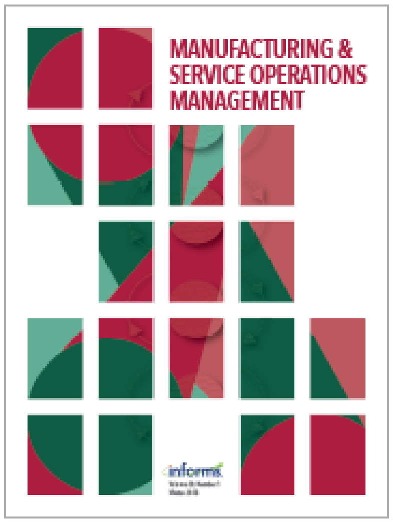


Manufacturing & Service Operations Management

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Inpatient Overflow: An Approximate Dynamic Programming Approach

J. G. Dai, Pengyi Shi

informs
http://pubsonline.informs.org/journal/msom

MANUFACTURING & SERVICE OPERATIONS MANAGEMENT
Vol. 21, No. 4, Fall 2019, pp. 894–911
ISSN 1523-4614 (print), ISSN 1526-5498 (online)

Inpatient Overflow: An Approximate Dynamic Programming Approach

J. G. Dai,[a,b] **Pengyi Shi**[c]

[a] School of Operations Research and Information Engineering, Cornell University, Ithaca, New York 14853; [b] Institute for Data and Decision Analytics, The Chinese University of Hong Kong, Shenzhen 518172, China; [c] Krannert School of Management, Purdue University, West Lafayette, Indiana 47907

Contact: jd694@cornell.edu, http://orcid.org/0000-0002-5223-0129 (JGD); shi178@purdue.edu, http://orcid.org/0000-0003-0905-7858 (PS)

Received: February 23, 2017
Revised: September 22, 2017; December 28, 2017
Accepted: January 21, 2018
Published Online in Articles in Advance: May 16, 2019

https://doi.org/10.1287/msom.2018.0730

Abstract. ***Problem definition*:** Inpatient beds are usually grouped into several wards, and each ward is assigned to serve patients from certain "primary" specialties. However, when a patient waits excessively long before a primary bed becomes available, hospital managers have the option to assign her to a nonprimary bed. although it is undesirable. Deciding when to use such "overflow" is difficult in real time and under uncertainty. ***Relevance*:** To aid the decision making, we model hospital inpatient flow as a multiclass, multipool parallel-server queueing system and formulate the overflow decision problem as a discrete-time, infinite-horizon average cost Markov decision process (MDP). The MDP incorporates many realistic and important features such as patient arrival and discharge patterns depending on time of day. ***Methodology*:** To overcome the curse-of-dimensionality of this formulated MDP, we resort to approximate dynamic programming (ADP). A critical part in designing an ADP algorithm is to choose appropriate basis functions to approximate the relative value function. Using a novel combination of fluid control and single-pool approximation, we develop analytical forms to approximate the relative value functions at midnight, which then guides the choice of the basis functions for all other times of day. ***Results*:** We demonstrate, via numerical experiments in realistic hospital settings, that our proposed ADP algorithm is remarkably effective in finding good overflow policies. These ADP policies can significantly improve system performance over some commonly used overflow strategies—for example, in a baseline scenario, the ADP policy achieves a congestion level similar to that achieved by a complete bed sharing policy, while reduces the overflow proportion by 20%. ***Managerial implications*:** We quantify the trade-off between the overflow proportion and congestion from implementing ADP policies under a variety of system conditions and generate useful insights. The plotted efficient frontiers allow managers to observe various performance measures in different parameter regimes, and the ADP policies provide managers with operational strategies to achieve the desired performance.

Funding: This research is supported in part by the National Science Foundation, Division of Civil, Mechanical and Manufacturing Innovation [Grants 1335724 and 1537795].
Supplemental Material: The online appendix is available at https://doi.org/10.1287/msom.2018.0730.

Keywords: multiclass multipool queueing system • inpatient flow management • approximate dynamic programming

1. Introduction

Inpatient bed management is crucial for hospital operations, particularly for upstream emergency department (ED) management. It is well known that a key contributor for ED overcrowding is ED boarding—holding admitted patients in the ED until downstream inpatient beds become available (Hoot and Aronsky 2008). Prolonged boarding time negatively affects patient outcomes (Singer et al. 2011) and increases hospital operational costs (Huang et al. 2010, Pines et al. 2011). Because of budgetary constraints, adding inpatient capacity such as increasing the number of inpatient beds or nurses is not always possible. Various strategies have been proposed with the aim of utilizing the existing capacity more efficiently—for example, smoothing elective surgical schedule or expediting inpatient discharges; see the summary and references in Rabin et al. (2012). This paper focuses on the *overflow* strategy.

1.1. Bed Pooling and Overflow

Hospitals usually partition their general inpatient beds ("floor beds") into different wards according to medical specialties, such as surgical wards and cardiology wards. Ideally, a patient should be admitted to a *primary* ward that matches her medical needs. However, because of the inherent variations in patient arrivals and discharges, sometimes the primary ward can be full while other nonprimary wards still have available beds. In this situation, hospital managers may choose to assign this patient to a nonprimary ward, especially when the patient has boarded in the ED for several hours. We call this practice *overflow* and the patient an *overflow patient*. An empirical study at a partner hospital in Singapore, which motivated this paper, shows that more than 20% inpatient admissions from ED were overflow patients between 2008 and 2010 (Shi et al. 2014).

On the one hand, overflow helps shorten the boarding time because resource pooling has the benefits of reducing waiting time. On the other hand, excessive overflow is not desirable for a number of reasons: A mixed patient population in the ward may require more coordination from the medical teams (Rabin et al. 2012); the quality of patient care could be compromised (Song et al. 2018); and physicians waste more time to travel among wards to do rounding (Gesenway 2010). Thus, hospital managers always need to balance the *key tradeoff* between excessive waiting and undesirable overflow. This is not an easy task. As pointed out by a group of practitioners (Teow et al. 2011), when deciding whether to assign a patient to a nonprimary ward or not, one needs to consider many factors, such as the current crowdedness in the ED, the projected bed requests and discharges (which are usually time-varying), and the medical similarities between the patients and wards. In addition, as we will further discuss in Section 2, overflow patients consume system capacity; they may block new patients from being admitted, causing even more congestion.

