Simulation-Based Approximate Policy Iteration with Generalized Logistic Functions

Journal:	*INFORMS Journal on Computing*
Manuscript ID:	JOC-2013-12-OA-238
Manuscript Type:	Original Article
Date Submitted by the Author:	16-Dec-2013
Complete List of Authors:	Sauré, Antoine; Sauder School of Business, University of British Columbia, Patrick, Jonathan; University Of Ottawa, Telfer School of Management Puterman, Martin; Sauder School of Business, University of British Columbia,
Keywords:	Markov decision processes, Approximate dynamic programming, Logistic approximation, Simulation, Patient appointment scheduling

Submitted to *INFORMS Journal on Computing*
manuscript (Please, provide the mansucript number!)

Simulation-Based Approximate Policy Iteration with Generalized Logistic Functions

Antoine Sauré
Sauder School of Business, University of British Columbia, Vancouver, British Columbia, Canada V6T 1Z2
antoine.saure@sauder.ubc.ca

Jonathan Patrick
Telfer School of Management, University of Ottawa, Ottawa, Ontario, Canada K1N 6N5
patrick@telfer.uottawa.ca

Martin L. Puterman
Sauder School of Business, University of British Columbia, Vancouver, British Columbia, Canada V6T 1Z2
martin.puterman@sauder.ubc.ca

We present an approximate dynamic programming method based on simulation, policy iteration, a post-decision state formulation and a logistic value function approximation. This method was developed as part of our efforts to determine whether non-linear value function approximation architectures could provide cost-effective policies for advance patient scheduling problems, and as a way of identifying the main advantages and disadvantages of using simulation versus linear programming to approximately solve dynamic capacity allocation problems. We first apply the proposed method to a queueing problem and then study a more practical application based on an appointment scheduling problem in which patients with different priorities must be assigned service dates prior to realizing future demand. We investigate the quality and practical implications of the resulting patient scheduling policies using simulation, and compare their performance to that of four other policies. Patient scheduling policies obtained by the new method not only depend on the number of appointments already booked on each given day but also on the overall system workload. In particular, these policies provide lower discounted cost values and shorter average wait times for lower priority patients than policies directly obtained using a linear programming approach and an affine value function approximation in the pre-decision state variables.

Key words: Markov decision processes; Approximate dynamic programming; Logistic approximation; Simulation; Patient appointment scheduling.

1. Introduction

Sequential decision-making problems occur in a wide variety of complex systems. The determination of optimal policies for such systems typically requires consideration of an

extremely large number of scenarios and alternative courses of action. Dynamic programming offers a mathematical formalization of the trade-off between immediate and future cost for alternative courses of action for such problems. However, it is well known that for many problems the computational requirements of dynamic programming are overwhelming. In such situations, only suboptimal solutions are available. Solutions for small or simplified problem settings, or alternatively existing decision policies for the same or a similar problem, can be used to gain insights into the characteristics of policies that could work well for the original problem. Unfortunately, there is no guarantee that policies obtained through these approaches will be scalable or effective for the original setting.

Sophisticated methods for dealing with the curse of dimensionality, called Approximate Dynamic Programming (ADP), have been developed in the last couple of decades (Bertsekas and Tsitsiklis 1996, Sutton and Barto 1998, Powell 2011). The literature in this field focuses primarily on simulation-based and linear programming algorithms. Simulation-based algorithms may be classified into two types: policy evaluation and Q-learning methods. Policy evaluation methods concern the approximation of the value function of a single policy and can be embedded within a policy iteration scheme, while Q-learning methods concern the approximation of the optimal value function. Policy evaluation algorithms can be further classified as direct or indirect. Direct methods use simulation to generate value function estimates for different system states and fit a pre-defined approximation architecture to these estimates using a normed error criterion. They allow for non-linear value function approximation architectures and flexibility in the collection of value function samples. Indirect methods such as TD(λ), on the other hand, determine the optimal value function approximation within a specific class of functions, usually a linear combination of basis functions, by solving an approximate form of the optimality equations. Direct and indirect methods differ in speed of convergence and in their suitability for different problems. However, they share a common bottleneck, the slow speed of simulation. Linear programming algorithms are based on formulating the optimality equations associated with a dynamic programming model as a linear program and on choosing an appropriate approximation architecture to represent the variables (value function) in it. This approach was initially proposed by Schweitzer and Seidmann (1985) and has been reconsidered by de Farias and Van Roy (2003, 2004), Adelman and Mersereau (2008) and Desai et al.

(2009). Although linear programming algorithms are mostly limited to affine approximation architectures, their main advantage is that they avoid the need for iterative learning and often provide good results.

In practice, approximating a value function involves two main challenges. First, we need the functional form to be rich enough so that it can provide a good approximation of the value function that we are trying to approximate. This usually requires a good practical or theoretical understanding of the structure of the specific problem at hand. Second, we need to be sure that the approximation architecture does not unnecessarily complicate the solution of the greedy optimization problem on the right-hand side of the optimality equations, or the tuning algorithm for the approximation parameters. This is why most approximation architectures found in the literature are either affine in the state variables or defined as an affine combination of a low-dimensional set of basis functions.

Typically value functions for problems with complex dynamics can be better represented by non-linear functional forms. However, the problem of finding an optimal action at a given state for non-linear approximation architectures involves solving a non-linear mathematical program. In some cases this can be done easily, in others, an additional approximation is needed, introducing a new source of error. Also, the tuning algorithm for the approximation parameters usually entails solving a non-linear least squares problem. Unlike in the linear case, this problem need not have a closed or simple recursive form. Furthermore, there is no guarantee that a non-linear solver will converge to a good solution starting from arbitrary initial values. Finally, though non-linear approximation architectures may provide a better fit to the true value function associated with a specific problem, the approximation parameters will not always have intuitive interpretations.

In this paper, we describe a simulation-based method for approximately solving discounted infinite-horizon Markov Decision Process (MDP) models that uses a post-decision state formulation and in which discounted cost-to-go values are expected to be non-decreasing in the post-decision state variables. The approach, a direct policy evaluation method, corresponds to a least squares approximate policy iteration algorithm in which the discounted cost-to-go function is approximated using a generalized logistic function in the post-decision states variables. It was developed as part of our efforts to identify appropriate value function approximation architectures for advance patient scheduling problems, and

as a way of determining the main advantages and disadvantages of using simulation versus linear programming to approximately solve dynamic capacity allocation problems.

To the best of our knowledge, this is the first time that an explicit non-linear approximation architecture in the approximation parameters and the state variables is used to identify dynamic capacity allocation policies. This is also the first paper to compare and discuss the use of simulation-based and linear programming techniques for the same problem – an advance patient scheduling problem. Patient scheduling policies obtained through the proposed method perform better, in most cases, than other policies for this problem. In particular, they provide lower discounted cost values and shorter average wait times for lower priority patients than policies directly obtained using linear programming and an affine cost-to-go approximation in the pre-decision state variables (Patrick et al. 2008).

The paper is organized as follows. In Section 2, we summarize the literature relevant to our work. In Section 3, we provide a detailed description of the proposed simulation-based algorithm. In Section 4, we present an illustrative controlled queueing example. In Section 5, we study a more practical application based on a multi-priority patient scheduling problem. Finally, in Section 6, we review our main conclusions and suggest possible extensions.

2. Literature

High-dimensional stochastic optimization problems arise in a wide range of operational settings. Usually these problems can be compactly formulated as dynamic programs, however, most practical instances encounter the curse of dimensionality. Fortunately, ADP techniques have proved to be effective in producing useful solution strategies for real-world problems. These techniques have been succesfully applied to several dynamic resource allocation problems, including patient scheduling problems. We refer the reader to Table 1 for a list of ADP applications classified by type of problem and approach.

The main ideas and potential benefits behind the proposed simulation-based algorithm are illustrated later in this paper by using an advance multi-priority patient scheduling problem. This problem involves allocating a fixed amount of daily service capacity, measured in discrete units called "appointment slots", among patients with different urgency levels. It assumes that arriving patients, who require only one appointment slot, can either be scheduled or diverted. Patrick et al. (2008) and Erdelyi and Topaloglu (2009) studied

Table 1 A list of ADP applications classified by type of problem and approach.

Type of Problem	Approach	
	Simulation-based	Linear Programming
Fleet Management	Powell (1987), Frantzeskakis and Powell (1990), Godfrey and Powell (2002), Simão et al. (2009), Simão et al. (2010), Maxwell et al. (2010a,b), Schmid (2012)	
Inventory Management	Van Roy et al. (1997), Lam et al. (2007), Simao and Powell (2009), Enders et al. (2010)	Adelman and Klabjan (2012)
Revenue Management	Gosavi et al. (2002), Hing et al. (2007)	Zhang and Adelman (2009)
Routing	Marbach et al. (2000)	Adelman (2004)
Healthcare Scheduling	Schütz and Kolisch (2012), Astaraky and Patrick (2013)	Patrick et al. (2008), Sauré et al. (2012), Gocgun and Puterman (2013)

this problem in the context of a diagnostic imaging facility and a generic job scheduling problem, respectively. Erdelyi and Topaloglu considered a finite planning horizon and focused on a class of policies characterized by a set of protection levels, whereas Patrick et al. considered an infinite planning horizon and sought optimal scheduling policies using an ADP approach. Patrick et al. modelled the problem as a discounted infinite-horizon MDP and, using linear programming and an affine value function approximation in the pre-decision state variables, were able to identify, under some conditions, an analytical solution for the optimal approximation parameters. Using this solution, they derived an approximate optimal scheduling policy which they evaluated using simulation.

The main drawback of the scheduling policy proposed by Patrick et al. is the sometimes unnecessarily long wait times for lower priority patients, especially if the wait time targets (acceptable waits) are set too far in the future. Unless there is available capacity on day one, lower priority patients are usually booked on their target dates, regardless of the level of system congestion. This is a direct consequence of using an affine value function approximation.

Patrick et al.'s approach fails to take into account that optimal booking actions should depend on the overall workload of the system (and not only on the available capacity on each day independently) and that the marginal discounted cost associated with booking a patient on a given day should increase as the number of available slots on that day decreases. In addition, when some necessary conditions regarding the expected demand over the infinite horizon and the expected number of appointment slots initially filled are violated, the optimal values of the approximation parameters are zero and the optimal

scheduling policy becomes a myopic policy which books patients on a first-come, first-served basis. This, for example, happens when capacity is not a significant limitation.

Next, we present a detailed description of the proposed simulation-based algorithm. As we will see later, patient scheduling policies obtained through this algorithm provide lower discounted cost values and shorter average wait times for lower priority patients than policies directly obtained or derived from the linear programming approach using an affine value function approximation in the pre-decision state variables.

