^n by

$$\alpha^n = \begin{cases} 1 & \text{, stationary} \\ 1 - \frac{\delta}{n} & \text{, nonstationary} \end{cases} \text{, where } n = 1, 2, ..., N. \tag{25}$$

where $1 - \frac{\delta}{n}$ is a function to determine α_n that works well in our experiments. We come back to setting α^n (and δ) in Section 6.1.

4.4 ILP to find a decision for large instances

In small, toysized problem instances, enumeration of the decision space to find the solution to (21) is possible. For real-life sized problem instances, this may become intractable, as explained in Section 3. In this case, we require an alternative strategy to enumeration. In case the basis functions are chosen to be linear with regards to the decision being made (or the resulting post-state description), we can apply ILP to solve (21). The ILP formulation is given in Appendix 8.1, and will be used in Section 6.3.

This concludes our theoretical explanation of our solution approach incorporating ADP and ILP. We have formulated an algorithm, an approximation approach involving features to estimate the 'future' costs, a method to update the approximation functions based on new observations, and an ILP formulation to determine the decisions. In the next section, we explain how these methods can be used to develop tactical plans.

5 Managerial implications

In the previous section, we developed the ADP algorithm to find the feature weights for the value function approximation. In this section we will explain how this ADP approach can be used to establish the tactical plans.

The ADP approach can be used to establish long-term tactical plans (e.g., three month periods) for real-life instances in two steps. First, N iterations of the ADP algorithm have to be performed to establish the feature weights for each time period $t \in \mathcal{T}$. Implicitly, by determining these feature weights, we obtain and store the value functions as given by (22) and (24) for each time period. Second, these value functions can be used to determine the tactical planning decision for each state and time period by enumeration of the decision space or the ILP as introduced in Section 4.4. In the next paragraph, we explain this in more detail.

For each time period in the time horizon, the value function approximations from the ADP approach are used to establish a tactical planning decision. State transitions are calculated by using the state in the current time period, the decision calculated with the value function approximations, the the expected number of patient arrivals and patient transfers between the queues. Subsequently, the value function approximations are used to determine the tactical planning decision for the new state in the following time period. This is repeated for all successive time periods until the end of the time horizon. The procedure may result in noninteger values for the post-state description, due to the patient transfer probabilities. To implement a tactical planning decision, it requires integer values for the number of patients to be served from each queue. While the ADP-model can contain noninteger values for each entry ($u \in \mathcal{U}$) in the waiting lists and the tactical planning decision, the integer restriction is on the *total* number of patients to be served from each queue (summed over all $u \in \mathcal{U}$ for one queue), like explained in [10]. In case only very few patients are included in the system, causing many queues to have 0 or 1 patient, developing a rounding procedure can be beneficial to ensure integer transfers between queues to obtain integer post-decisions states.

The actual tactical plan is implemented using a rolling horizon approach, in which for example tactical plans are developed for three consecutive months, but only the first month is actually implemented and new tactical plans are developed after this month. The rolling horizon approach is recommended for two reasons. First, the finite horizon approach, apart from the benefits it provides to model time dependent resource capacities and patient demand

for example, may cause unwanted and short-term focused behavior in the last time periods. Second, recalculation of tactical plans after several time periods have passed, ensures that the most recent information on actual waiting lists, patient arrivals, and resource capacities is used. As time progresses during the execution of a tactical plan, more information becomes available on the actual realized number of patient arrivals and patient transfers between queues. This information can be used to align the tactical plan with the actual state of the system. The updated tactical planning decision can be calculated with the existing value function approximations and the actual state of the system. If resource requirements, resource capacities, arrival probabilities, or transfer probabilities change, the value functions have to be recalculated using the ADP algorithm.

In the following section, we will determine the features and various other settings for the ADP algorithm, and discuss the algorithm's performance for small and large instances.

6 Computational results

In this section, we test the ADP algorithm developed in Section 4. One of the methods prescribed by [16] is to compare the values found with the ADP algorithm with the values that result from the exact DP solution for small instances. We will use this method in Sections 6.1 and 6.2. In Section 6.3, we study the performance for large instances, where we compare the ADP algorithm with 'greedy' planning approaches to illustrate its performance. We first discuss setting for the ADP algorithm in Section 6.1.

6.1 Settings for the ADP algorithm

In this section, we use information from the DP solution to set the basis functions and the parameters for the ADP algorithm. The DP recursions and the ADP algorithm are programmed in Delphi, and for the computational experiments we use a computer with an Intel Core Duo 2.00 GHz processor and 2GB RAM.

First, we explain the parameters used for the problem instance to calculate the exact DP solution. Second, we explain the selected basis functions, and third, we explain general settings for the ADP algorithm.

6.1.1 Parameter settings

Some settings in the ADP algorithm, such as the basis functions and double pass or single pass, can be analyzed by comparing the results from the ADP approach with the results from the exact DP approach. The values of the DP can be calculated for extremely small instances only, due to the high dimensions in states and the expectation of the future value in the tactical

planning problem. Only for these small instances, we have the opportunity to compare the ADP approximation with the exact DP values. We do not compare the calculated decision policies from both methods, but compare the obtained values. This comparison provides a clear evaluation of the quality of the approximation in the ADP approach for small instances. Since we use exactly the same ADP algorithm for small and real-life sized large instances, this also provides a strong indication of the quality of the approximation accuracy of the ADP approach for large instances (for which we cannot calculate the exact DP value).