In this paper, we develop a decision support tool to help hospital managers tackle this difficult overflow decision problem. The model and algorithm we develop are for general multipool parallel-server queueing systems; below, we use an example of a five-pool system to demonstrate that our developed algorithm can significantly improve various system performance over some commonly used overflow strategies.

1.2. A Five-Pool System Example

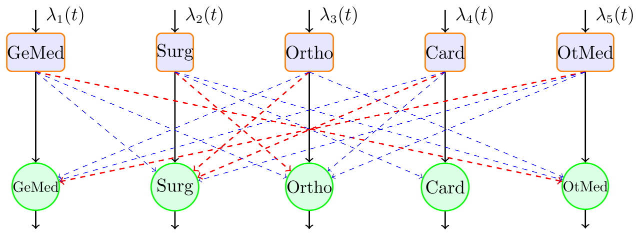
Consider a five-class, five-pool parallel-server queueing system that is motivated from the aforementioned Singaporean hospital. The customers model patients who require inpatient service, such as patients who need to be transferred from ED or intensive care units (ICUs) to the general inpatient wards, or patients admitted for elective surgeries. Their bed-requests correspond to the arrivals in this system. The five customer classes model five medical specialties: General Medicine (GeMed), Surgery (Surg), Orthopedic (Ortho), Cardiology (Card), and Other Medicine (OtMed). Each of the five server pools models one inpatient ward (sometimes a group of similar wards) that is allocated to the corresponding specialty. Within each pool, there are multiple servers corresponding to the beds in the ward(s). In this paper, we use patient and customer, bed and server, server pool and ward, bed-request and arrival, and discharge and departure, interchangeably.

Based on the hospital's internal bed assignment guideline (National University Hospital 2011), we assume that each specialty has one dedicated ward as its primary ward, and its patients can be routed to three nonprimary wards: one "preferred" overflow ward (i.e., beds in this ward are preferred for overflow use) and two secondary overflow wards. Figure 1 shows a schematic representation of the patient-bed configuration, with the solid arrows indicating the primary patient-bed assignment, and the thick and thin dashed arrows indicating the preferred and secondary overflow assignments, respectively.

We assume that patient arrivals from each speciality follow a periodic, time-nonhomogeneous Poisson process. Each patient, upon admission to a bed, spends a random amount of *service time* before being discharged from the hospital. This service time consists of (1) a random number of days spent, referred as the length-of-stay (LOS), and (2) some (random) extra hours spent on the day of discharge. The former captures the time for a patient to recover and is driven by her underlying medical conditions; the latter captures the extra delay presented during the discharge process and is usually caused by medical staff schedule such as physician's rounding time. We specify the details of this *two-time-scale* service time in Section 3.1. Note that the descriptions here are for the basic queueing model to develop our decision framework and algorithm. In the numerical study reported at the end of this section and in Section 7, we go beyond this basic model and test our developed algorithm in a more realistic simulation model populated from hospital data—for example, using separate, non-Poisson arrival processes for elective and ICU-transfer patients. We leave the details of the extended simulation model to Section 7.1.

Overflow Decisions. The decision maker (e.g., the bed management unit in a hospital) observes the system

Figure 1. (Color online) Illustration of the Five-Class, Five-Pool System



Notes. Pools 1–5 are the primary server pools for General Medicine, Surgery, Orthopedic, Cardiology, and Other Medicine, respectively. Solid lines represent the primary bed assignment, thick dashed lines represent the preferred overflow assignment, and thin dashed lines represent the secondary overflow assignment.

state at certain fixed time epochs every day and decides on (1) whether to assign a waiting patient to a nonprimary ward, and (2) which ward to use if multiple nonprimary wards have available beds. A primary bed assignment, if possible, is made immediately upon a patient arrival or bed release because this is the most ideal situation, and, thus, it is excluded from the decisions at each epoch. We model the overflow decisions as a Markov decision process (MDP), where the decision maker aims to minimize the long-run average cost over (infinitely) many epochs. The total cost in each epoch includes:

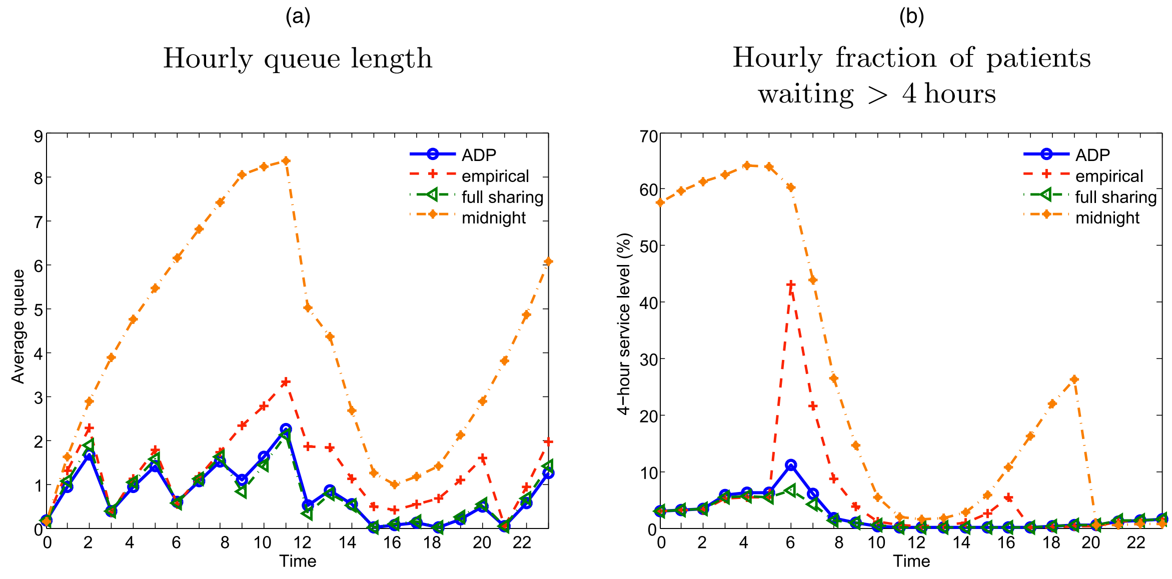
- The *holding cost* associated with the number of waiting patients, which captures the system congestion; and
- The *overflow cost* associated with each overflow patient, which captures the nondesirableness of placing a patient in a nonprimary ward.