3. Methodology

Let $\vec{s} \in S$ be a state of the system, $\vec{a} \in A_{\vec{s}}$ be an action compatible with state $\vec{s}$, $c(\vec{s},\vec{a})$ be the immediate cost associated with taking action $\vec{a} \in A_{\vec{s}}$ when the system is in state $\vec{s} \in S$, and $p(\vec{s'}|\vec{s},\vec{a})$ be the probability of a transition to state $\vec{s'} \in S$ as a result of choosing action $\vec{a} \in A_{\vec{s}}$ in state $\vec{s} \in S$. We consider the following form of Bellman's optimality equations, where $v(\vec{s})$ represents the λ-discounted cost associated with state $s \in S$.

$$v(\vec{s}) = \min_{\vec{a} \in A_{\vec{s}}} \left\{ c(\vec{s},\vec{a}) + \lambda \sum_{\vec{s'} \in S} p(\vec{s'}|\vec{s},\vec{a}) v(\vec{s'}) \right\} \quad \forall \vec{s} \in S \tag{1}$$

We consider dynamic programs in which the effect of actions and information on the state variables can be separated and thus can be formulated in terms of post-decision states. A post-decision state, denoted by $s^x \in S^x$, represents the state of a system immediately after decisions are made, but prior to the acquisition of new information. This idea was first introduced by Sutton and Barto (1998), who refer to it as the "after-state" variable. The benefits of using post-decision states are mainly computational. It considerably reduces the size of the state space and avoids the need to compute the expectation on the right-hand side of the optimality equation, an often intractable task for high-dimensional models.

Two transition functions are required to model the evolution of the system from pre- to post-decision states and vice versa. We denote them by $F^x : \{(\vec{s},\vec{a}) : \vec{s} \in S, \vec{a} \in A_{\vec{s}}\} \to S^x$ and $F : S^x \to S$, respectively. The relationship between $v^x : S^x \to \mathbb{R}_0^+$, the value of being in a post-decision state, and $v : S \to \mathbb{R}_0^+$, the value function defined in (1), is given by:

$$v(\vec{s}) = \min_{\vec{a} \in A_{\vec{s}}} \{ c(\vec{s},\vec{a}) + \lambda v^x(F^x(\vec{s},\vec{a})) \} \qquad \forall \vec{s} \in S \tag{2}$$

The proposed method, an approximate policy iteration algorithm, exploits the fact that the right-hand side of (2) is a deterministic optimization problem and uses a parametric non-linear approximation to v^x known as a generalized logistic function.

3.1. Approximate Policy Iteration

The general structure of a post-decision state implementation of an approximate policy iteration algorithm (Bertsekas 2010) is the same as that of the classical policy iteration algorithm (Puterman 1994). The algorithm starts with an arbitrary policy d^0 and iteratively generates a sequence of new policies $d^1, d^2, \ldots$ until convergence is achieved. The algorithm, however, assumes that a parametric approximation architecture $\widetilde{v}_{\vec{b}} : S^x \to \mathbb{R}_0^+$ is used to represent the value of being in different post-decision states, where $\vec{b}$ denotes the approximation parameters. In each iteration j, we first simulate the policy d^j, which is obtained from $\widetilde{v}_{\vec{b}^{j-1}}$, by solving (3) (policy evaluation) and then determine a new approximation function $\widetilde{v}_{\vec{b}^j} : S^x \to \mathbb{R}_0^+$ by updating the value of $\vec{b}$ (policy improvement).

$$d^j(\vec{s}) \in \arg\min_{\vec{a} \in A_{\vec{s}}} \left\{ c(\vec{s}, \vec{a}) + \lambda \widetilde{v}_{\vec{b}^{j-1}}(F^x(\vec{s}, \vec{a})) \right\} \qquad \forall \vec{s} \in S \tag{3}$$

The process is repeated until the difference between consecutive sets of approximation parameters is sufficiently small. In practice, an initial stationary policy d^0 is needed to initiate the algorithm. We found that choosing this policy carefully was necessary to ensure convergence. A suitable choice for d^0 may be based on problem structure, on generalizing a policy for a smaller instance of the same problem, or on what is used in practice.

3.2. A Simulation-Based Approximate Policy Iteration Algorithm

We now describe the proposed simulation-based approximate policy iteration algorithm. We assume that a parametric approximation architecture $\widetilde{v}_{\vec{b}} : S^x \to \mathbb{R}_0^+$ is used to represent the value function in terms of post-decision states, where $\vec{b}$ denotes the approximation parameters.

1. *Initialization.* We first use an initial policy d^0 to generate a set of post-decision states and expected discounted cost estimates through the policy evaluation step explained next. We then determine the initial value of the approximation parameters, $\vec{b}^0$, by solving the least squares problem that is part of the policy improvement step explained later.

2. *Policy Evaluation.* First, in each iteration j, the algorithm generates a set $\{\vec{s}_0^{\,xr}, \bar{v}^r\}_{r=1}^R$ of R post-decision states and value function estimates. The value function estimate $\bar{v}^r$ associated with post-decision state $\vec{s}_0^{\,xr}$ is computed as the average discounted cost over K simulation runs starting from $\vec{s}_0^{\,xr}$, where each run simulates the greedy policy associated with $\vec{b}^{j-1}$ for T days. The post-decision state $\vec{s}_0^{\,xr}$ is determined by sampling an initial state at random and simulating the stationary policy d^0 for T_0 decision epochs. The parameter T_0 is called the warm-up period. The greedy policy associated with $\vec{b}^{j-1}$ is given by $d^j(\vec{s}) \in \arg\min_{\vec{a}\in A_{\vec{s}}}\{c(\vec{s},\vec{a}) + \lambda \widetilde{v}_{\vec{b}^{j-1}}(F^x(\vec{s},\vec{a}))\}\ \forall \vec{s} \in S$, where $\widetilde{v}_{\vec{b}^{j-1}}(F^x(\vec{s},\vec{a}))$ represents the value of being in post-decision state $F^x(\vec{s},\vec{a})$, $\vec{a} \in A_{\vec{s}}$. Transition functions $F^x : \{(\vec{s},\vec{a}) : \vec{s} \in S, \vec{a} \in A_{\vec{s}}\} \to S^x$ and $F : S^x \to S$ are used to simulate the evolution of the system from pre- to post-decision states and from post- to pre-decision states, respectively. We choose R to be ten times the number of approximation parameters (a rule of thumb) and K to be the minimum number of replications needed to stabilize the value function estimates associated with a random sample of ten post-decision states. T is chosen so that the tail of the discounted cost after the T-th decision epoch in an infinite trajectory is no larger than ε, which we set arbitrarily small. T_0 is determined as the minimum number of days after which the system reaches steady-state. K and T_0 are determined by simulating d^0 for different parameter values. The configuration parameters are defined in the following order. First, the sample size R. Second, the warm-up period T_0. Third, the length of a simulation run T. Last, the number of replications K.

3. *Policy Improvement.* Next, given a sample $\{\bar{v}^r\}_{r=1}^R$ of R average discounted cost observations produced by the policy evaluation step starting from R different post-decision states $\{\vec{s}_0^{\,xr}\}_{r=1}^R$, the algorithm solves the least squares problem (4) to obtain a new set of parameter values $\vec{b}^*$.

$$\vec{b}^* \in \arg\min_{\vec{b}\geq 0} \left\{ \sum_{r=1}^{R} \left[\bar{v}^r - \widetilde{v}_{\vec{b}}(\vec{s}_0^{\,xr})\right]^2 \right\} \tag{4}$$

The algorithm then uses $\vec{b}^*$ and the set of approximation parameter values from the previous iteration, $\vec{b}^{j-1}$, to compute $\vec{b}^j$ as $\vec{b}^j = (1-\alpha_j) \times \vec{b}^{j-1} + \alpha_j \times \vec{b}^*$, where α_j follows a generalized harmonic stepsize rule. That is $\alpha_j = \bar{\alpha}/(\bar{\alpha} + j - 1)$, $\bar{\alpha} > 0$. We smooth the value of $\vec{b}$ rather than just using $\vec{b}^*$ because of the noise in our estimates.

4. *Stopping Criteria*. The algorithm finds a final value function approximation, and thus a final policy, by alternating policy evaluation and policy improvement steps until the difference between consecutive sets of approximation parameter values is less than or equal to δ percent (in the sup-norm sense), that is $\||\vec{b}^j - \vec{b}^{j-1}|/\vec{b}^{j-1}\|_\infty \leq \delta$, or a maximum number of iterations J is reached. The value of J is defined arbitrarily large ($J > 1,000$).

Next, we explain our choice of a generalized logistic function to approximate the value of being in different post-decision states.

3.3. A Generalized Logistic Value Function Approximation

In an effort to determine whether sophisticated value function approximations could provide cost-effective booking policies for advance patient scheduling problems, we solved a number of very small instances of the MDP model in Patrick et al. (2008). The instances involved two priority classes, booking horizons of at most three days and a daily capacity of no more than four appointment slots. Then, using the optimal solutions, we constructed first and second-order regression models with the pre-decision state components as the independent variables and the discounted cost as the dependent variable. These models produced R^2 values of over 0.9 but showed a poor fit to the data, as indicated by the corresponding residual plots. The results, as we initially conjectured, suggested that a better value function approximation for this problem should be non-linear in the number of bookings on each day of the booking horizon and dependent on the overall workload of the system and not only on the available capacity on each day independently. We also found that affine approximation architectures in the pre-decision state variables under-estimate the value of pre-decision states when the system workload is low or high and over-estimate it for intermediate workload levels.

In a second effort to capture the actual form of the true value function of the MDP model in Patrick et al. (2008), we approximately solved a number of small instances of the model using a linear programming approach. To this end, we first modified Patrick et al.'s MDP formulation to consider the state of each appointment slot individually and then used an affine value function approximation in the pre-decision state variables to approximately solve the corresponding equivalent linear program. The alternative formulation was intended to capture potential non-linearities in the marginal discounted cost associated

with booking a patient on a specific day as a function of the available capacity that day. Unfortunately, the alternative model failed on its purpose as the optimal approximate solution remained the same as the solution obtained from the original ADP formulation. This despite the fact that our initial analysis suggested a clear non-linear relationship between the marginal discounted cost and the pre-decision state components.

Subsequently, we carried out a second regression analysis. Data for this analysis were generated by simulating Patrick et al.'s approximate optimal policy for a small clinic starting from 5,000 different post-decision states. We chose to use post-decision states instead of pre-decision states in anticipation of implementing a simulation-based approach. The value function estimate for each post-decision state (determined as the state of the system after a warm-up period of 1,000 days) was computed as the average discounted cost over 300 1,500-day replications. Using the estimates, we constructed first and second-order regression models with the post-decision state components as the independent variables and the average discounted cost as the dependent variable, obtaining similar results as in the previous regression analysis. We then incorporated additional predictors into the simple linear regression model (e.g., the minimum and maximum daily number of bookings over the booking horizon) and tested several transformations of the post-decision state components and the average discounted cost without much success in terms of fit. After constructing several other models, we discovered that an improved fit to the data was provided by high-order (4th-order and up) polynomial regression models with the average discounted cost as the dependent variable and the total number of bookings over the booking horizon as the only independent variable (R^2 values of 0.99). The main issue with choosing a polynomial model to represent the value function was the extrapolation out of the range of the data. Scheduling policies derived from a polynomial approximation could potentially generate extreme workload levels for which the approximation could be very poor. Furthermore, there would be no guarantee of always having a non-decreasing functional form in the total number of bookings. Fortunately, we observed a common S-shaped pattern across polynomial models. As a result, we were able to identify a function that produced a similar fit to the discounted cost values as high-order polynomial models but with a set of desirable properties. This function is called the logistic function.