For our experiments with small problems in this section, we use the following instance. The routing probabilities $q_{i,j}$ are: $q_{1,2} = 0.8, q_{2,3} = 0.8, q_{1,1} = q_{2,1} = q_{2,2} = q_{3,1} = q_{3,2} = q_{3,3} = 0$. Hence, a patient that is served at Queues 1 or 2 exits the system with probability 0.2, and a patient that is served at Queue 3 will always exit the system. Since there are 3 queues and there are 2 periods that a patient can wait: 0 and 1 time period, the state description for a time period t becomes:$[S_{t,1,0}, S_{t,1,1}, S_{t,2,0}, S_{t,2,1}, S_{t,3,0}, S_{t,3,1}]$. The exact DP-problem is restricted by limiting the number of patients that can be waiting in each queue to 7. The state holding the most patients is thus $[7, 7, 7, 7, 7, 7]$. If there are transitions or new arrivals that result in a number greater than 7 for a particular queue and waiting time, the number for that particular entry is set to 7. So if, after transitions, we obtain a State $[3, 1, 6, 8, 5, 4]$, this state is truncated to $[3, 1, 6, 7, 5, 4]$. In the same way, the states in the ADP are also restricted for comparison, even though this is not necessary. For large instances, when comparison with an exact DP solution is impossible, this state truncation method is not used. The state truncation may affect the ADP-approximation slightly, as it introduces nonlinearity around the edges of the state space. Using the number of time periods, the truncated state space, the number of queues, and the maximum number of time periods waiting, there are $8 \cdot 8^{(3 \cdot 2)} = 2{,}097{,}152$ entries to be calculated. The weights θ^n in the value function approximations are initialized to $\theta^0 = 1$ for all time periods, and the matrix $B^0 = \epsilon I$ as explained in Section 8.2. All other parameters are given in Table 2.

6.1.2 Selection of basis functions

In Section 4, we introduced basis functions to approximate the future value of a particular decision in a particular state. Basis functions are used because of their relative simplicity. The selection of the features however, requires careful design. The challenge in this careful design is to make sure the choice of basis functions actually contributes to the quality of the solution. The basis functions can be observed as independent variables in the regression literature [16]. Hence, to select a proper set of basis functions that have significant impact on the value function, we use a regression analysis. In the regression analysis, the dependent variables are the computed values in the exact DP approach for the first time period, and the independent variables are the basis functions calculated from the state description.

Parameter	*Description*	*Used values for testing*
T	The number of time periods	$8, \mathcal{T} = \{1, 2, \ldots, 7, 8\}$
R	The number of resource types	1
G	The number of care processes	1
e_g	The number of stages in each care process	$3, \mathcal{J} = \{1, 2, 3\}$
U	The number of periods waiting	$2, \mathcal{U} = \{0, 1\}$
$s_{j,r}$	Expected service time from resource type $r \in \mathcal{R}$ for a patient in queue $j \in \mathcal{J}$ in time units	1
$\eta r, t$	Resource capacity for resource type $r \in \mathcal{R}$ in time $t \in \mathcal{T}$ in time units	6
$\lambda_{1,t}$	Poisson parameter for new demand in the Queue 1 in time period $t \in \mathcal{T}$	5
$C_{t,j,u}$	Costs per patient waiting in a queue $j \in \mathcal{J}$, for $u \in \mathcal{U}$ time periods, in time period $t \in \mathcal{T}$	$\frac{(u+1)}{j}$

Table 2 The parameters that characterize the test instance.

Table 3 shows the regression results on various basis functions. The R^2 depicts the variation in the value that is explained by a regression model that uses the features as mentioned in the table as independent variables. The higher R^2, the better suitable the basis functions are for predicting (and thus approximating) the value. One can observe that the features with high level of detail about the state description score significantly better (are higher in the ordered table). Obviously, in addition to the basis functions in Table 3, a significant number of alternatives are available.

For our ADP-model, we choose to use the features 'The number of patients in queue j that are u periods waiting' from the list in Table 3. These basis function explain a large part of the variance in the computed values with the exact DP approach ($R^2 = 0.954$), and the basis functions can be straightforwardly obtained from the state or post-state description. We choose these functions as they seize the highest level of detail on the state description, and therefore are likely to provide high quality approximations. In case there is no independent constant in the set of predictors $\mathcal{F}$ in a linear regression model, the model is forced to go through the origin (all dependent and independent variables should be zero at that point). This may cause a bias in the predictors. To prevent this bias, we add a constant term as one of the elements in $\mathcal{F}$. The feature weight θ_f^n may vary, but the feature value $\phi_f\left(S_t^x\right)$ of this constant is always 1, independent of the state S_t^x.

6.1.3 Double pass

In Section 4 we introduced the possible use of double pass, where the algorithm first steps through all time periods before updating the value functions. Our experiments confirm that double pass leads to faster convergence of the ADP algorithm than single pass.