These two costs represent the key tradeoff in the overflow decision and provide us a "tuning knob" to achieve target system performance, as we will further discuss below. We assume that both costs are class-dependent, where for each patient class: (1) the overflow cost of assigning a patient to a preferred overflow ward is cheaper than that to a secondary overflow ward; and (2) the holding cost is linear in the number of waiting patients. Also see section B.4 in the online appendix for a setting in which patients who wait longer incur a more expensive holding cost.

Current Practice. From our discussions with staff members at the partner hospital in Singapore, we understand that for decision (a)—"whether to"—the internal guideline suggests performing more aggressive overflow during the night and early morning, because a primary bed, if not available upon a patient arrival, is unlikely to become available in the next few hours, because few discharges occur in this period. For decision (b)—"which ward"—a commonly used practice is the *priority strategy*—that is, use beds in the preferred overflow wards first and then use beds in the secondary overflow wards. Accordingly, we choose the following three naive policies as the benchmark for policy comparison: (1) *full-sharing policy*: allow overflow at each decision epoch; (2) *midnight policy*: allow overflow only at the midnight epoch every day; and (3) *empirical policy*: allow overflow only between 7 p.m. and 7 a.m. the next day. All three policies use the priority strategy and admit as many patients as possible at each epoch allowing overflow. We call the last policy the empirical policy, because it mimics the hospital's current practice and can produce a set of performance curves that are close to the empirical ones; see model validation in section B.2 in the online appendix.

Policy Comparison. We propose an approximate dynamic programming (ADP) algorithm to solve the MDP and compare the corresponding ADP policy with the three naive policies. Under a "baseline scenario" to be specified in Section 7, Figure 2, (a) and (b), plots the hourly queue length (average number of patients boarding in the ED) and the four-hour service level (fraction of patients waiting more than four hours in the

Figure 2. (Color online) Hourly Average Queue Length and Fraction of Patients Waiting at Least Four Hours (Four-Hour Service Level) Estimated from Simulating Different Overflow Policies in the Baseline Five-Pool Model



Note. The half-width of the 95% confidence interval is 0.001–0.005 for the queue length across all tested policies and is 0.01%–0.15% for the four-hour service level.

ED prior to ward admission), respectively, for each policy. See section B.2 of the online appendix for the motivation of choosing this specific service level and section 6.4 of the online supplement in Dai and Shi (2018) for plots on other service levels. Figure 3 shows the average overflow proportions (on the vertical axis) from the ADP policy and naive policies. We observe that the ADP policy approximately achieves the hourly queue length and four-hour service level under the full-sharing policy, which is the least congested scenario any policy can ever achieve. Meanwhile, comparing to the 15.5% overflow proportion under the full-sharing policy, the ADP policy reduces the overflow proportion to 12.5%, a 20% reduction. For a hospital with a daily throughput of 100 patients and an average LOS of 5 days, this reduction translates to $100(0.155 - 0.125) * 5 \approx 15$ fewer patients in a nonprimary ward every day by Little's Law. In addition, the ADP policy reduces the long-run average cost by more than 17% compared with the naive policies.

The ADP policy shown in Figure 2 is from one set of parameters of the unit holding cost and overflow cost. In practice, estimating these cost parameters can be difficult, if not impossible. In this paper, we mainly use these cost parameters as a "tuning knob" to affect both aggregate and class-level performance. For example, Figure 3 plots the overflow proportion against the peak queue length and four-hour service level from a series of ADP policies, where we keep the unit holding cost the same as in the baseline scenario, but incrementally change the overflow costs. Such "efficient frontier" plots allow managers to directly observe various performance measures under different cost parameters and identify the operating regime according to their desired performance. More importantly, the ADP policies provide managers with operational strategies to *achieve* the desired performance.

1.3. Contributions

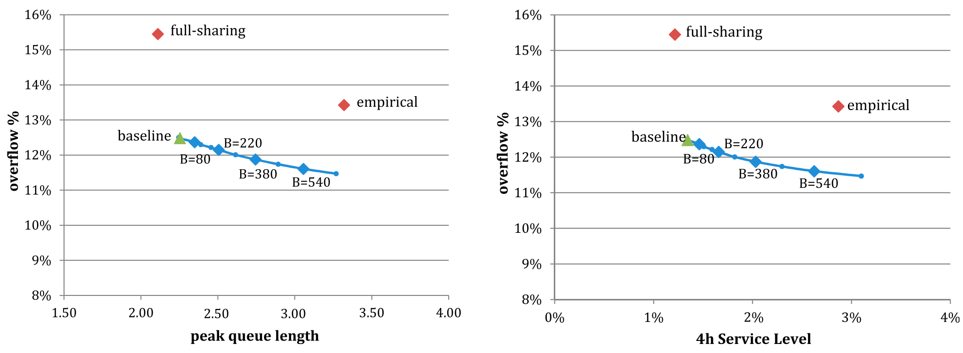
This paper makes three major contributions to the literature.

First, we formulate the overflow decision problem as a discrete-time, infinite-horizon average cost MDP within a multiclass, multipool queueing network setting. This MDP incorporates many realistic features of hospital inpatient operations. In particular, by explicitly modeling the time-varying arrival and discharge patterns, the decision makers not only need to account for the randomness in arrivals and departures on the *daily scale*, but also have to cope with the randomness on the *hourly scale*. The latter indeed is a major challenge for hospital managers, because every hour of waiting matters in ED, and they have to decide whether to assign a waiting patient to a nonprimary bed without knowing for sure when a primary bed will become available in the next few hours.