The general form of the logistic function is:

$$f(x) = b_0 + \frac{b_1}{1 + e^{(b_2 x - b_3)}} \quad \text{with } b_1 > 0 \tag{5}$$

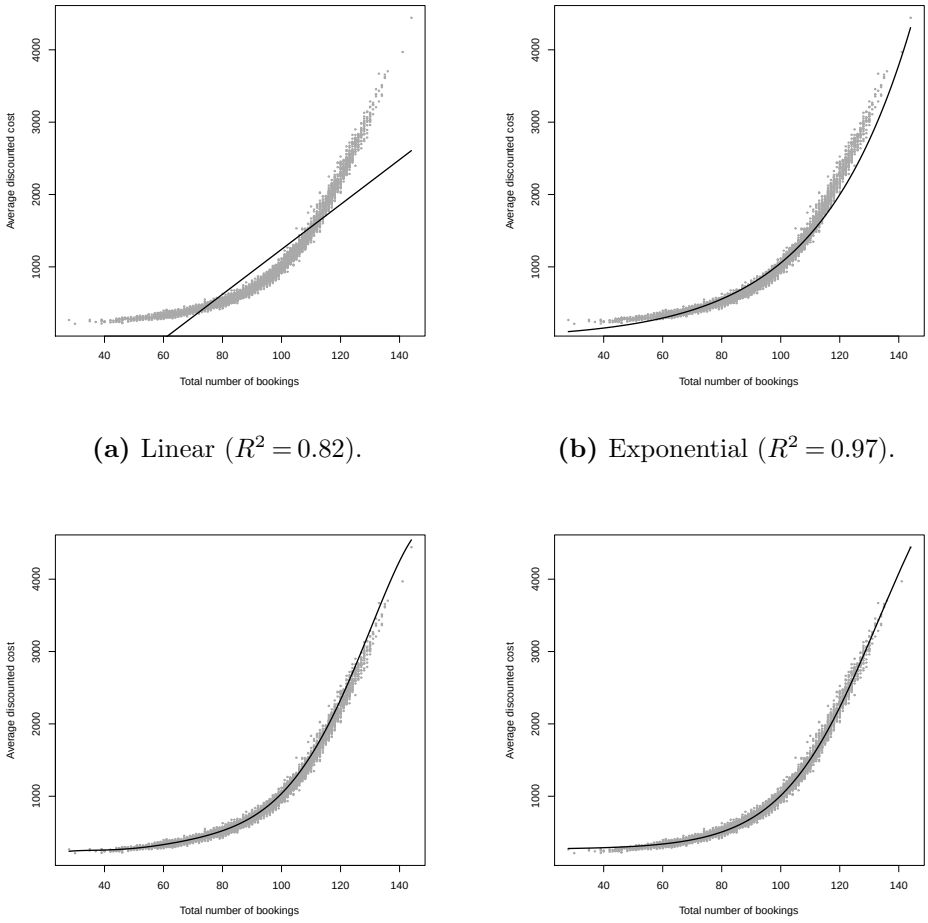
Figure 1 depicts the fit provided by different approximation architectures. Average discounted cost estimates are plotted against the total number of bookings over the booking horizon. Although the R^2 values are the same, visual inspection shows that a generalized logistic function provides a better fit to the data than a 6th-order polynomial model. Figure 1 also shows that an affine model under-estimates the value of post-decision states when the system workload is low or high and over-estimates the value of post-decision states for intermediate workload levels. It also provides the evidence we used to choose the generalized logistic function approximation in the first place.

Plots of the logistic function appear as a stretched S-shape and are increasing or decreasing between two horizontal lines at levels b_0 and $b_0 + b_1$. The value of b_2 determines the spread of the function and its direction. The function is increasing if $b_2 < 0$, decreasing if $b_2 > 0$, and constant if $b_2 = 0$. For small values of x, an increasing logistic function behaves like an increasing exponential function. However, for large values of x the two functions behave quite differently. There is only one change in curvature in the logistic function, at point $(b_3/b_2,\ b_0 + b_1/2)$. An increasing (decreasing) logistic function changes from convex (concave) to concave (convex) at this point. Figure 2 illustrates these properties and highlights the region in which the function is approximately linear. Additionally, when $b_2 = 1$, the middle 50% of a normalized logistic curve (i.e. re-scaled between 0 and 1) spans a little more than two units on the horizontal axis. We will use all these properties later to specify starting values for the approximation parameters in our algorithm.

The proposed algorithm uses a slightly more general version of the logistic function, presented in (6), referred to as a *generalized logistic function*. It considers one approximation parameter for each post-decision state variable s_l^x and thus depends on the utilization levels of different system components. Note that a post-decision state is denoted by a vector $\vec{s}^{\,x} = (s_1^x, \ldots, s_n^x)$, where n represents the number of system components.

$$\widetilde{v}_{\vec{b}}(\vec{s}^{\,x}) = b_0 + \frac{b_1}{1 + \exp\left(-\sum_{l=1}^{n} b_{2l} s_l^x + b_3\right)} \qquad \text{with } \vec{b} \geq 0 \tag{6}$$

Figure 1 **Fit provided by different models to the data generated by simulating Patrick et al.'s policy. Average discounted cost estimates are plotted against the total number of bookings over the booking horizon.**



(a) Linear ($R^2 = 0.82$). **(b)** Exponential ($R^2 = 0.97$).

(c) 6th order polynomial ($R^2 = 0.99$). **(d)** Logistic ($R^2 = 0.99$).

We believe this functional form provides a good approximation architecture for the expected discounted cost in dynamic resource allocation problems in general – and advance scheduling problems in particular – for the following reasons:

- It is non-negative and non-decreasing in the utilization levels of the different system components, and thus suitable for modeling cumulative cost values.
- Not only does it provide a similar fit to data as high-order polynomial models, it also behaves well for values outside of the range of the data.
- It is more flexible in fitting different data patterns (convex, concave and approximately linear) as illustrated in Figure 2.

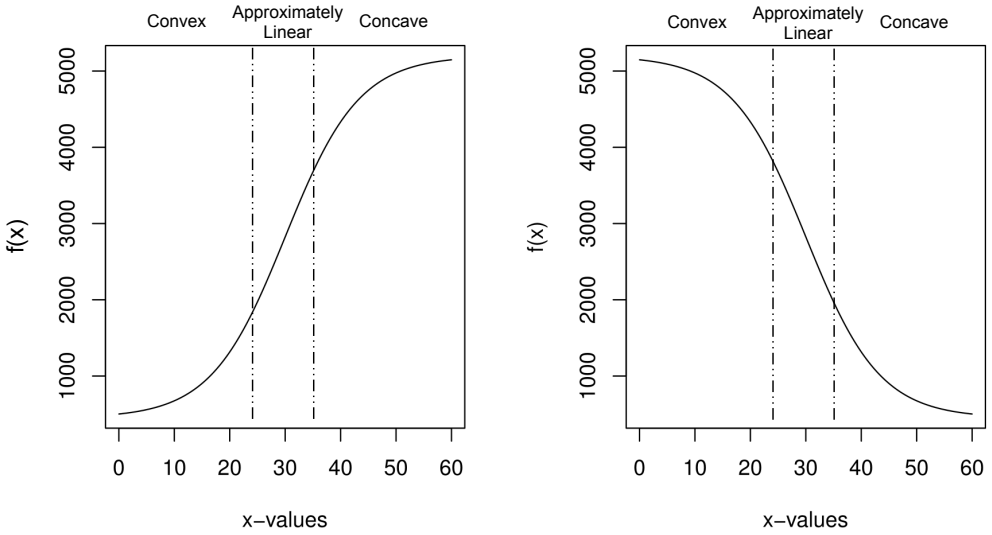


Figure 2 **On the left, a logistic function with parameters $b_0 = 450$, $b_1 = 4,750$, $b_2 = -0.15$ and $b_3 = -4.50$. On the right, a logistic function with parameters $b_0 = 450$, $b_1 = 4,750$, $b_2 = 0.15$ and $b_3 = 4.50$.**

- It makes the marginal discounted cost associated with the level of utilization of a given system component non-constant and dependent on the level of utilization of other components.

The use of a generalized logistic approximation architecture introduces some additional challenges in the context of the approximate policy iteration algorithm above. The solution to the least squares problem in (4), unlike in the linear case, does not have a closed or recursive form. This is because the generalized logistic function is non-linear in some of the approximation parameters. For this reason, we use a non-linear solver to obtain an optimal solution to this problem. Fortunately, the properties of the logistic function allow us to enhance the likelihood of convergence to an optimal solution by specifying good starting values for the approximation parameters in each algorithm iteration. For example, we can initially have $b_0 = \min_r\{\bar{v}^r\}$ and $b_1 = \max_r\{\bar{v}^r\} - b_0$ as the values of b_0 and $b_0 + b_1$ represent the floor and the ceiling of a logistic curve. The initial values of $\vec{b}_2$ and b_3 will depend on the specific problem (see Section 5.1 for an example). In addition, it is known that non-linear optimization algorithms often use large steps as the initial steps in a line search and that non-linear functions are very likely to be called with some variables at their lower or upper bounds. The problem with this is that the exponential function is likely to get an exponentiation overflow error if its argument has no upper bound. Thus, to prevent the non-linear optimization algorithm from moving far away from optimal solutions or into

regions with unreasonable objective or derivative values, we model the argument of the exponential function in (6) as an intermediate variable and restrict its value to be less than or equal to five, as suggested in some commercial non-linear solvers (GAMS 2011). We also set up appropriate scale factors for each approximation parameter in order to further influence the search path taken by the algorithm. Scale factors are defined so that the expected values of the approximation parameters are around one.

The algorithm, with the exception of the least squares problem in (4), was implemented in Java. The least squares problem was coded in GAMS with IPOPT as the non-linear solver. The time the exponential function takes in Java is considerable and actually varies depending on the input value. For this reason, we opted for using a faster implementation of the exponential function based on the following limit:

$$\exp(x) \equiv \lim_{k\to\infty} \left(1+\frac{x}{k}\right)^k \tag{7}$$

We programmed $\exp(x)$ as $(1+x/k)^k$ with $k=256$ because there was no need for more precision. Execution times in this case were on average more than three times faster than when using Java's implementation of $\exp(x)$.

To illustrate the quality of the policies obtained through the proposed simulation-based algorithm, we first study a small queueing example and then present a more practical application based on an advance multi-priority patient scheduling problem.