Features	**Basis functions**	**# vars**	R^2
The number of patients in queue j that are u periods waiting	$S_{t,j,u}, \forall j \in \mathcal{J}, \forall u \in \mathcal{U}, t = 1$	$\lvert\mathcal{J}\rvert \times \lvert\mathcal{U}\rvert$	0.954
Combination of the total number of patients in queue j and the sum of the number of time periods all patients are waiting in queue j	$\sum_{u=0}^{U} S_{t,j,u}$ and $\sum_{u=0}^{U} u \cdot S_{t,j,u}$, $\forall j \in \mathcal{J}, t = 1$	$2 \times \lvert\mathcal{J}\rvert$	0.954
Combination of the total number of patients in queue j and the longest waiting time currently in queue j	$\sum_{u=0}^{U} S_{t,j,u}$ and $\max_{u \in \mathcal{U}} S_{t,j,u}$, $\forall j \in \mathcal{J}, t = 1$	$2 \times \lvert\mathcal{J}\rvert$	0.954
The total number of patients in queue j	$\sum_{u=0}^{U} S_{t,j,u}$, $\forall j \in \mathcal{J}, t = 1$	$\lvert\mathcal{J}\rvert$	0.950
The sum of the number of time periods all patients are waiting in queue j	$\sum_{u=0}^{U} u \cdot S_{t,j,u}, \forall j \in \mathcal{J}, t = 1$	$\lvert\mathcal{J}\rvert$	0.879
The longest waiting time currently in queue j	$\max_{u \in \mathcal{U}} S_{t,j,u}, \forall j \in \mathcal{J}, t = 1$	$\lvert\mathcal{J}\rvert$	0.199
The average waiting time in queue j	$\frac{\sum_{u=0}^{U} u \cdot S_{t,j,u}}{\sum_{u=0}^{U} S_{t,j,u}}, \forall j \in \mathcal{J}, t = 1$	$\lvert\mathcal{J}\rvert$	0.033

Table 3 The basis functions and their R^2 regression on the given value function. In each regression, a constant is added as a variable. All R^2 values are obtained with significance of 0.000, indicating a good fit of the model. The third column '# vars' indicates the number of variables when the particular basis function is used.

To illustrate the effect, we compare the values from the exact DP solution with the found ADP values for 5000 randomly generated states. The ADP algorithm uses the recursive least squares method for nonstationary data, with $\delta = 0.95$ in (25). To evaluate the speed of the ADP algorithm, we display the number of iterations required until the algorithm is within 5% of the DP value, so either 95% or 105% of the DP value for a particular state. The average number of iterations before the ADP value is within 5% of the DP value for the 5000 states is 1131.0 when double pass is not used, and 100.3 when double pass is used. Hence, double pass is a significantly faster method to get an accurate approximated value. This effect can also be observed in Figure 1 for a single state. Also for other values of δ in (25), we find that the use double pass leads to faster convergence to the DP value. For the remainder of our experiments, we use double pass.

6.1.4 Setting α

The parameter α is set in (25). When $\alpha = 1$ is chosen, the recursive least square method for stationary data is selected, and equal weight is given to each

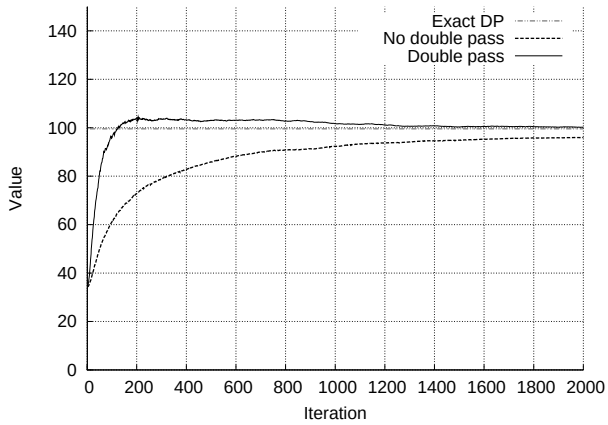

Fig. 1 The values approximated with the ADP algorithm (Settings: recursive least squares for nonstationary data and $\delta = 0.95$) and calculated with the exact DP approach for initial state [2,7,5,1,7,4]. These graphs illustrate the significantly faster convergence when double pass is used.

observation. Because the ADP algorithm is initialized with given arbitrary weights for θ^n and B^n, there is a 'warm-up period' before the weights are properly iterated and getting closer to the actual value. Hence, it seems useful to put less emphasis on the first observations, and more emphasis on later ones. To achieve this the recursive least squares method for nonstationary data is used, as explained in Section 4.3.

To find a good value for δ, we compare the values from the exact DP solution with the found ADP values for 5000 randomly generated states. We compare the number of iterations required until the algorithm is within 5% of the DP value, and the average difference between the ADP value and the DP value ((ADP value - DP value)/DP value) for various settings of δ. Figure 2 shows the results of these experiments. Note that δ cannot be equal to 1, because this would result in $\alpha^1 = 0$ and a division by 0 in (8.2) in the first iteration.

The recursive least squares method for *stationary data* requires 83 runs to reach a value within 5% of the DP value, and has an average difference of 2.2% (standard deviation of 2.7%) after 2500 iterations. The recursive least squares method for *nonstationary data* achieves similar average difference, but in fewer iterations. We explain the results for the recursive least square method for nonstationary data below.

The left side of Figure 2 shows that when δ is closer to 1 (and α^1 is close to 0), the number of iterations required to reach a value within 5% of the DP value is significantly lower. This is due to the structure of (25), where a higher δ causes a lower α, which puts less emphasis on prior observations. In the first iteration, the prior observations are initializations, done by the modeler and independent of the instance or state. Hence, a 'warm-up period' is required to 'forget' the initializations and approximate the actual values.