Second, the formulated MDP has a high-dimensional state space, which renders traditional approaches, such as policy iteration, computationally infeasible, even with a small number of pools. Instead, we resort to the ADP approach with feature-based value function approximation. A critical part in designing the ADP algorithm is to choose appropriate basis functions. We start by studying a special midnight MDP and develop a novel combination of a fluid control model and an integrated single-pool model to approximate its relative value functions. The analytical form of this approximation motivates our choice of the basis functions for the midnight epoch. Then, we derive the basis functions for other times of day and develop explicit forms to evaluate the cost-to-go function efficiently, further contributing to the overall effectiveness of our ADP algorithm.

Third, we demonstrate, via extensive numerical experiments, that our ADP algorithm is effective in finding

Figure 3. (Color online) Overflow Proportion Against Peak Queue Length or Four-Hour Service Level



Notes. In each plot, the two red dots correspond to performance from the full-sharing and empirical policies. The solid curves correspond to performance from a series of ADP policies with the overflow cost pair—costs to a preferred and a secondary overflow ward—changing from (5, 10) to (615, 620); the latter number in the pair is shown following *B*. The dots correspond to performance from the baseline ADP policy with the overflow cost pair (30, 35). The midnight policy is not included here due to its poor performance under the baseline "nonbalanced" bed allocation; see plots under a more balanced bed allocation in section 6.5 of the online supplement.

good overflow policies in a variety of settings. In a two-pool system where the "exact analysis" is feasible, we show that the algorithm achieves near-optimal performance (see Section 6.4). We further implement the ADP algorithm in a more realistic, five-pool simulation model populated with hospital data, where we remove restrictive model assumptions made for developing the analytical framework. The ADP algorithm is still able to find good overflow policies that improve various system performance over the benchmark policies, as already mentioned in this introduction; also see Section 7. We perform various sensitivity analyses and generate useful insights into the benefits of the ADP policies under different system conditions. In addition, we show in Section 5 that the complexity of the ADP algorithm is in the order of J^2, where J is the number of server pools. Thus, the algorithm is scalable and can be applied to other large systems beyond the five-pool system tested in this paper.

1.4. Outline

The remainder of this paper is organized as follows. In Section 2, we review relevant literature. In Section 3, we describe the queueing system to model inpatient flow and the long-run average cost MDP to model the overflow decision problem. In Section 4, we derive a time-decomposed Poisson equation for the long-run average cost MDP with periodic arrivals and departures. In Section 5, we describe the general framework of the ADP algorithm. In Section 6, we study a special midnight MDP and propose the choice of the basis functions for the midnight epoch, which then motives the basis choice for other epochs. In Section 7, we test the ADP algorithm in the five-pool simulation model. Finally, we conclude this paper in Section 8.

2. Literature Review

We review three streams of literature that are related to our work.

2.1. Hospital Inpatient Operations and Bed Assignment

Inpatient flow management is an important area for hospital operations (see Armony et al. 2015 for a detailed empirical study and references). Among this stream of literature, the most relevant work is Thompson et al. (2009). They model the inpatient bed assignment problem as a discrete-time MDP that shares several similar features with our MDP, but the objective is to maximize the total reward ("revenue") brought by patients admitted over a *finite* horizon. Under this objective, the optimal policy may tend to admit more high-revenue patients, but keep low-revenue patients waiting longer. Our aim is to reduce waiting time across all types of patients. Moreover, the algorithm developed in Thompson et al. (2009) assumes a zero terminal cost and uses a two-period total cost to approximate the value function, which is likely to perform suboptimally in our long-run average cost setting. The independent, contemporary work by Kilinc et al. (2016) studies a similar overflow problem, with the key tradeoff also being ED congestion versus the non-desirableness of overflow. They use a continuous-time MDP model and do not incorporate the time-varying discharge pattern. We believe that explicitly modeling the discharge pattern is important in inpatient operations; see Chan et al. (2017), Dai and Shi (2017), and Dong and Perry (2017) for relevant discussions. In addition, Kilinc et al. (2016) derive structural properties and develop heuristics in a two-pool setting, whereas our focus is to develop efficient algorithms for the multipool setting of realistic hospital sizes. Mandelbaum et al. (2012) study bed assignment to balance idleness and workload among wards, where each ward can take any incoming patient (i.e., no concept of overflow). Best et al. (2015) study bed allocation from a more strategic level—that is, deciding the number of "wings" and the number of beds allocated to each wing; overflow between wings is not allowed.

2.2. Routing and Scheduling in Queueing Network

Huang et al. (2015) develop an asymptotically optimal policy for a multiclass queueing network modeling patient flow through physicians within the ED. He et al. (2019) develop a data-driven robust framework to study a similar ED scheduling problem. Their framework utilizes the P-model to maximize the fraction of patients whose total time in the ED is within some mandatory targets. Later, Han et al. (2016) adapt the P-model to the inpatient bed assignment problem and provide a myopic heuristic to optimize the x-hour service level. Besides healthcare applications, there are many works using multipool parallel-server models to study routing policies in call centers; see, for example, Armony and Ward (2010), Dai and Tezcan (2008), Gurvich and Whitt (2009a, b, 2010), Stolyar and Tezcan (2010, 2011), and Tezcan and Dai (2010). A major difference between this line of works and our paper is the cost structure. In addition to the waiting cost that is accumulated continuously in all these models, our model has a *one-time* overflow cost per overflow patient. This overflow cost is different from the one-time abandonment or rejection cost in call-center models. Abandoned or rejected customers in call-center models do not consume any system capacity. However, an overflow patient in our model does occupy a bed. A new patient, who arrives after the overflow patient and could have taken this occupied bed as a primary bed, might need to wait and even be assigned to a nonprimary bed later. Because of this overflow cost, we find that a good bed assignment policy may *not* be nonidling—sometimes it is beneficial to hold a patient in the buffer even a nonprimary ward has an available bed. This finding is consistent with the hospital's current practice as discussed in Section 1.2 and with the empirical evidence (Teow et al. 2011); also see similar findings in Kilinc et al. (2016) and Baron et al. (2017) for strategic idling in open-shop scheduling. In contrast, call-center papers usually assume a nonidling policy. Perry and Whitt (2009, 2011a, b) study the so-called X-model, where the servers from one pool are only allowed to help the other pool during a demand surge period, not the normal random fluctuations in our setting. Pang and Yao (2013) study a multipool system that allows customers to switch buffers, which is uncommon for boarding patients.