4. A Queueing Example

We consider the single queueing model with controllable service rate in de Farias and Van Roy (2003). The problem is formulated as a discounted infinite-horizon MDP with state space $S=\{0,1,\ldots,N-1\}$, where $s\in S$ represents a possible number of jobs in the queue. The job arrival probability p is the same for all states. The action $a\in A$ to be chosen in each state $s\in S$ is the service rate, where A is a finite set. We assume $a_{max}=1-p>p$ so that the queue is stabilizable. As a result of choosing service rate $a\in A$ in state $s\in\{1,\ldots,N-2\}$, the next state of the system, denoted by $s'\in S$, is given by:

$$s' = \begin{cases} s-1, \text{ with probability } a; \\ s+1, \text{ with probability } p; \\ s, \quad \text{with probability } 1-p-a. \end{cases} \qquad \forall s\in\{1,\ldots,N-2\}, a\in A \tag{8}$$

From state 0, a transition to state 1 or 0 occurs with probabilities p or $1-p$, respectively. From state $N-1$, a transition to state $N-2$ or $N-1$ occurs with probabilities a or $1-a$, respectively, depending on the choice of $a \in A$. The immediate cost incurred at state $s \in S$ if action $a \in A$ is taken is given by $c(s,a) = s + m(a)$, where $m : A \to \mathbb{R}_0^+$ is an increasing function in $a \in A$.

We assume that jobs arrive with probability $p = 0.2$, service rates are chosen from $A = \{0.2, 0.4, 0.6, 0.8\}$ and that the cost incurred at state $s \in S$ if action $a \in A$ is taken is given by $c(s,a) = s + 60a^3$. The buffer size N is set at 49,999 and the discount factor is 0.98.

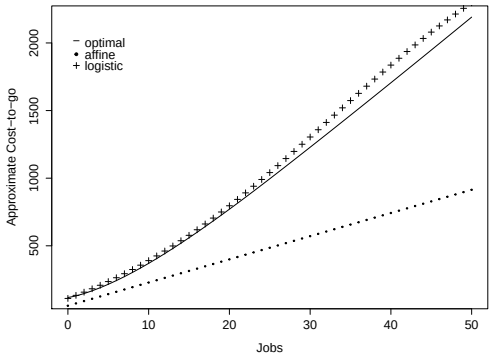
Since pre- and post-decision states are the same for this example, the generalized logistic approximation architecture simply becomes:

$$\widetilde{v}_{\vec{b}}(s) = b_0 + \frac{b_1}{1 + e^{(-b_2 s + b_3)}} \quad \text{with } \vec{b} \geq 0 \tag{9}$$

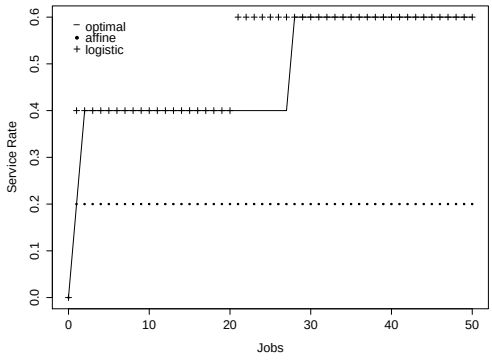
The algorithm required 62 iterations and approximately 15 hours to converge to the final cost-to-go approximaton shown in Figure 3a. The following algorithm configuration was used: number of simulation runs $R = 100$; number of replications in each simulation run $K = 1,500$; warm-up period $T_0 = 0$; number of decision epochs in each simulation run $T = 1,000$ days; stepsize parameter $\bar{\alpha} = 1$; stopping criterion $\delta = 0.001$; maximum number of iterations $J = 500$. The initial policy d^0 was the policy obtained using a linear programming approach and an affine value function approximation in the number of jobs in the queue. Initial states were generated randomly according to state-relevance weights of the form $w(s) = (1-\xi)\xi^s \ \forall s \in S$, with $\xi = 0.9$. The computer used to run the algorithm was a 3.00GHz Quad Core PC with 16GB of RAM.

Figure 3a shows the generalized logistic approximation to the cost-to-go function generated by our algorithm in the range of relevant states (states 0 to 50). It also displays the optimal cost-to-go function and the approximation obtained by using a linear programming approach and an affine approximation architecture in the state variables. Note that given the higher state-relevance weights of small states, the affine approximation estimates their values better than the values associated with large states, making the affine approximation far from optimal. Figure 3b presents the optimal policy and the greedy policy associated with each approximation. We can see from Figure 3b that, unlike the affine approximation, the generalized logistic approximation allow us to identify the optimal action for most

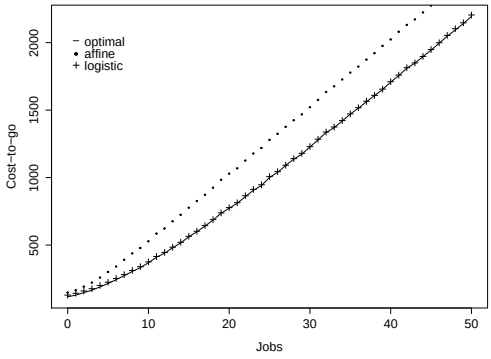
Figure 3 Illustrative single queue example with controlled service rate.



(a) Approximate cost-to-go functions.



(b) Policies corresponding to approximate cost-to-go functions through (3).



(c) Simulated cost-to-go functions.

of the relevant states. Figure 3c shows the cost-to-go function, computed as the average discounted cost over 1,500 1,000-day simulation runs starting from each relevant state, associated with each policy. We can observe that despite taking suboptimal actions for some states, the policy induced by the generalized logistic approximation more closely approximates the optimal cost-to-go function than that generated by the affine approximation for almost all states, and it is close in value to the optimal policy even in states for which it does not take the optimal action. From this example, we conclude that a logistic approximation architecture is rich enough to provide a good approximation of the value function associated with problems with complex dynamics.

5. A Multi-Priority Patient Scheduling Problem

We now consider a health care facility that can perform C fixed-length procedures a day. The capacity of the system is measured in discrete units called "appointment slots". Each

day, the booking agent receives appointment requests from I pre-specified patient types. Patients are classified into types according to their priorities, each priority i having a different medically acceptable wait time, also called target time, denoted by T_i. We assume the number of service requests of type i the agent receives on a given day follows an independent Poisson distribution with positive mean m_i. Patient types and demand distributions do not change over time and demand is assumed uncorrelated over patient types.

Appointment scheduling decisions are made at the end of each day over an infinite horizon and patients can be scheduled at most N days in advance. The cost associated with booking a type i patient on day n (from today) is represented by c_{in}. The agent also has the ability to divert patients at a cost of h per patient.

A pre-decision state of the system is represented by a vector $\vec{s}=(\vec{u},\vec{w})$, where u_n is the number of appointment slots already booked on day n and w_i is the number of patients of type i waiting to be booked. An action available to the agent is represented by a vector $\vec{a}=(\vec{x},\vec{y})$, where x_{in} is the number of patients of type i to book on day n and y_i is the number of patients of type i to divert. The value of $c(\vec{s},\vec{a})$ is given by:

$$c(\vec{s},\vec{a})=\sum_{i=1}^{I}\sum_{n=1}^{N}c_{in}x_{in}+\sum_{i=1}^{I}hy_i \qquad \forall \vec{s}\in S, \vec{a}\in A_{\vec{s}} \tag{10}$$

To make results comparable, we adopt Patrick et al. (2008)'s definition of c_{in} in (11). The parameter g_i in (11) can be thought of as a daily late booking penalty associated with patients of type i. The values of c_{in}, although arbitrary, allow us to model the different wait time targets. The parameter λ is the discount factor.

$$c_{in}=\begin{cases}0, & n\leq T_i;\\ \displaystyle\sum_{k=T_i}^{n}\lambda^{k-T_i-1}g_i, & n>T_i.\end{cases} \qquad \forall i,n \tag{11}$$

For simplicity, we assume that all patients waiting to be booked on a given day must be either scheduled or diverted. In this formulation, there is no upper limit on the number of diversions. Thus, the set of feasible actions in state $(\vec{u},\vec{w})\in S$, denoted by $A_{(\vec{u},\vec{w})}$, is defined by the following two constraints:

$$\sum_{n=1}^{N} x_{in} + y_i = w_i \qquad \forall i \tag{12}$$

$$u_n + \sum_{i=1}^{I} x_{in} \leq C \qquad \forall n \tag{13}$$

Constraint (12) requires the number of bookings and diversions for patients of type i to be equal to the number of patients of that type waiting to be booked. Constraint (13) limits the total number of appointment slots booked for day n to be less than or equal to the available service capacity that day.

Once actions are taken, the only source of uncertainty in the transition to the next pre-decision state of the system is the new demand for service. As a result of choosing booking action $\vec{a} = (\vec{x}, \vec{y})$ in state $\vec{s} = (\vec{u}, \vec{w})$, $\vec{a} \in A_{\vec{s}}$ and $\vec{s} \in S$, and having q_i new service requests from patients of type i, the next pre-decision state of the system, denoted by $\vec{s'} = (\vec{u'}, \vec{w'})$, is determined by the following probability distribution:

$$p(\vec{u'}, \vec{w'} | \vec{u}, \vec{w}, \vec{x}, \vec{y}) = \begin{cases} \prod_{i=1}^{I} \Pr(q_i), \text{ if } \vec{u'} = \left(u_2 + \sum_{i=1}^{I} x_{i2}, \ldots, u_N + \sum_{i=1}^{I} x_{iN}, 0 \right) \\ \qquad\qquad \text{and } \vec{w'} = (q_1, \ldots, q_I)\,; \\ 0, \qquad\qquad \text{otherwise.} \end{cases} \tag{14}$$

In Equation (14), the new number of appointment slots booked on day n is equal to the number of slots previously booked on day $(n+1)$ plus all new bookings that day. Also, the new number of patients of each type waiting to be booked is equal to the number of new requests of each type. The term $\Pr(q_i)$ represents the probability of having q_i new service requests from patients of type i. Note that, as a consequence of considering an N-day booking horizon, the number of appointment slots booked on day N for any valid state of the system is equal to zero.

A post-decision state is represented by a vector $\vec{s}^{\,x} = \vec{u}^{\,x} \in S^x$, where u_n^x is the number of appointment slots booked on day n after taking action $\vec{a}^* = (\vec{x}^*, \vec{y}^*) \in A_{\vec{s}}$ in pre-decision state $\vec{s} = (\vec{u}, \vec{w}) \in S$. To model the transitions from pre- to post-decision states, and vice versa, we define transition functions F^x and F as follows:

$$F^x(\vec{s},\vec{a}) = \vec{u}^{\,x} = \left(u_1 + \sum_{i=1}^{I} x^*_{i1}, \ldots, u_{N-1} + \sum_{i=1}^{I} x^*_{iN-1}, \sum_{i=1}^{I} x^*_{iN} \right) \qquad \forall \vec{s} \in S, \vec{a} \in A_{\vec{s}} \qquad (15)$$

$$F(\vec{s}^{\,x}) = (\vec{u},\vec{w}) = (u^x_2, \ldots, u^x_N, 0, q_1, \ldots, q_I) \qquad \forall \vec{s}^{\,x} \in S^x \qquad (16)$$

Equation (15) defines the next post-decision state as the current appointment schedule plus all new bookings. Equation (16) characterize the next pre-decision state as the schedule at the subsequent decision epoch plus the new demand for service.