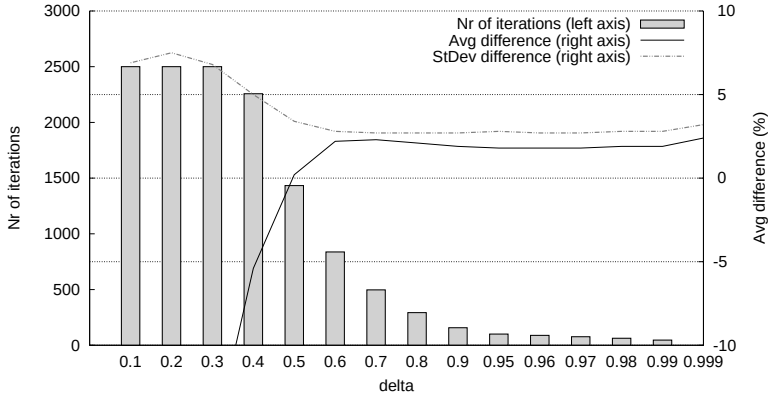

Fig. 2 The average and standard deviation of the difference between the ADP value and the DP value (ADP value - DP value divided by the DP value) are depicted with lines (on the right axis) and the average number of runs required until the algorithm is within 5% of the DP value is given with block diagrams (on left axis). The number of iterations are bounded to 2500 for this experiment.

From the experiments it is clear that setting $\delta \geq 0.6$ decreases the warm-up period significantly, and results in stable performance on the average and standard deviation of the difference. Setting $\delta \leq 0.5$ results in unstability in the matrix operations of the nonstationary least squares method, resulting in strongly decreasing average difference (resulting in longer runtimes required to get to proper results). We obtain even more stringent conclusions if we observe the results for average and standard deviation of the difference: 200 iterations after the ADP algorithm finds values within 5% of the DP value. It appears that $\delta \geq 0.8$ gives the best results. Note that with the division of δ by the iteration number n in (25), α increases fast. After 10 iterations, $\alpha = 0.901$, with $\delta = 0.99$.

From the above it is clear that setting $\delta = 0.99$ results in stable, relatively good performance. For the remainder of our experiments, we use this setting which approximates the DP value within 5% in an average of 46.1 iterations and accurately with an average difference with the DP value of 1.9% and standard deviation of this difference of 2.8% after 2500 iterations for 5000 states.

6.2 Comparison of ADP, DP and greedy approaches for small instances

In this section, the values calculated with the ADP approach are compared with the exact DP solution and two greedy approaches.

6.2.1 Convergence of the ADP algorithm

We have calculated the ADP-algorithm for 5000 random states and found that the values found with the ADP algorithm and the value from the exact DP

solution converge. For these 5000 random states, there is an average deviation between the value approximated with the ADP algorithm and the value calculated with the exact DP approach of 2.51%, with standard deviation 2.90%, after 500 iterations. This means the ADP algorithm finds slightly larger values on average than the exact DP approach. This may be caused by the truncated state space, as explained in Section 6.1.1.

For two initial states, Figure 3 illustrates that the calculated values with the ADP-algorithm (with $\delta = 0.99$ and double pass) converge to the values calculated with the exact DP approach as the number of iterations grow. In the first iterations, the ADP-values may be relatively volatile, due to the low value for α and thus the high impact of a new observation on the approximation. When the number of iterations increases, the weight on prior observations increases as α increases in (25), and the ADP approximations become less volatile.

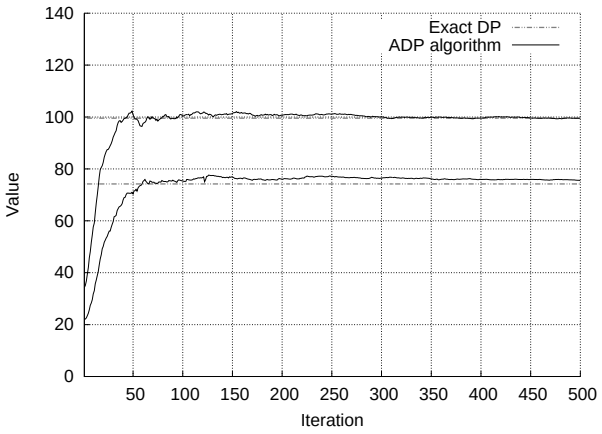

Fig. 3 Example for two initial states. The values approximated with the ADP algorithm (with $\delta = 0.99$ and double pass) converge to the values from the exact DP approach.

The calculation time of the ADP algorithm is significantly lower than the calculation of the exact DP solution. Obtaining the DP solution requires over 120 hours. Calculating the ADP solution for a given initial state (with $N = 500$) takes on average 0.439 seconds, which is 0.0001% of the calculation time for the exact DP solution. Obviously the calculation times depend on the used computation power, but these results indicate that solving a toy problem with the exact DP approach is already very time intensive, and solving such a problem with the ADP approximative approach is significantly faster.

6.2.2 Comparing the use of the feature weights, with DP and two greedy approaches

In the sections above, we have evaluated the performance of the ADP algorithm to find the feature weights. After the ADP algorithm has established the feature weights θ^n that accurately approximate the value associated with a state and a decision, these weights for all time periods are fixed and used to calculate planning decisions for each time period, like explained in Section 5. In this section, we evaluate the accuracy of the ADP approach by comparing the values obtained with the ADP approach, the DP approach, and two greedy approaches.