2.3. ADP and Its Applications in Healthcare and Queueing Network Control

ADP is a powerful technique for tackling high-dimensional MDP problems; see Bertsekas (2012) and Powell (2011) and their references. Our proposed ADP algorithm falls into the category of feature-based value function approximation, where the value function is approximated by a linear combination of basis functions ("features"). This type of methods requires (1) finding good basis functions and (2) tuning the associated coefficients. There are a number methods to address (2). In this paper, we use the temporal-difference learning method (Sutton 1988); other methods include Q-learning (Watkins and Dayan 1992) and linear programming approach (ALP) (de Farias and Roy 2003, Adelman and Mersereau 2008). However, there is no standard recipe for (1). Choosing good basis functions generally requires substantial knowledge of the specific problem structure and needs one to "combine and customize ingredients from the literature to generate an effective algorithm" as pointed by Moallemi et al. (2008). One of the main contributions of our paper is to design good basis functions in the setting of hospital inpatient bed assignment. The two most relevant papers are Moallemi et al. (2008) and Veatch (2005), both of which use fluid models to guide the choice of basis functions in the setting of queueing network control. We use a novel combination of a fluid model and an integrated single-pool model to design basis functions, which has several advantages over the pure fluid basis functions; see more discussion in Section 6.3. Moreover, we take the time-varying nature into consideration when developing basis functions, whereas Moallemi et al. (2008) and Veatch (2005) focus on time-stationary settings. Examples of successful applications of ADP in healthcare and service operations include appointment scheduling (Feldman et al. 2014), patient admission control (Samiedaluie et al. 2017), ambulance redeployment (Maxwell et al. 2010, 2013), HIV treatment (Khademi et al. 2015), and call center management (Koole and Pot 2005, Roubos and Bhulai 2010).

3. Model Description

In Section 3.1, we introduce a multiclass, multipool parallel-server queueing system that models patient flows to inpatient wards. This queueing system incorporates time-varying arrivals and a two-time-scale service time feature. In Section 3.2, we specify the MDP framework based on this queueing system. We make several stylized assumptions for the purpose of developing the MDP framework. Later, in Section 7, we relax these restrictive assumptions in an extended simulation model, for example, further differentiating whether patients are admitted from ED or non-ED sources within a specialty (class); using nongeometric, empirical LOS distributions that depend on specialties and admission sources. We demonstrate that our ADP algorithm still produces good overflow policies in the more realistic, extended simulation model.

3.1. Multiclass, Multipool Queueing System for Inpatient Flows

The queueing system has J classes of patients and J parallel servers pools, with each pool dedicated to serve one primary patient class. For simplicity, we assume that pool j serves class j—that is, pool j is the *primary* pool for class j patients, $j \in \{1, 2, \ldots, J\}$. Each server pool j has N_j identical servers. We use $N = \sum_{j=1}^{J} N_j$ to denote the total number of servers in the system.

Upon a class j customer (patient) arrival, if the primary pool has an idle server, the customer is admitted into service immediately; we call this admission a *primary assignment*. Otherwise, the customer waits in buffer j of infinite size. Depending on the overflow decisions, to be specified in Section 3.2, the customer may wait until a primary bed becomes available or, before that, be admitted to a nondedicated pool at a decision epoch; we call the latter admission an *overflow assignment*. Upon a customer departure from pool j, the just-freed server admits a customer from buffer j following a first-come, first-served rule if the buffer is not empty. If buffer j is empty, the server becomes idle until a primary customer arrives, or before that, admits an overflow customer at a decision epoch depending on the overflow decisions. Figure 1 illustrates a five-pool example with $J = 5$.

Time-Varying Arrivals. Each of the J classes of customers arrive to the system following a *time-nonhomogeneous* Poisson process, where the associated arrival rate function $\lambda_j(t)$ is periodic with a period T—that is,

$$\lambda_j(s) = \lambda_j(s + T) \quad \text{for } s \geq 0, \quad j = 1, 2, \ldots, J. \tag{1}$$

For the ease of exposition, we use *day* as the time unit and assume that $T = 1$ and set

$$\Lambda_j = \int_0^T \lambda_j(s)ds,$$

to be the daily arrival rate for class j customers. Figure 4(a), which illustrates the empirical hourly arrival distribution observed at our partner hospital, clearly shows a time-varying pattern.

Two-Time-Scale Service Time. Upon being admitted into service, a customer occupies the server for a random amount of time until departing from the system. We assume that the service time of a customer takes the following *two-time-scale* form:

$$\text{service time} = \text{LOS} + h_{\text{dis}} - h_{\text{adm}}, \tag{2}$$

where LOS denotes the number of midnights a customer spends in the system, and h_{adm} and h_{dis} denote the time-of-day for the customer's admission and discharge time. Shi et al. (2016) show that this two-time-scale service time is a critical feature to capture inpatient flow dynamics, because LOS is mainly driven by the patient's underlying disease, whereas the discharge time h_{dis} is often the results of staff scheduling such as the physician's rounding time. We further assume that (1) the distributions of LOS and h_{dis} are *pool-dependent*; (2) the LOS distribution associated with pool j follows a *geometric* distribution with mean $1/\mu_j$; (3) the distribution of h_{dis} associated with pool j follows a general random distribution characterized by the cumulative distribution function