The challenge with this MDP model is that even for very small problem settings, the size of the pre-decision state space and the size of the corresponding action sets make the computation of the true value function intractable. The state variable $\vec{s} = (\vec{u},\vec{w})$ and the action variable $\vec{a} = (\vec{x},\vec{y})$ have $(N+I)$ and $(I \times N + I)$ dimensions, respectively. Assuming that w_i can take up to Q_i possible values, this means that we might have up to $(C+1)^N \times \prod_{i=1}^{I} Q_i$ different states and $\prod_{i=1}^{I} Q_i^{(N+1)}$ different, although not necessarily feasible, actions. A small instance with a capacity of six appointment slots per day, three patients classes with $\vec{Q} = (9,6,3)$ and a twelve-day booking horizon involves more than 10^{12} possible pre-decision states and 10^{28} potential actions.

5.1. Implementation of the Proposed Algorithm

The post-decision states from which sample trajectories are generated in an approximate policy evaluation step are obtained by sampling the number of appointment slots occupied on each day of the booking horizon from an integer uniform distribution on the interval $[0, C]$ and simulating the system dynamics under Patrick et al. (2008)'s policy for T_0 days. The warm-up period T_0 is the minimum number of days after which the number of bookings on each day of the booking horizon stabilizes. It is determined by simulating the system under Patrick et al.'s policy for different warm-up periods.The same simulation process is used to determine $\vec{b}^0$, the initial values of the approximation parameters.

Using a generalized logistic function to approximate the discounted cost makes the problem of finding an exact solution to $\min_{\vec{a}\in A_{\vec{s}}}\{c(\vec{s},\vec{a}) + \lambda \widetilde{v}_{\vec{b}^{j-1}}(F^x(\vec{s},\vec{a}))\}$ a non-linear integer program. Although several commercial solvers can be used to find an exact solution to this problem (e.g., IPOPT or MINOS), calling a solver every time that an action needs to be determined requires a considerable amount of time and thus becomes impractical. For this reason, a good approximate solution is preferable. Given a pre-decision state $\vec{s} = (\vec{u},\vec{w}) \in S$,

we first identify the set of patient types for which there are new appointment requests, denoted by $I(\vec{w})$, and the set of days in which there is still available capacity, denoted by $N(\vec{u})$. We then compute the approximate marginal discounted cost associated with each possible individual booking decision with indexes in $I(\vec{w}) \times N(\vec{u})$ and find the type-day combination $(i^*, n^*) \in I(\vec{w}) \times N(\vec{u})$ with the minimum value. If the minimum approximate marginal discounted cost is negative, this heuristic approach books a patient of type i^* on day n^*. If not, it diverts all the remaining demand. After an individual booking decision has been determined, $I(\vec{w})$ and $N(\vec{u})$ are updated. This process is repeated until no patients are waiting. Booking actions obtained through this sequential approach do not show major differences with respect to the optimal solution provided by non-linear solvers. To validate this method, we compared the actions suggested by this procedure with the optimal actions obtained from a non-linear integer programming solver for thousands of different pre-decision states. We found that the booking decisions differed in less than 2% of the cases.

To enhance the likelihood of convergence to an optimal set of parameter values, we specify the following starting values for $\vec{b}$ every time we solve the least squares problem (4). We set $b_0 = \min_r\{\bar{v}^r\}$ and $b_1 = \max_r\{\bar{v}^r\} - b_0$, as explained earlier. We consider the total number of bookings in the system, that is $x = \sum_{n=1}^{N} s^x_{n0}$, and the fact that when normalized the middle 50% of a logistic curve spans a little more than two units on the horizontal axis to set $b_{2n} = 2/\,\mathrm{IQR}(x)\ \forall n$, where $\mathrm{IQR}(x)$ is the interquartile range of x. We also use the fact that b_3/b_2 is the value on the horizontal axis at which a logistic curve reaches its midway point to set $b_3 = b_2 \times \mathrm{median}(x)$. We additionally set up appropriate scale factors for each approximation parameter in order to further influence the search path taken by the algorithm (most non-linear optimization algorithms assume that models are well scaled). The scale factors passed to the non-linear solver are 100, 1000, 0.1 and 1 for b_0, b_1, $\vec{b_2}$ and b_3, respectively.

5.2. Alternative Scheduling Policies

We compare the performance of the appointment scheduling policies suggested by our algorithm, which are greedy with respect to the final "S-Shaped" cost-to-go approximation and therefore called *SS* policies, to that of the four other policies listed below.

- *Myopic policy* (M). Patients are booked as soon as possible, in increasing priority order, according to the immediate cost function defined in (10). This policy resorts to

diversions of type i only when there is no available capacity within the first $\bar{n}_i$ days of the booking horizon, where $\bar{n}_i = \max_n\{n : c_{in} < h\}$. This policy ignores the impact of today's decisions on the future performance of the system.

- *"Day with the Minimum number of Bookings" policy* (DMB). Patients are booked as soon as possible, in increasing priority order, on the day associated with the minimum number of bookings within the corresponding wait time targets. Diversions occur only when there is no available capacity within the targets.

- *"Affine Approximation" policy* (AA). Greedy policy provided by the approximate policy iteration algorithm using the affine approximation architecture below.

$$\widetilde{v}_{\vec{b}}(\vec{u}^x) = b_0 + \sum_{n=1}^{N} b_n u_n^x \qquad \vec{b} \in \mathbb{R}^{N+1} \qquad \forall \vec{u}^x \in S^x \tag{17}$$

With this affine approximation architecture, the least squares problem in (4) reduces to a linear regression for which a closed form solution is available. The approximate optimal action $\vec{a}^* = (\vec{x}^*, \vec{y}^*)$ in state $\vec{s} = (\vec{u}, \vec{w}) \in S$ is given by:

$$(\vec{x}^*, \vec{y}^*) \in \underset{(\vec{x},\vec{y})\in A_{(\vec{u},\vec{w})}}{\arg\min} \left\{ \sum_{i=1}^{I}\sum_{n=1}^{N}(c_{in} + \lambda b_n)x_{in} + \sum_{i=1}^{I} h y_i \right\} \tag{18}$$

Thus, given a final set of parameter values $\{b_n^*\}_{n=0}^N$, the AA policy only books patients on those days for which $(c_{in} + \lambda b_n^* - h) < 0$, diverting any remaining demand.

- *"Patrick, Puterman and Queyranne" policy* (PPQ). Book patients in priority order. Book as much priority 1 demand as possible into the interval $[0, T_1]$, starting with day one and working up to day T_1. For each successive patient priority i, book incoming demand into any available appointment slot within the interval $[1, T_i]$, starting with day one, then day T_i, and working backwards to day two. Divert any remaining demand. This policy corresponds to the booking guidelines derived by Patrick et al. (2008) using the linear programming approach and an affine value function approximation architecture in the pre-decision states variables.

Some observations follow. First, it is important to note that the PPQ policy does not necessarily correspond to the approximate optimal policy suggested by the linear programming approach. In some cases, the optimal values of the affine approximation parameters are zero and the consequent appointment scheduling policies become myopic policies. This is the case for the three instances analyzed in this paper. Second, the affine version of the approximate policy iteration algorithm and the linear programming method developed by Patrick et al. (2008) solve different problems. The former fits an affine approximation architecture to cost-to-go samples using a least squares approach while the latter uses linear programming to solve an approximate form of the optimality equations. Also, the affine version of the algorithm is formulated in terms of post-decision states while the linear programming method developed by Patrick et al. is formulated in terms of pre-decision states. The affine version of the algorithm provides a least upper bound on the discounted cost in the class of affine approximations.

The different patient scheduling policies are compared in terms of mean discounted cost, mean average wait times, mean average time to first available appointment slot and mean percentage of late bookings. We will start by studying a small example and then analyze two more practical settings, one taken from Patrick et al. (2008). The computer used to run the algorithm was a 3.00GHz Quad Core PC with 16GB of RAM.

5.3. A Small Clinic

Consider a clinic with a capacity of 6 appointment slots per day. The clinic divides demand into three priority classes with wait time targets of 4, 8 and 12 days, and it chooses a 12-day booking horizon. Demand from each patient type is assumed to be Poisson with means 3, 2 and 1 requests per day, respectively. The overtime cost is 100, the late booking penalties are 20, 10 and 5, and the discount factor is 0.99.

The algorithm required 190 iterations and approximately eight hours to converge to the final parameter values shown in Figure 4. The algorithm was configured as follows: number of simulation runs $R = 150$; number of replications in each simulation run $K = 300$; warm-up period $T_0 = 100$ days; number of days in each simulation run $T = 1,225$; stepsize parameter $\bar{\alpha} = 1$; stopping criterion $\delta = 0.1$; maximum number of iterations $J = 1,500$.

Figure 4 displays the value of b_2 for each post-decision state variable. It gives us an idea of the relative cost associated with each day in the booking horizon. Since the larger the value of b_2 the more costly it is to use capacity on a given day, this figure suggests

Parameter	b_0	b_1	b_3
Value	411.61	4,261.95	4.05

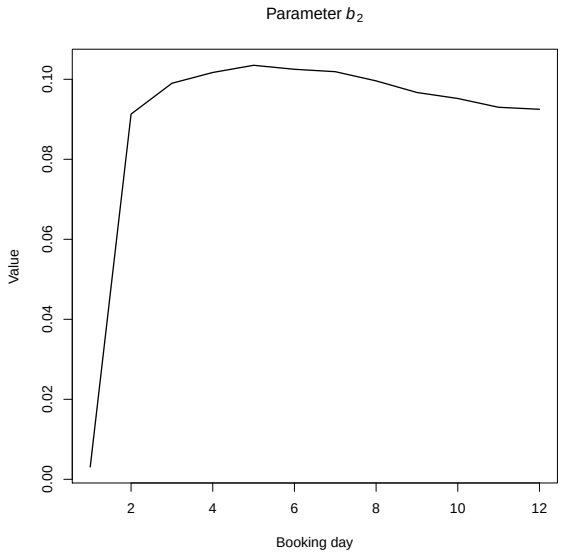


Figure 4 Final values of the approximation parameters.

that the booking agent should schedule patients on days one and two before scheduling them later in the booking horizon. Figure 5 compares the performance of the resultant *SS* policy with the performance of three of the other four policies. The *M* policy is omitted to make the results easier to read (it performs very poorly compared to the other policies and therefore distorts the scales). Each policy was simulated for 1,400 days using common patient arrivals with statistics collected for each of the 1,000 simulation runs after a warm-up period of 100 days. The warm-up period was simulated under Patrick et al.'s policy. Simulation results are summarized in Table 2.