The two greedy approaches are rules that can be used to calculate a planning decision for a particular state and time period. We call the two approaches 'HighestNumberOfWaitingPatientsFirst' and 'HighestCostsFirst'. In the greedy approach 'HighestNumberOfWaitingPatientsFirst', the queue with the highest number of waiting patients is served until an other queue has the highest number of waiting patients, or until resource capacity constraints do not allow serving another patient of this queue anymore. After that, the next highest queue is served in the same way, until all queues are served and/or resource capacity constraints do not allow serving another patient anymore. In the greedy approach 'HighestCostsFirst', the queue with the highest costs (calculated with the cost function and the state description) is served until an other queue has the highest costs, or until resource capacity constraints do not allow serving another patient of this queue anymore. After that, the next highest queue is served in the same way, until all queues are served and/or resource capacity constraints do not allow serving another patient anymore. The ADP approach and the two greedy approaches can be used to calculate a tactical plan for a complete time horizon, following the steps explained in Section 5.

To compare the value calculated with the four approaches, we calculate a planning decision for each separate time period as follows. As a first step, we generate an initial state for the first time period in time horizon $\mathcal{T}$. We can find the exact DP value associated with this initial state from the already calculated DP solution. To establish the values for the ADP approach and the two greedy approaches, we use simulation as follows. We use the value function approximations from the ADP approach and the described methods from the greedy approaches, to establish a planning decision for the chosen initial state in the first time period. Then simulate the outcomes for patient transfers and patient arrivals. This leads to a particular state in the following time period for which we can establish the planning decision using the ADP approach and greedy approaches. These steps are repeated until the end of the time horizon $\mathcal{T}$. We sum the values associated with each state in each time period in the time horizon $\mathcal{T}$, to obtain the value for the initial state in the first time period. These values are then compared between the different approaches. By following this method, we can properly evaluate and compare the ADP approach in a wide range of possible outcomes for patient transfers

and patient arrivals. When one aims to establish a tactical plan for a complete time horizon upfront, the random patient transfers and patient arrivals are replaced by the expectation for these processes, as explained in Section 5.

We randomly choose a set of 5000 initial states, that we each simulate with 5000 sample paths for the ADP approach and the two greedy approaches. We calculate the relative difference with the DP value for each of the 5000 initial states. Figure 4 displays the average over all initial states. The graph illustrates that the ADP approach provides a relatively accurate approximation for the value of a particular state, and the approximation is significantly better than two greedy approaches. The value resulting from the policy (the value function approximations) obtained with the ADP approach is very close to the values obtained with the optimal policy (found with the exact DP approach). Consequently, the fast and accurate ADP approach is very suitable to determine tactical planning decisions for each time period, and thus to establish a tactical plan for a complete time horizon following the steps explained in Section 5.

These results indicate that the ADP algorithm is suitable for the tactical resource capacity and patient admission planning problem.

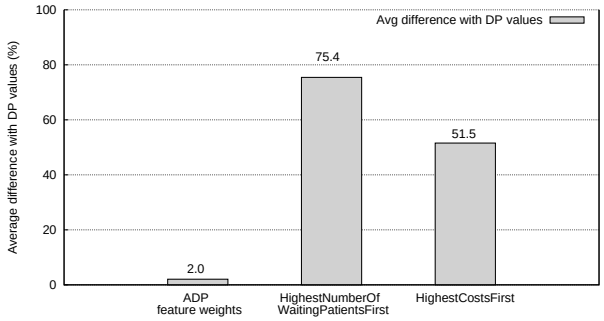

Fig. 4 The average difference with the DP value when using the feature weights from ADP or the two greedy approaches to develop a tactical plan. The average value calculated with the ADP approach is 92.5.

6.3 Performance of the ADP algorithm for large instances

In the previous sections, we analyzed the performance of the ADP algorithm for small, toysized problems to compare the results with the DP approach. In this section, we investigate the performance of the ADP algorithm for large, real-life sized instances. Since for large instances, computation of the exact DP approach is intractable, we evaluate the performance of the ADP algorithm with the two greedy approaches as introduced in Section 6.2.2.

6.3.1 Parameters for large problem instances

As explained in Section 3, for large instances, the decision space becomes too large to allow for complete enumeration. Hence, we use an ILP to compute the optimal decision and it is given in Appendix 8.1. We use a CPLEX 12.2 callable library for Delphi to solve the ILP, and tolerate solutions with an integrality gap of 0.01%.

Parameter	*Description*	*Used values for testing*
$\|\mathcal{T}\|$	The number of time periods	$8, \mathcal{T} = \{1, 2, \ldots, 7, 8\}$
$\|\mathcal{R}\|$	The number of resource types	4
$\|\mathcal{G}\|$	The number of care processes	8
e_g	The number of stages in each care process	$\{3, 5, 7\}, \mathcal{J} = \{1, 2, \ldots, 40\}$
$\|\mathcal{U}\|$	The number of periods waiting	$4, \mathcal{U} = \{0, 1, 2, 3\}$
$s_{j,r}$	Expected service time from resource type $r \in \mathcal{R}$ for a patient in queue $j \in \mathcal{J}$ in time units (four value sets)	$\{10, 15, 20\}, \{30, 45, 60\}$, $\{100, 120, 140\}$, $\{200, 220, 240\}$
$\eta_{r,t}$	Resource capacity for resource type $r \in \mathcal{R}$ in time $t \in \mathcal{T}$ in time units	$\{0, 750, 1000, 1200, 1250,$ $2000, 3600, 5000, 8750,$ $9600, 10000, 17600\}$
$q_{i,j}$	The routing probabilities between queue $i, j \in \mathcal{J}$	$\{0, 0.25, 0.5, 0.75, 1\}$
$\lambda_{1,t}$	Poisson parameter for new demand in the first queue of each care process $g \in \mathcal{G}$ in time period $t \in \mathcal{T}$	$\{1, 3, 5\}$
$C_{t,j,u}$	Costs per patient waiting in a queue $j \in \mathcal{J}$, for $u \in \mathcal{U}$ time periods, in time period $t \in \mathcal{T}$	$\frac{(u+1)}{j}$

Table 4 The parameters that characterize the large test instances.