$$H_j(t) = \mathbb{P}(h_{\text{dis}} \le t),$$

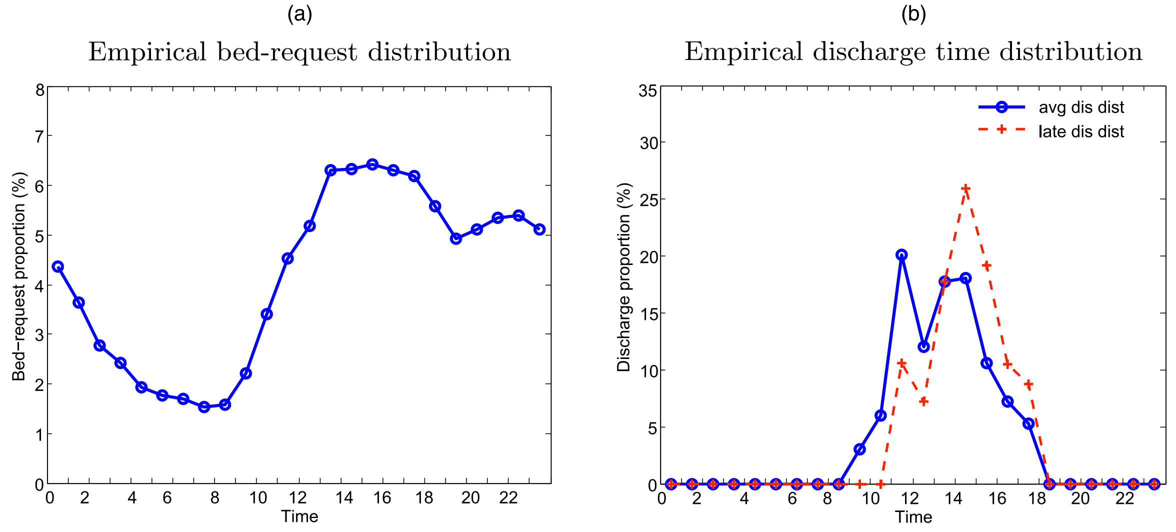
and is independent of the LOS distribution. Figure 4(b) shows two sets of empirical discharge distributions of h_{dis}, using 2010 data from the partner hospital.

The random LOS and h_{dis} create uncertainties in both the *daily and hourly* time scales: (1) the number of patients to be discharged each day is uncertain, and (2) the number of discharges in each hour is uncertain, even when (1) is known. The first uncertainty is captured through the distribution of LOS, while the second uncertainty is captured through the discharge distribution $H_j(\cdot)$. The admission time h_{adm} is internally determined by the system dynamics, and, thus, the conventional independent and identically distributed assumption no longer holds for the service time.

3.2. Overflow Decisions and MDP Formulation

We formulate the overflow decision problem as an *infinite-horizon*, discrete-time average-cost MDP. The decision maker observes the system state $S(\cdot)$ at predetermined decision epochs $t_0, t_1, \ldots$ and takes actions based on the observed states at these epochs. These epochs do not have to be equally spaced, but we assume that they repeat at the same times each day with a total of m epochs per day—for example, decisions are made daily at 3 a.m., 6 a.m., . . . , 5 p.m., and 9 p.m. Note that the queueing system still evolves on a continuous-time basis as described in Section 3.1, and $S(\cdot)$ is a continuous-time stochastic process with possible jumps

Figure 4. (Color online) Empirical Hourly Bed-Request (Arrival) and Discharge Time Distributions



Notes. The distributions are estimated from 2010 year data. For the discharge distributions, the solid curve is estimated from all wards, whereas the dashed curve is estimated from the Card and Gastro wards, which have a later discharge peak than other wards. We set the discharge probability in an hour to be 0 if less than 3% patients discharged in that hour from the data and renormalize the remaining probabilities. Avg, average; dist, distribution.

at the arrival and departure instances or at the decision epochs. In this paper, we use the following conventions: $S(\cdot)$ is assumed to be right continuous having left limits, the action is taken at each epoch t_k, $S(t_k-)$ is the left limit of $S(\cdot)$ at t_k and is called the *preaction state*, and $S(t_k)$ is called the *postaction state*. In other words, the preaction and postaction states used in the MDP are discrete-time samples of the state process $S(\cdot)$ at given epochs t_k- and t_k, respectively. In the next three subsections, we further specify the MDP's state space, action and cost, and state transitions.

3.2.1. State Space.

The system state is a $(2J+1)$-dimensional vector

$$S(\cdot) = (X_1(\cdot), \ldots, X_J(\cdot), Y_1(\cdot), \ldots, Y_J(\cdot), h(\cdot)).$$

Below, we use the postaction state $S(t_k)$ as an example to introduce each coordinate; we can also adapt the explanation for the preaction state $S(t_k-)$.

- $X_j(t_k)$ denotes the *customer count* for pool j, meaning the sum of the number of class j waiting customers and the number of customers in service at pool j at epoch t_k. The customers in service in pool j can be from any customer class, but the waiting customers must be from its primary class j.
- $Y_j(t_k)$ denotes the *to-be-discharged count* for pool j—that is, the number of customers that are ready to be discharged between epoch t_k and the start of the next day. For example, if t_k corresponds to 10 a.m. on a day, then $Y_1(t_k) = 8$ says that there will be eight patients *to be discharged* from pool 1 today between 10 a.m. and midnight. However, because the discharge time h_{dis} is random, we do not know the precise times the eight patients will leave today, except that they will leave after 10 a.m.
- $h(t_k) = k \bmod m$ denotes the epoch index within a day. In other words, $h(t_k) \in \{0, \ldots, m-1\}$ represents the numerical order of epoch t_k within the day that t_k belongs to, among the m total decision epochs. For a general $t \geq 0$, we set $h(t) = h(t_k)$ if $t_{k-1} < t \leq t_k$. Note that because the arrival and discharge time patterns are periodic with 1 day as the period, we still have a time-homogeneous MDP when this epoch index is included in the state space.

We use $\mathcal{S}$ to denote the state space for the MDP and $\mathcal{S}^h$ to denote the subspace of $\mathcal{S}$, which contains all possible states $S(\cdot)$ with the epoch index being h. Because there is no overlap among epochs, we can partition $\mathcal{S}$ into $(\mathcal{S}^0, \ldots, \mathcal{S}^{m-1})$. In this paper, we assume that the to-be-discharged information is available to the decision maker because discharges are usually planned at least 1 day ahead, and bed managers can access such "planned discharge" information to make bed assignments. In section 6.6 of the online supplement of Dai and Shi (2018), we test a setting where the to-be-discharged information is not available. We can still implement our ADP algorithm with slight modification in this setting, and the resulting long-run average cost is around 6% larger than that from the setting with full to-be-discharged information.