Table 2 shows that the *SS* policy outperforms all the other policies in terms of the mean discounted cost and that it behaves relatively well (no worse than third place) with respect to all other performance metrics. The *M* policy is better than the *SS* policy with respect to the mean average wait time for the lowest priority patients, the mean average daily capacity utilization and the mean number of diversions. However, it is much worse in terms of the mean average wait times for higher priority patients, the mean average time to the first available appointment slot and the mean percentage of late bookings. It books, on average, almost 30% of the patients late. The *PPQ* policy is slightly better than the *SS* policy with respect to the mean average wait time for the highest priority patients and the

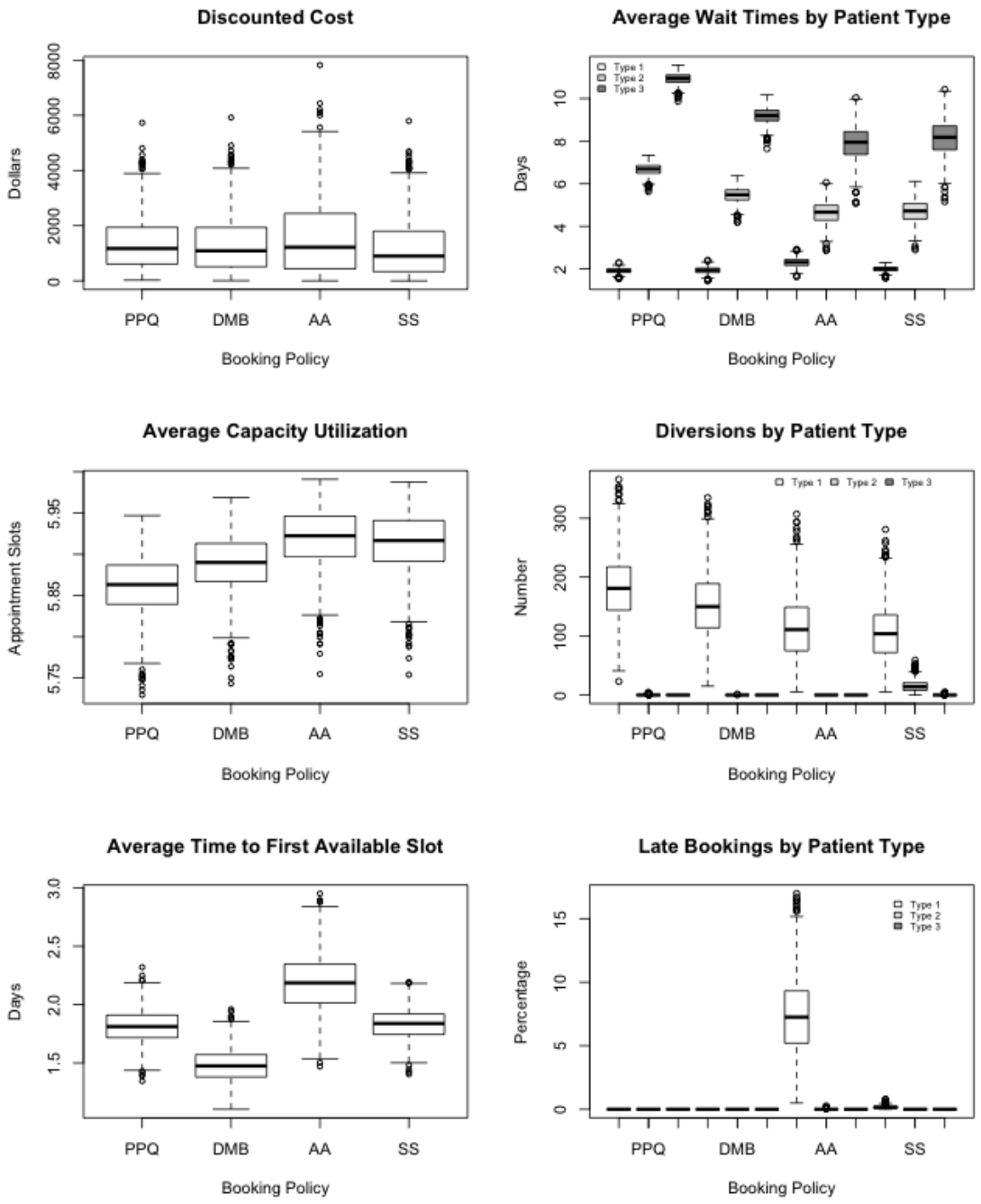


Figure 5 **Simulation results for a clinic with capacity of 6 appointment slots per day. Each policy was simulated for 1,400 days using common patient arrivals with statistics collected for each of 1,000 simulation runs after a warm-up period of 100 days.**

mean average time to the first available appointment slot. This is as a direct consequence of lower daily capacity utilization levels. Additionally, it does not book any patients late. The *PPQ* policy, however, provides much longer average wait times for lower priority patients

Table 2 Summary of the performance of the clinical booking policies. The resulting 95% confidence interval is provided for each statistic. The best values for each criterion are highlighted in bold.

Criterion	Urgency	M	PPQ	DMB	AA	SS
Discounted cost	–	9,229 ± 431	1,390 ± 60	1,332 ± 64	1,573 ± 85	**1,180 ± 64**
Average wait times	1	4.89 ± 0.05	**1.92 ± 0.01**	1.94 ± 0.01	2.30 ± 0.01	2.00 ± 0.01
	2	5.48 ± 0.06	6.67 ± 0.02	5.47 ± 0.02	**4.63 ± 0.03**	4.71 ± 0.03
	3	**5.73 ± 0.06**	10.93 ± 0.02	9.19 ± 0.02	7.89 ± 0.05	8.13 ± 0.05
Average capacity utilization	–	**5.95 ± 0.00**	5.86 ± 0.00	5.89 ± 0.00	5.92 ± 0.00	5.91 ± 0.00
Diversions	1	**70.93 ± 3.14**	182.02 ± 3.30	152.88 ± 3.29	114.47 ± 3.27	106.06 ± 2.85
	2	**0.00 ± 0.00**	0.04 ± 0.02	**0.00 ± 0.00**	**0.00 ± 0.00**	15.57 ± 0.59
	3	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	0.24 ± 0.04
Average time to first available slot	–	4.84 ± 0.06	1.81 ± 0.01	**1.48 ± 0.01**	2.18 ± 0.01	1.83 ± 0.01
Percentage late	1	54.66 ± 0.98	**0.00 ± 0.00**	**0.00 ± 0.00**	7.38 ± 0.18	0.17 ± 0.01
	2	15.92 ± 0.59	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**
	3	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.000**	**0.00 ± 0.00**	**0.00 ± 0.00**

and a much larger mean number of diversions. It diverts, on average, about 5% of the highest priority patients. Relatively speaking, the *DMB* policy performs similar to the *PPQ* policy. The difference in their mean discounted costs is not statistically significant. The *DMB* policy, however, provides shorter average wait times for lower priority patients and the best mean average time to the first available appointment slot. Nevertheless, it still diverts on average almost 4% of the highest priority patients. The *AA* policy is slightly better than the *SS* policy with respect to the mean average wait time for lower priority patients, the mean average daily capacity utilization and the mean number of diversions. However, it schedules on average about 7% of the highest priority patients late while the *SS* policy less than 0.2%.

Three distinctive characteristics allow the *SS* policy to achieve the lowest mean discounted cost among all the policies:

1. It reacts to low workload levels early in the booking horizon by using available capacity to serve lower priority patients sooner.
2. It reacts to low workload levels late in the booking horizon by booking some of the highest priority patients late.
3. It anticipates high workload levels by diverting patients of all types preemptively, even if there still is available capacity within the corresponding wait time targets.

By doing this, the *SS* policy eliminates the need for greater numbers of diversions, and possibly late bookings, later. To illustrate these properties, Figure 6 provides a comparison of the actions suggested by the *SS* policy and the *PPQ* policy (the second best in terms of mean discounted cost) for six selected pre-decision states. Pre-decision states are labeled Case 1 to Case 6 and the differences are highlighted in grey. Most of the time the actions suggested by these two policies will not differ (Case 1). However, if there is a low workload level early in the booking horizon, the *SS* policy will book lower priority patients on days one and two while the *PPQ* policy will book them on their respective wait time targets (Case 2). If the future workload level is low and there is no available capacity within the first four days (the wait time target for the highest priority patients), the *SS* policy will book highest priority patients late whereas the *PPQ* policy will divert them at a higher cost (Case 3). Finally, if the workload level is high early in the booking horizon, the *SS* policy will divert patients of all types preemptively while the *PPQ* policy will always use the available capacity within the corresponding wait time targets (Cases 4 to 6). The number of diversions, however, will depend on the workload level. In some cases the *SS* policy will divert all the new demand, in others, only a fraction of it.

To justify our choice of stepsize parameter ($\bar{\alpha}$) and stopping criterion (δ), we performed independent sensitivity analyses on these two configuration parameters. We first ran the algorithm for values of $\bar{\alpha}$ ranging from 1 to 20. We found that the final value function approximations, as depicted in Table 3 and Figure 7, did not differ significantly in terms of $\vec{b}_2$ and b_3, but they did with respect to b_0 and b_1. Even though value function estimates remained the same in a relative sense, early booking, late booking and preemptive diversion thresholds slightly changed, altering the behavior of the consequent policy. In spite of this, the performance of the system did not change significantly. We ended up choosing $\bar{\alpha} = 1$ mainly because of time considerations. We then ran the algorithm for values of δ ranging from 1 to 0.01. We observed that values of δ smaller than or equal to 0.1 provided almost the same value function approximation and therefore an almost identical patient scheduling policy. For this reason, we adopted $\delta = 0.1$. We also analyzed the effect of independent changes in the diversion cost (h) and the demand rates ($\vec{m}$). Results are presented in Appendix A.