The parameters to generate the large instances are given in Table 4. When multiple entries are listed, we randomly choose one for each variable. For example, for each initial queue in a care process, we randomly pick the Poisson parameter for new demand from the set 1, 3, or 5. The resource capacities $\eta_{r,t}$ for each resource $r \in \mathcal{R}$ and $t \in \mathcal{T}$ are selected from the given set, this means that we can for example have: $\eta_{1,1} = 1200$, $\eta_{1,2} = 0$, $\eta_{1,3} = 1200$, $\eta_{2,1} = 3600$, $\eta_{2,2} = 3600$, and $\eta_{2,3} = 1200$ can be 0. As real-life instances may have changing patient arrivals and changing resource capacities over time, we vary these parameters over the time periods for each queue and each resource respectively. In contrast with the exact DP approach, truncation of the state space is not required for the ADP algorithm, and we will not truncate the state space in the experiments for large instances. We truncate the initial starting state, to ensure that it is in line with the selected resource capacities and resource requirements. To generate the initial states, we randomly pick the number of patients, for each queue and each number of time periods waiting,

from the set $[0, 1, \ldots, 4]$. This set is bounded to align the initial state with the generated instance for the available resource capacity with the settings in the Table 4.

The weights θ^n in the value functions are initialized to $\theta^0 = 1$ for all time periods, and the matrix $B^0 = \epsilon I$ as explained in Appendix 8.2.

6.3.2 Comparison with greedy approaches

After running the ADP algorithm for $N = 100$ iterations, we fix the established feature weights θ^n, and use these to calculate tactical planning decisions in each time period. In this section, we compare the use of the feature weights calculated in the ADP algorithm with the two greedy approaches introduced in Section 6.2.2: 'HighestNumberOfWaitingPatientsFirst' and 'HighestCostsFirst'. For the comparison, we use the same simulation approach as explained in Section 6.2.2, but for larger instances, we simulate 30 initial states over 10 different generated instances. Similar to Section 6.2.2, we perform 5000 simulation runs per initial state. The relative difference between 'HighestNumberOfWaitingPatientsFirst' and the ADP approach is 50.7%, and the relative difference between 'HighestCostsFirst' and the ADP approach is 29.1%. The average value calculated with the ADP approach is 129.0. The lower average value from the ADP approach indicates that the ADP approach develops tactical plans resulting in lower costs than the two greedy approaches for large instances. The lower values indicate that the ADP approach supports and improves tactical planning decision making, and therefore we can conclude that the ADP approach is a suitable method to calculate a tactical plan for real-life sized instances.

Running the ADP algorithm for a given initial state (with $N = 100$) takes approximately 1 hour and 5 minutes for the large instance. This seems reasonable for finding the feature weights that approximate the value functions for 40 queues and 8 time periods. The feature weights that are calculated for the complete time horizon can be used to adjust the tactical plan in later time periods, as time progresses. Hence, the algorithm does not have to be run on a daily or even weekly basis. Since the algorithm converges fast, one may further decrease the number of iterations, resulting in lower runtimes.

6.3.3 Benefit of considering future costs through the ADP approach

Compared to the greedy approach 'HighestCostsFirst', the ADP approach offers an advantage by also considering costs of the future effects of the evaluated decision. The benefits of this advantage seem especially strong, when parameters such as resource capacity and patient arrivals change over time periods. The finite time horizon in the ADP approach allows for setting time dependent parameters for the problem instance, thereby ensuring that changing parameters over time are incorporated in the decision making. To illustrate the additional benefits of considering future costs in instances where parameters change over time, we have conducted the following experiment. We generated

instances with Table 4, but we limited the number of resource types required for each queue to 1 resource type only. Next, for each resource type separately, we set the resource capacities for each time period to be equal. We obtain the following settings for resource capacities: $\sum_{t=1}^{|\mathcal{T}|} \eta_{1,t} = 6000$, $\sum_{t=1}^{|\mathcal{T}|} \eta_{2,t} = 10000$, $\sum_{t=1}^{|\mathcal{T}|} \eta_{3,t} = 30000$, and $\sum_{t=1}^{|\mathcal{T}|} \eta_{4,t} = 70000$. Leaving these total resource capacities (summed over the full time horizon) for each resource type constant, we set a number of entries for the resource capacities $\eta_{r,t}$ for $r \in \mathcal{R}$ and $t \in \mathcal{T}$ to zero. The total capacity for a resource type is kept constant by increasing the resource capacities for that resource type in time periods where the resource capacities are not set to zero. We compare three scenarios: setting 2, 8 and 14 entries of the total 32 entries for $\eta_{r,t}$ to zero in the complete time horizon $t = 1, \ldots, |\mathcal{T}|$. For the comparison of the three scenarios, we use the same simulation approach as explained in Section 6.2.2, but we simulate 8 initial states. We perform 5000 simulation runs per initial state. In each of the three scenarios, we use the same instances and the same 8 initial states for our calculations and simulation. The results in Figure 5 illustrate that a higher variation of resource capacities in the time horizon, gives a higher benefit of using the ADP approach compared to the 'HighestCostsFirst' approach. These results indicate that the benefit of considering future costs in making a tactical planning decision increases when volatility in resource capacities increases.