3.2.2. Action and Cost.

We denote the set of actions at epoch t_k as

$$f(t_k) = \{f_{ij}(t_k),\ i \neq j,\ i = 1, \ldots, J,\ j = 1, \ldots, J\},$$

where f_{ij} is the number of class i waiting customers assigned to nonprimary pool j. As mentioned in Section 3.1, the primary assignment has a high priority and is immediately done when a primary customer arrives or a primary server becomes available. Thus, the primary assignment is excluded from the decisions at each epoch, and the decision maker has two options: do nothing, or make (some) overflow assignments. For practical purposes, we assume $f_{ij}(t_k) \leq w_{ij}$—that is, each class is allowed to send a maximum number of w_{ij} patients to a nonprimary ward at any decision epoch. We set $w_{ij} = 0$ if pool j is not an overflow ward for class i.

We define the one-epoch cost associated with action $f(t_k)$ and the preaction state $S(t_k-)$ as

$$g(S(t_k-), f(t_k)) = \sum_{i=1}^{J} \sum_{j \neq i, j=1}^{J} B_{ij} \cdot f_{ij}(t_k) + \sum_{j=1}^{J} C_j \cdot Q_j(t_k), \quad (3)$$

where the first part denotes the overflow cost with B_{ij} being the per-patient overflow cost, and the second part denotes the holding cost with C_j being the unit holding cost and $Q_j(\cdot) = (X_j(\cdot) - N_j)^+$ being the number of waiting customers, or, say, queue length. For convenience of analysis, we use the queue length immediately after the action is taken at $t_k, Q_j(t_k)$ as a proxy for the average queue length between two decision epochs, where $Q_j(t_k) = (X_j(t_k) - N_j)^+$ can be recovered via $X_j(t_k-)$ and $f(t_k)$.

Our objective is to find an optimal overflow policy that minimizes the long-run average cost, defined as

$$\lim_{n \to \infty} \frac{1}{n} \mathbb{E}\left(\sum_{k=1}^{n} g(S(t_k-), f(t_k)) \right). \quad (4)$$

See section B.4 in the online supplement for a nonlinear holding cost setting, where patients who wait longer have a more expensive holding cost.

3.2.3. State Transitions.

We denote the preaction state at a given epoch as $s = (x_1, \ldots, x_J, y_1, \ldots, y_J, h)$ and the preaction state after one-step transition as $s' = (x'_1, \ldots, x'_J, y'_1, \ldots, y'_J, h')$. We specify the transition dynamics from s to s' for nonmidnight epochs $h = 1, \ldots, m-1$ here; see section 1.2 of the online supplement in Dai and Shi (2018) for additional details, including the transition probabilities $p(s'|s, f)$.

The state s' is updated as

$$x'_j = x_j + a_j - d_j - \sum_{l \neq j, l=1}^{J} f_{jl} + \sum_{i \neq j, i=1}^{J} f_{ij}, \quad j = 1, \ldots, J, \quad (5)$$

$$y'_j = y_j - d_j, \quad j = 1, \ldots, J, \quad (6)$$

$$h' = (h+1) \mod m, \quad (7)$$

where a_j and d_j denote the arrivals from class j customers and the departures from pool j between h and the next epoch h', respectively, $\sum_{l \neq j, l=1}^{J} f_{jl}$ represents the total number of class j customers assigned to nonprimary pools, and $\sum_{i \neq j, i=1}^{J} f_{ij}$ represents the total number of customers from other classes that are assigned to pool j. Given that d_j patients have been discharged between h and h', the to-be-discharged count at the next epoch h' will be reduced by the amount of d_j, explaining Equation (6); Equations (5) and (7) are straightforward. In the remainder of this paper, we omit the operation "mod m" when referring to a future epoch for h, and use the convention that $h + k = (h + k) \mod m$ for $k = 1, 2, \ldots$.

4. Time-Decomposed Poisson Equation

In this section, we explore the time-periodic properties in the arrivals and discharges and derive a time-decomposed Poisson equation that will aid the design of the ADP algorithm in Section 5.

For the long-run average cost problem, we need to solve the Bellman equation

$$\gamma^* + v^*(s) = \min_f \left\{ g(s, f) + \sum_{s'} p(s'|s, f) v^*(s') \right\}, \quad s \in \mathcal{S}. \quad (8)$$

Here, s denoting the preaction state, γ^* is the optimal (minimal) long-run average cost (per each decision epoch), $v^*(\cdot)$ is the optimal *relative value function*, $g(s, f)$ is the one-epoch cost defined in Equation (3), and $p(s'|s, f)$ is the transition probability. Unless specified otherwise, in this paper s and v^* always refer to the preaction state and the associated relative value function, respectively. We make the following assumption for convenience.

Assumption 1. *A solution pair (γ^*, v^*) exists for Equation (8) and the stationary policy f^* with*

$$f^*(s) = \arg\min_f \left\{ g(s, f) + \sum_{s'} p(s'|s, f) v^*(s') \right\}, \quad s \in \mathcal{S}, \quad (9)$$

achieves the optimal long-run average cost γ^*, namely, f^* is the optimal policy.

See Chapter 8 of Puterman (1994) for conditions under which this assumption holds. We also assume the stability condition

$$\sum_{j=1}^{J} \Lambda_j < \sum_{j=1}^{J} \mu_j N_j \quad (10)$$

holds; otherwise, the queue length will explode, and γ^* is not well defined.

For a given stationary policy f, let $v = \{v(s) : s \in \mathcal{S}\}$ be the corresponding relative value function. Then, the pair (f, v) satisfies the following *Poisson equation*:

$$v = \mathbf{g}_f - \gamma \mathbf{e} + \mathbf{P}_f \boldsymbol{v}, \quad (11)$$

where γ is the long-run average cost associated with policy f, $\mathbf{e}$ is the unit vector, and $\mathbf{g}_f = \{g(s,f) : s \in \mathcal{S}\}$ and $\mathbf{P}_f = \{p(s'|s, f) : s, s' \in \mathcal{S}\}$ are the one-epoch cost vector and the transition probability matrix under policy f, respectively. The optimal pair $(f^*, \boldsymbol{v}^*)$ satisfies Equation (11).