Finally, we observed that the mean discounted cost associated with the *SS* policy was 20% lower than the mean discounted cost associated with the *PPQ* policy for the base

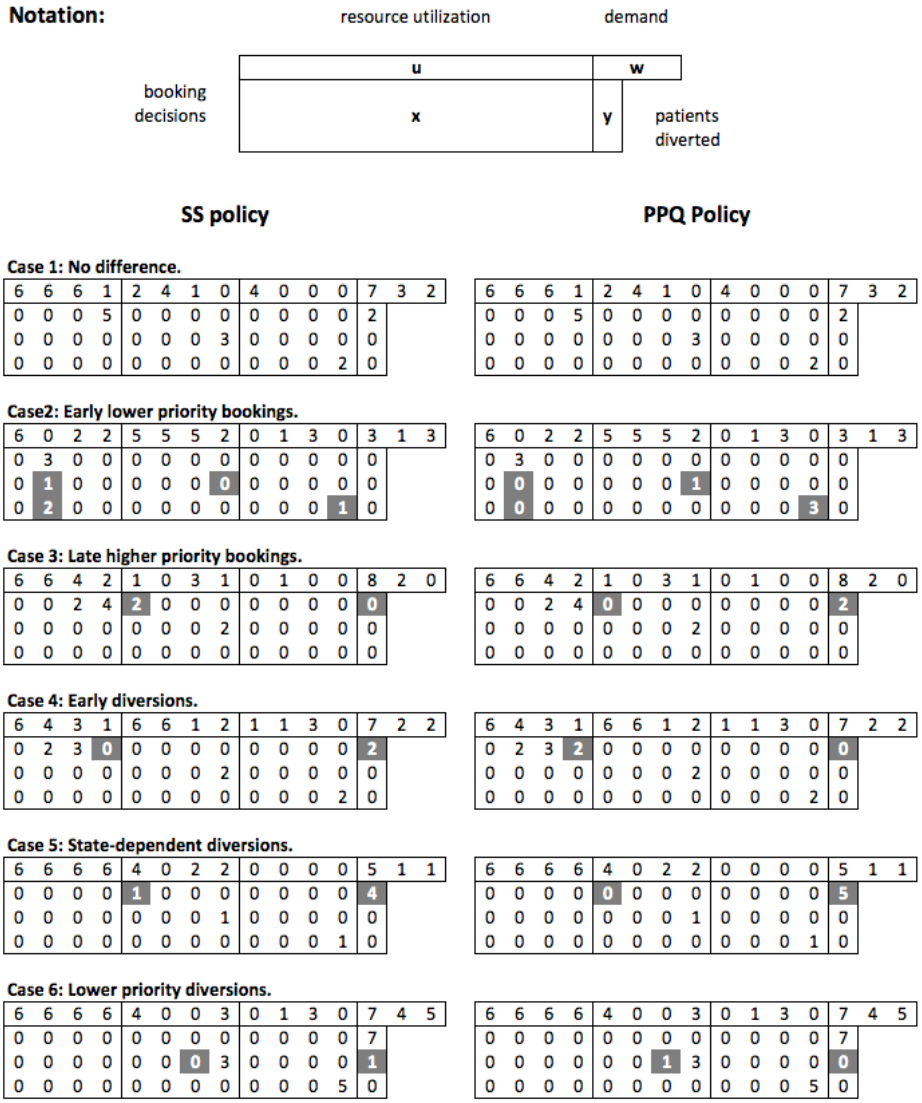


Figure 6 **A comparison of the SS policy (left) and the PPQ policy (right) for six pre-decision states. States and actions are presented according to the notation at the top of the figure. For each case (side-by-side diagrams), the shaded cells indicate how the booking actions differ from each other.**

case when a discount factor of 0.999 was used. In this setting, the *SS* policy provided the lowest mean discounted cost and the lowest mean number of diversions, with almost no patients booked late, closely followed by the *DMB* policy. The algorithm required 154 iterations and approximately 7 days to determine the final value function approximation for this scenario. The following configuration was used: number of simulation runs $R = 150$; number of replications in each simulation run $K = 600$; warm-up period $T_0 = 100$ days; number of decision epochs in each simulation run $T = 14,630$ days; stepsize parameter $\bar{\alpha} = 1$; stopping criterion $\delta = 0.1$; maximum number of iterations $J = 500$.

Table 3 Final values of approximation parameters b_0, b_1 and b_3, and computation times, for different values of $\bar{\alpha}$ and δ.

Scenario	b_0	b_1	b_3	# iterations	Time [hrs.]
$\bar{\alpha} = 1$	411.61	4,261.95	4.05	190	8.1
$\bar{\alpha} = 2$	406.29	4,457.74	4.11	390	17.0
$\bar{\alpha} = 3$	408.28	4,520.01	4.13	390	17.0
$\bar{\alpha} = 5$	427.36	4,999.01	4.16	738	32.4
$\bar{\alpha} = 10$	426.15	5,077.50	4.13	1,500*	64.7
$\bar{\alpha} = 20$	430.61	4,950.87	4.09	1,500*	64.9
$\delta = 1.00$	509.22	3,779.90	4.03	24	1.0
$\delta = 0.10$	411.60	4,261.95	4.05	190	8.1
$\delta = 0.01$	408.56	4,546.71	4.11	1,500*	42.7

"*" indicates the cases for which the maximum number of iterations was reached.

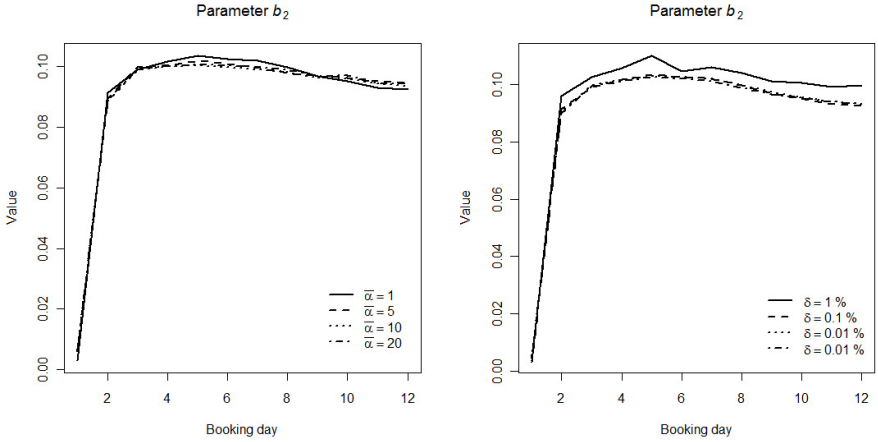


Figure 7 Final value of approximation parameter $\vec{b}_2$ for different values of $\bar{\alpha}$ and δ.

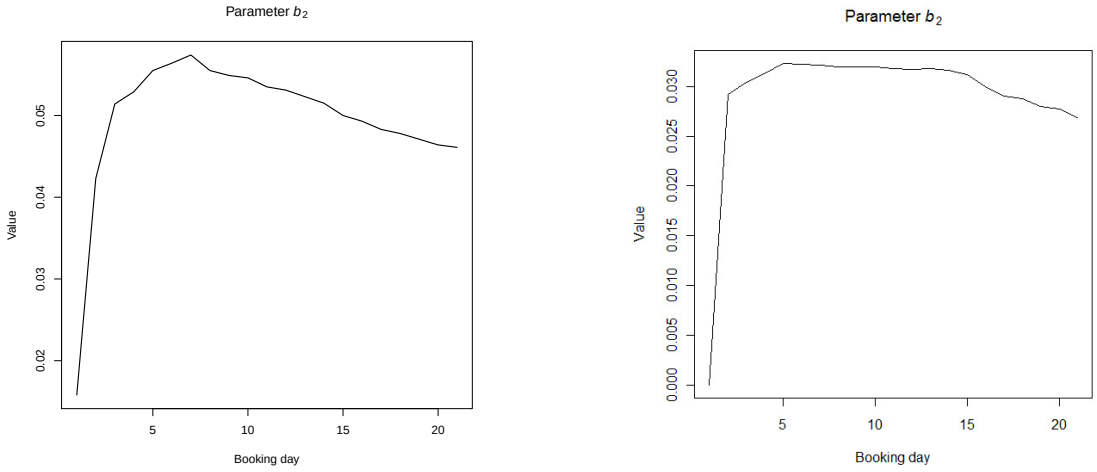
5.4. Two Larger Clinics

Consider now two larger clinics, Clinic 1 and Clinic 2, one with a capacity of 10 appointment slots per day and the other with a capacity of 30 appintment slots per day. The clinics divide demand into three priority classes with wait time targets of 7, 14 and 21 days, and they choose a 21-day booking horizon. Demand from each patient type is assumed to be Poisson with means 5, 3 and 2 requests per day for Clinic 1, and 15, 10 and 5 requests per day for Clinic 2. The diversion cost, the late booking penalties and the discount factor remain the same as in the small clinic setting.

The overall shape of $\vec{b}_2$ in Figure 8a is similar to the one obtained for the small clinic example. However, the actual b_2 values are smaller and increase and decrease faster. This translates into more patients being booked late and fewer diversions. The values of b_2 in Figure 8b are significantly lower than the values obtained before. They also drop quickly

Table 4 **Final values of approximation parameters b_0, b_1 and b_3, and computation tiems, for two larger clinics.**

Scenario	b_0	b_1	b_3	# iterations	Time [hrs.]
Clinic 1 ($C=10$)	21.10	7,684.77	6.51	212	42
Clinic 2 ($C=30$)	55.04	13,975.09	10.17	332	144



(a) Clinic 1 ($C=10$). **(b)** Clinic 2 ($C=30$).

Figure 8 **Final value of approximation parameter $\vec{b}_2$ for two larger clinics.**

below the level associated with day two, suggesting that unless there is available capacity on day one the resulting booking policy will never book lower priority patients early in the booking horizon. Table 4 makes evident one of the major limitations of our algorithm in comparison to a linear programming approach, the large amount of time required to find the final values of the approximation parameters. The algorithm, for example, required 332 iterations and approximately six days in the case of the larger clinic. Linear programming approaches, however, are mostly limited to affine approximation architectures. They are not suitable for non-linear architectures such as the generalized logistic function.

Table 5 shows the mean discounted cost associated with each booking policy for clinics 1 and 2. Each policy was simulated using common patient arrivals with statistics collected for each of 1,000 simulation runs. The simulation results for Clinic 1 did not differ considerably from those obtained for the small clinic example, so similar conclusions apply. The *SS* policy outperformed all the other policies in terms of the mean discounted cost and behaved relatively well with respect to all the other performance metrics. The difference in terms of mean discounted cost observed between the *SS* policy and the *PPQ* policy was smaller but statistically significant. This was corroborated by a paired- samples t-test, with the level of

Table 5 Mean discounted cost associated with the booking policies for two clinics with capacities of 10 and 30 appointment per day. The resulting 95% confidence intervals are provided. The best values are highlighted in bold.

Scenario	M	PPQ	DMB	AA	SS
Clinic 1 ($C=10$)	19,507 ± 813	919 ± 70	1,063 ± 79	1,772 ± 127	**889** ± **75**
Clinic 2 ($C=30$)	56,827 ± 2,598	**943** ± **102**	1,083 ± 112	15,479 ± 476	1,399 ± 132

significance set at 0.05. The performance of the *SS* policy for Clinic 2 was clearly different. It only achieved the third best mean discounted cost, after the *DMB* and the *PPQ* policy. Although the *DMB* and the *SS* policy produced higher discounted cost values than the *PPQ* policy, they diverted on average fewer patients. Since none of these three policies booked patients late, the only explanation for this result was the use of diversions earlier in the simulation runs. Consequently, both the *DMB* and *SS* policy ended up performing better than the *PPQ* policy in steady-state conditions. The average cost associated with the *PPQ*, the *DMB* and the *SS* policy were 12.78, 11.07 and 10.58 when each policy was simulated for 10,000 days with cost values collected after a warm-up period of 1,500 days.

Natural extensions of the research in this paper would be a study of the impact of the initial state conditions on the performance of each policy and the development of an average cost version of our algorithm.

6. Conclusion

In this paper, we describe a simulation-based algorithm for solving high-dimensional dynamic programming models that is formulated in terms of post-decision states and in which the value function is approximated using a generalized logistic function.