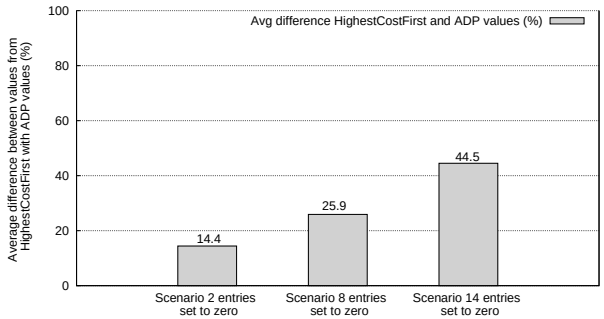

Fig. 5 The average difference between the value calculated by using the feature weights from ADP and the value calculated by using the greedy approach 'HighestCostsFirst'. The average value calculated with the ADP approach is 165.1.

7 Conclusion

We provide a stochastic model for tactical resource capacity and patient admission planning problem in health care. Our model incorporates stochasticity in two key processes in the tactical planning problem, namely the arrival of patients and the sequential path of patients after being served. A Dynamic Programming (DP) approach, which can only be used for extremely small instances, is presented to calculate the exact solution for the tactical planning

problem. We illustrate that the DP approach is intractable for large, real-life sized problem instances. To solve the tactical planning problem for large, real-life sized instances, we developed an approach within the frameworks of Approximate Dynamic Programming (ADP) and Mathematical Programming.

The ADP approach provides robust results for small, toyproblem instances and large, real-life sized instances. When compared with the exact DP approach on small instances, the ADP algorithm provides accurate approximations and is significantly faster. For large, real-life sized instances, we compare the ADP algorithm with the values obtained with two greedy approaches, as the exact DP approach is intractable for these instances. The results indicate that the ADP algorithm performs better than the two greedy approaches, and does so in reasonable run times.

We conclude that ADP is a suitable technique for developing tactical resource capacity and patient admission plans in healthcare. The developed model incorporates the stochastic processes for (emergency) patient arrivals and patient transitions between queues in developing tactical plans. It allows for time dependent parameters to be set for patient arrivals and resource capacity in order to cope with anticipated fluctuations in demand and resource capacity. The ADP model can also be used as for readjusting existing tactical plans, after more detailed information on patient arrivals and resource capacities are available (for example when the number of patient arrivals were much lower than anticipated in the last week). The developed ADP model is generic, where the objective function can be adapted to include particular targets, such as targets for access times, monthly 'production' or resource utilization. Also, the method can be extended with additional constraints and stochastic elements can be added to suit the hospital situation at hand. It can potentially be used in industries outside healthcare. Future research may involve these extensions, and may also focus on further improving the approximation approach, developing tactical planning methods to adjust a tactical plan when it is being performed, or using the ADP approach for other tactical planning objectives.

8 Appendix

8.1 The ILP for large instances

$$\min_{x_t \in \mathcal{X}_t(S_t)} \left(C_t\left(S_t, x_t\right) + \sum_{f \in \mathcal{F}} \theta_f \phi_f\left(S_t^x\right) \right),$$

subject to

$$S_{t,j,0}^x = \sum_{i \in \mathcal{J}} \sum_{u \in \mathcal{U}} q_{i,j} x_{t,i,u} \qquad \forall j \in \mathcal{J}, t \in \mathcal{T}, \tag{26}$$

$$S_{t,j,U}^x = \sum_{u=U-1}^{U} \left(S_{t,j,u} - x_{t,j,u}\right) \qquad \forall j \in \mathcal{J}, t \in \mathcal{T}, \tag{27}$$

$$S_{t,j,u}^x = S_{t,j,u-1} - x_{t,j,u-1} \qquad \forall j \in \mathcal{J}, t \in \mathcal{T}, u \in \mathcal{U} \backslash \{0, U\}, \tag{28}$$

$$x_{t,j,u} \leq S_{t,j,u} \qquad \forall j \in \mathcal{J}, t \in \mathcal{T}, u \in \mathcal{U}, \tag{29}$$

$$\sum_{j \in \mathcal{J}^r} s_{j,r} \sum_{u \in \mathcal{U}} x_{t,j,u} \leq \eta_{r,t} \qquad \forall r \in \mathcal{R}, t \in \mathcal{T}, \tag{30}$$

$$x_{t,j,u} \in \mathbb{Z}_+ \qquad \forall j \in \mathcal{J}, t \in \mathcal{T}, u \in \mathcal{U} \tag{31}$$

Constraints (26) to (28) stipulate that the waiting list variables are consistent. Constraint (29) stipulates that not more patients are served than the number of patients on the waiting list. Constraint (30) assures that the resource capacity of each resource type $r \in \mathcal{R}$ is sufficient to serve all patients, and Constraint (31) is an integrality constraint.

8.2 Updating method based on regressive least squares for nonstationary data

The method for updating the value function approximations with the recursive least squares method for nonstationary data is explained in detail in [16]. The equations used in our solution approach are given below.

The weights θ_f^n, for all $f \in \mathcal{F}$, are updated each iteration (n is the iteration counter) by

$$\theta_f^n = \theta_f^{n-1} - H_n \phi_f\left(S_t^x\right) \left(\overline{V}_{t-1}^{n-1}\left(S_{t-1}^x\right) - \widehat{v}_t^n \right), \qquad \forall f \in \mathcal{F},$$

where H^n is a matrix computed using

$$H^n = \frac{1}{\gamma^n} B^{n-1}.$$

B^{n-1} is an $|\mathcal{F}|$ by $|\mathcal{F}|$ matrix, which is updated recursively using

$$B^n = \frac{1}{\alpha^n} \left(B^{n-1} - \frac{1}{\gamma^n} \left(B^{n-1} \phi\left(S_t^x\right) \left(\phi\left(S_t^x\right)\right)^T B^{n-1} \right) \right).$$

The expression for γ^n is given by

$$\gamma^n = \alpha^n + \phi\left(S_t^x\right)^T B^{n-1} \phi\left(S_t^x\right).$$

B^n is initialized by using $B^0 = \epsilon I$, where I is the identity matrix and ϵ is a small constant. This initialization especially works well when the number of observations is large [16]. The parameter α^n determines the weight on prior observations of the value, and it is discussed in Sections 4 and 6.1.

References

1. R.E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton NJ, USA, 1957.
2. R.J. Boucherie and N.M. van Dijk. Product forms for queueing networks with state-dependent multiple job transitions. *Advances in Applied Probability*, 23(1):152 – 187, 1991.
3. B. Cardoen and E. Demeulemeester. Capacity of clinical pathways - a strategic multi-level evaluation tool. *Journal of Medical Systems*, 32(6):443–452, 2008.
4. A. Erdelyi and H. Topaloglu. Approximate dynamic programming for dynamic capacity allocation with multiple priority levels. *IIE Transactions*, 43(2):129–142, 2010.
5. N.R. Every, J. Hochman, R. Becker, S. Kopecky, and C.P. Cannon. Critical pathways: a review. *Circulation*, 101(4):461–465, 2000.
6. L. Garg, S. McClean, B. Meenan, and P. Millard. A non-homogeneous discrete time Markov model for admission scheduling and resource planning in a cost or capacity constrained healthcare system. *Health Care Management Science*, 13:155–169, 2010.
7. S.C. Graves. A tactical planning model for a job shop. *Operations Research*, 34(4):522–533, 1986.
8. R.W. Hall. *Patient flow: reducing delay in healthcare delivery*. Springer Verlag, 2006.
9. E.W. Hans, M. van Houdenhoven, and P.J.H. Hulshof. A framework for healthcare planning and control. *In: Handbook of Healthcare System Scheduling (Hall, R.W. (editor))*, International Series in Operations Research & Management Science, Vol. 168:303–320, 2012.
10. P.J.H. Hulshof, R.J. Boucherie, E.W. Hans, and J.L. Hurink. Tactical resource allocation and elective patient admission planning in care processes. *Health Care Management Science*, 16(2):152–166, 2013.
11. A.S. Kapadia, S.E. Vineberg, and C.D. Rossi. Predicting course of treatment in a rehabilitation hospital: a Markovian model. *Computers & Operations Research*, 12(5):459–469, 1985.
12. M.S. Maxwell, M. Restrepo, S.G. Henderson, and H. Topaloglu. Approximate dynamic programming for ambulance redeployment. *INFORMS Journal on Computing*, 22(2):266–281, 2010.
13. L.G.N. Nunes, S.V. de Carvalho, and R.C.M. Rodrigues. Markov decision process applied to the control of hospital elective admissions. *Artificial Intelligence in Medicine*, 47(2):159–171, 2009.
14. J. Patrick, M.L. Puterman, and M. Queyranne. Dynamic multipriority patient scheduling for a diagnostic resource. *Operations Research*, 56(6):1507–1525, 2008.
15. M.E. Porter and E.O. Teisberg. How physicians can change the future of health care. *Journal of the American Medical Association*, 297(10):1103, 2007.

16. W.B. Powell. *Approximate Dynamic Programming: solving the curses of dimensionality*. Wiley Series in Probability and Statistics. John Wiley & Sons, 2nd edition, 2011.
17. M.L. Puterman. *Markov decision processes: Discrete stochastic dynamic programming*. Wiley, NY, USA, 1994.
18. V. Schmid. Solving the dynamic ambulance relocation and dispatching problem using approximate dynamic programming. *European Journal of Operational Research*, 219(3):611–621, 2012.
19. H.J. Schütz and R. Kolisch. Approximate dynamic programming for capacity allocation in the service industry. *European Journal of Operational Research*, 218(1):239 – 250, 2012.
20. H. Simao and W.B. Powell. Approximate dynamic programming for management of high-value spare parts. *Journal of Manufacturing Technology Management*, 20(2):147–160, 2009.
21. H. Topaloglu and W.B. Powell. Dynamic-programming approximations for stochastic time-staged integer multicommodity-flow problems. *INFORMS Journal on Computing*, 18(1):31–42, 2006.
22. R.Y.T. Yeung, G.M. Leung, S.M. McGhee, and J.M. Johnston. Waiting time and doctor shopping in a mixed medical economy. *Health Economics*, 13(11):1137–1144, 2004.