The benefits from using a generalized logistic function in the post-decision state variables are clear, in most cases, for an advance patient scheduling problem. However, this approximation architecture introduces numerous additional challenges into an approximate policy iteration algorithm. For example, the problem of finding an action at a given decision epoch involves solving a non-linear integer program. In some cases this can be done easily, in others, an additional approximation is needed, introducing a new source of error. In addition, a policy improvement step entails solving a non-linear least squares problem. Unlike in the linear case, this problem does not necessarily have a closed or simple recursive form, and, additionally, there is no guarantee that a non-linear solver will converge to a good solution starting from an arbitrary initial solution. For this reason, a good initial solution, reasonable bounds and appropriate scaling factors are needed to enhance

the likelihood of convergence. Furthermore, the convergence rate of an approximate policy iteration algorithm depends on the quality of the initial policy. Thus, having either a very good understanding of the system dynamics or knowing a good existing policy is key.

Based on our experience solving an advance patient scheduling problem, the main limitation of the proposed algorithm is the large amount of time required to perform a policy evaluation step. This is a direct consequence of the slow speed of simulation and the complexity of the optimization problem solved at each decision epoch. In order to be able to solve large instances of high-dimensional problems, a more efficient code should be written. If this is not possible, alternative simulation-based algorithms such as temporal-difference learning or direct search methods (Maxwell et al. 2010b) could be adapted to the particular setting. Parallel computing could also be used to speed up evaluation steps. Unfortunately, there is no guarantee of success. In this context, although the linear programming approach is mostly limited to affine approximation architectures, its main advantage is that it avoids the need for iterative learning and still provides good results.

The results for a more practical application demonstrate that the value function for a multi-priority patient scheduling model can be better represented by a generalized logistic function. They also show that patient scheduling policies obtained through the proposed algorithm perform better, in most cases, than other policies for this problem. In particular, they provide lower discounted cost values and shorter average wait times for lower priority patients than policies directly obtained from the linear programming approach using an affine value function approximation.

Acknowledgments

This work was partially supported by the Natural Sciences and Engineering Research Council of Canada [Grant RGPIN-5527] and by the Canadian Institutes of Health Research [Grant AQC-83512]. The authors would also like to thank Maurice Queyranne and Steven Shechter for providing considerable assistance throughout the course of this research.

Appendix A: Additional Sensitivity Analysis

We also analyzed the effect of independent changes in the diversion cost (h) and the demand rates ($\vec{m}$). We first ran the algorithm for values of h between 80 and 120. We observed that b_0 and b_1 increased a bit less than proportional to the changes in h while $\vec{b}_2$ remained practically the same (see Table 6 and Figure 9). The resulting policies kept similar utilization and service levels by booking more highest priority patients late and by diverting almost the same number of patients but in a different mix. The larger the value of h, the fewer highest priority patients and the more lower priority patients were diverted. We then ran the algorithm

Table 6 **Final values of approximation parameters b_0, b_1 and b_3, and computation times, for different values of h and $\vec{m}$.**

Scenario	b_0	b_1	b_3	# iterations	Time [hrs.]
$h = 80$	326.50	3,328.13	4.06	284	12.1
$h = 90$	369.37	3,746.55	4.05	190	8.1
$h = 100$	411.60	4,261.95	4.05	190	8.1
$h = 110$	458.00	4,835.75	4.06	220	9.3
$h = 120$	505.57	5,321.90	4.06	220	9.3
$\vec{m} = 0.9 \times \vec{m}$	15.41	1,143.43	5.17	544	22.9
$\vec{m} = 1.0 \times \vec{m}$	411.60	4,261.95	4.05	190	8.1
$\vec{m} = 1.1 \times \vec{m}$	6,445.19	270,534.12	2.04	655	30.2

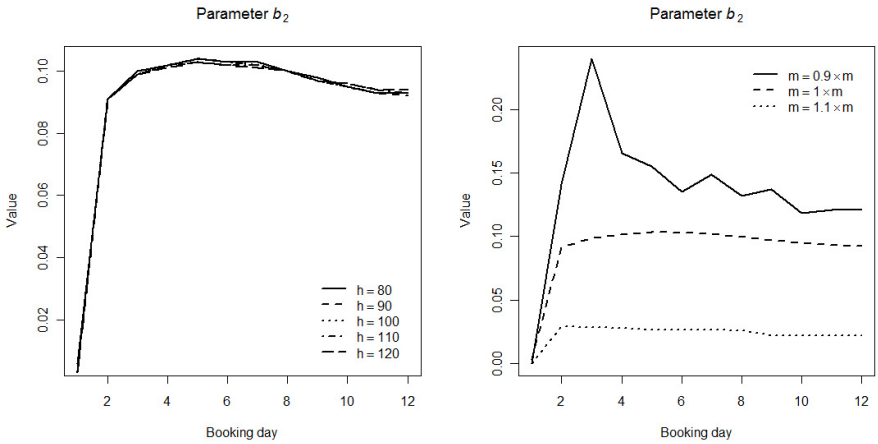


Figure 9 **Final value of approximation parameter $\vec{b}_2$ for different values of h and $\vec{m}$.**

for demand rates 10% lower and 10% higher than in the base case. The final value function approximation, as depicted in Table 6 and Figure 9, changed drastically. In the first case, the resulting policy recognized the system was over-capacitated. It booked patients earlier in the booking horizon and handled demand peaks using late bookings rather than diversions. In the second case, the resulting policy reacted to the lack of capacity by booking patients on day one, and sometimes on day two, and by diverting any remaining demand.

References

Adelman, D. 2004. A price-directed approach to stochastic inventory/routing. *Oper. Res.* **52** 499–514.

Adelman, D., D. Klabjan. 2012. Computing near-optimal policies in generalized joint replenishment. *INFORMS J. Comput.* **24** 148–164.

Adelman, D., A. Mersereau. 2008. Relaxations of weakly coupled stochastic dynamic programs. *Oper. Res.* **56** 712–727.

Astaraky, D., J. Patrick. 2013. A simulation based approximate dp approach to multi-class, multi-resource surgical scheduling. Unpublished.

Bertsekas, D. 2010. *Dynamic Programming and Optimal Control*, vol. II, chap. 6. 3rd ed. Unpublished, 320–527.

Bertsekas, D., J. Tsitsiklis. 1996. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, Mass.

Das, T., A. Gosavi, S. Mahadevan, N. Marchalleck. 1999. Solving semi-Markov decision problems using average reward reinforcement learning. *Management Sci.* **45** 560–574.

de Farias, D., B. Van Roy. 2003. The linear programming approach to approximate dynamic programming. *Oper. Res.* **51** 850–865.

de Farias, D., B. Van Roy. 2004. On constraint sampling in the linear programming approach to approximate dynamic programming. *Math. Oper. Res.* **29** 462–478.

Desai, V., V. Farias, C. Moallemi. 2009. A smoothed approximate linear program. *Adv. Neur. In.* **22** 459–467.

Enders, J., W. Powell, D. Egan. 2010. Robust policies for the transformer acquisition and allocation problem. *Energy Syst.* **1** 245–272.

Erdelyi, A., H. Topaloglu. 2009. Computing protection level policies for dynamic capacity allocation problems by using stochastic approximation methods. *IIE Trans.* **41** 498–510.

Frantzeskakis, L., W. Powell. 1990. A successive linear approximation procedure for stochastic, dynamic vehicle allocation problems. *Transport. Sci.* **24** 40–57.

GAMS. 2011. GAMS - the solver manuals. Tech. rep., GAMS Development Corporation.

Gocgun, Y., M. Puterman. 2013. Dynamic scheduling with due dates and time windows: An application to chemotherapy patient appointment booking. *Health Care Manage. Sci.* .

Godfrey, G., W. Powell. 2002. An adaptive dynamic programming algorithm for dynamic fleet management, i: Single period travel times. *Transport. Sci.* **36** 21–39.

Gosavi, A., N. Bandla, T. Das. 2002. A reinforcement learning approach to a single leg airline revenue management problem with multiple fare classes and overbooking. *IIE Trans.* **34** 729–742.

Hing, M., A. Van Harten, P. Schuur. 2007. Reinforcement learning versus heuristics for order acceptance on a single resource. *J. Heuristics* **13** 167–187.

Lam, S., L. Lee, L. Tang. 2007. An approximate dynamic programming approach for the empty container allocation problem. *Transport. Res. C-Emer.* **15** 265–277.

Marbach, P., O. Mihatsch, J. Tsitsiklis. 2000. Call admission control and routing in integrated services networks using neuro-dynamic programming. *IEEE J. Sel. Area. Comm.* **18** 197–208.

Maxwell, M., S. Henderson, H. Topaloglu. 2010a. Tuning approximate dynamic programming policies for ambulance redeployment via direct search. Unpublished.

Maxwell, M., M. Restrepo, S. Henderson, H. Topaloglu. 2010b. Approximate dynamic programming for ambulance redeployment. *INFORMS J. Comput.* **22** 266–281.

Patrick, J., M. Puterman, M. Queyranne. 2008. Dynamic multipriority patient scheduling for a diagnostic resource. *Oper. Res.* **56** 1507–1525.

Powell, W. 1987. An operational planning model for the dynamic vehicle allocation problem with uncertain demands. *Transport. Res. B-Meth.* **21** 217–232.

Powell, W. 2011. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley-Interscience, Hoboken, N.J.

Puterman, M. 1994. *Markov decision processes: discrete stochastic dynamic programming*. Wiley-Interscience, New York.

Sauré, A., J. Patrick, S. Tyldesley, M. Puterman. 2012. Dynamic multi-appointment patient scheduling for radiation therapy. *Eur. J. Oper. Res.* **223** 573–584.

Schmid, V. 2012. Solving the dynamic ambulance relocation and dispatching problem using approximate dynamic programming. *Eur. J. Oper. Res.* **219** 611–621.

Schütz, H., R. Kolisch. 2012. Approximate dynamic programming for capacity allocation in the service industry. *Eur. J. Oper. Res.* **218** 239–250.

Schweitzer, P., A. Seidmann. 1985. Generalized polynomial approximations in Markovian decision processes. *J. Math. Anal. Appl.* **110** 568–582.

Simão, H., J. Day, A. George, T. Gifford, J. Nienow, W. Powell. 2009. An approximate dynamic programming algorithm for large-scale fleet management: A case application. *Transport. Sci.* **43** 178–197.

Simão, H., A. George, W. Powell, T. Gifford, J. Nienow, J. Day. 2010. Approximate dynamic programming captures fleet operations for schneider national. *Interfaces* **40** 342–352.

Simao, H., W. Powell. 2009. Approximate dynamic programming for management of high-value spare parts. *J. Manuf. Technol. Manage.* **20** 147–160.

Sutton, R., A. Barto. 1998. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, Mass.

Van Roy, B., D. Bertsekas, Y. Lee, J. Tsitsiklis. 1997. A neuro-dynamic programming approach to retailer inventory management. *Decision and Control, 1997., Proceedings of the 36th IEEE Conference on*, vol. 4. IEEE, 4052–4057.

Zhang, D., D. Adelman. 2009. An approximate dynamic programming approach to network revenue management with customer choice. *Transport. Sci.* **43** 381–394.

Zhang, W., T. Dietterich. 1995. High-performance job-shop scheduling with a time-delay TD(λ) network. *Adv. Neur. In.*. 1024–1030.