Time Decomposition of the Poisson Equation

The Poisson Equation (11) applies to the entire state space $\mathcal{S}$ and solves the relative value functions for all m epochs at the same time. Now, we show that Equation (11) can be decomposed into m self-contained equations for each subspace $\mathcal{S}^h$, $h = 0, \ldots, m-1$. Define $\boldsymbol{v}^h$ and $\mathbf{g}_f^h$ as the vectors of the relative value function and cost (under the given stationary policy f) for all states in $\mathcal{S}^h$, respectively, and $\mathbf{P}_f^{i,j}$ as the transition matrix for transitions from all states in $\mathcal{S}^i$ to all states in $\mathcal{S}^j$.

Proposition 1. *For each $h = 0, 1, \ldots, m-1$, we have*

$$v^h = (\tilde{\mathbf{g}}_f^h - m\gamma \mathbf{e}) + \tilde{\mathbf{P}}_f^h v^h. \quad (12)$$

Here, $\tilde{\mathbf{P}}_f^h$ and $\tilde{\mathbf{g}}_f^h$ denote the one-*period* transition matrix (from states in epoch h of this period to states in epoch h of the next period) and the one-*period* cumulative cost, respectively—that is,

$$\tilde{\mathbf{P}}_f^h = \mathbf{P}_f^{h,h+1} \mathbf{P}_f^{h+1,h+2} \cdots \mathbf{P}_f^{h-1,h},$$

$$\tilde{\mathbf{g}}_f^h = \mathbf{g}_f^h + \mathbf{P}_f^{h,h+1} \mathbf{g}_f^{h+1} + \cdots + (\mathbf{P}_f^{h,h+1} \cdots \mathbf{P}_f^{h-2,h-1}) \mathbf{g}_f^{h-1},$$

and $m\gamma$ is the long-run average cost over one *period*.

The proof is straightforward by realizing that most $\mathbf{P}_f^{i,j}$ are zeros except those with $j = (i+1) \mod m$, which allows us to write Equation (11) as

$$v^h = \left(\mathbf{g}_f^h - \gamma \mathbf{e} \right) + \mathbf{P}_f^{h,h+1} v^{h+1}, \quad h = 0, 1, \ldots, m-1. \quad (13)$$

See the proof in section 1.3 of the online supplement of Dai and Shi (2018).

Note that the main difference between Equations (13) and (12) is that in Equation (12), the value function on both the left- and right-hand sides is v^h. Thus, Equation (12) is self-contained in the sense that we can solve v^h from it without the need of obtaining the relative value functions for other time epochs. In other words, instead of solving the original Poisson Equation (11) to get $v = (v^0, \ldots, v^{m-1})$ at the same time, we can solve the sub-Equation (12) and get each v^h, separately. This time-decomposition plays an important role in the algorithm we develop in Section 5, by allowing us to use different functional forms to approximate v^h for different h and separately estimate the coefficients associated with the approximation.

5. Simulation-Based ADP: Approximate Policy Iteration

A conventional method to solve Equation (8) is the policy iteration (PI) algorithm; see, for example, section 8.6 in Puterman (1994). Specifically, start from an arbitrary v and $n=1$,

1. Policy improvement: Obtain greedy policy f using

$$f(s)=\arg\min_f\left\{g(s,f)+\sum_{s'}p(s'|s,f)v(s')\right\},\quad \forall s\in\mathcal{S}.$$

2. Policy evaluation: solve the Poisson Equation (11) using policy f generated from above and obtain the updated relative value function v'.
3. Set $v=v'$, increase n by 1, and go to Step 1.

The algorithm stops when f is the same from the current and previous iterations, which achieves the optimal policy f^*. However, steps 1 and 2 in each iteration are notoriously difficult to perform when the state space size $|\mathcal{S}|$ is large. Indeed, even for a five-pool system with just one midnight decision epoch and a truncation at 60 for each X_j $(j=1,\dots,5)$, it is almost infeasible to just store all v, because the state space size is $60^5=777600000$, let alone to solve the Poisson equation.

To address this curse-of-dimensionality, we use feature-based approximations for the relative value functions and conduct simulation-based *approximate policy iteration* (API). Below, we describe the general framework of the ADP algorithm.

Relative Value Function Approximation

To deal with the high-dimensional state space, we approximate the relative value function with a linear combination of a finite set of basis functions, or, say, features. To capture the time dependency, we approximate the relative value functions separately for each epoch. That is, for each $h=0,\dots,m-1$,

$$v^h(s)\approx\phi^h(s)\beta^h=\sum_{i=1}^{K^h}\beta_i^h\phi_i^h(x,y),\qquad \forall s\in\mathcal{S}^h. \tag{14}$$

Here, $\phi^h(s)=(\phi_1^h(x,y),\dots,\phi_{K^h}^h(x,y))$ is a set of K^h basis functions, and $\beta^h=(\beta_1^h,\dots,\beta_{K^h}^h)'$ is the associated coefficient vector. The specific form of the basis functions, the coefficients, and the number of basis functions K^h can all be time-dependent, and we use the superscript h to emphasize the time-dependency. The basis functions we choose in this paper is a combination of quadratic functions in x and y, and a time-dependent value function from an integrated single-pool system (see Section 6).

Simulation-Based API

Using Equation (14), the approximate relative value function is completely determined by the coefficient vector β^h and the prechosen basis functions. The following approximate policy iteration iteratively updates the coefficient vectors. Start from an arbitrary coefficient vector $(\beta^0,\dots,\beta^{m-1})$ and set $n=1$.

1. Policy improvement: simulate the J-pool system using improved policy $f^{(n)}$, where the action at each decision epoch is generated according to

$$f^{(n)}(s)=\arg\min_f\Big\{g(s,f)+\sum_{s'\in\mathcal{S}^{h+1}}p(s'|s,f)\,\phi^{h+1}(s')\,\beta^{h+1}\Big\},\quad s\in\mathcal{S}^h. \tag{15}$$

2. Policy evaluation: *approximately* solve the h sub-equations in Equation (12) under the improved policy $f^{(n)}$, and get an updated set of coefficients $(\hat\beta^0,\dots,\hat\beta^{m-1})$. In this paper, we use *temporal difference learning* (TD learning) with simulation samples collected in step 1 to perform this update mechanism; see online supplement, section A.1 for details.
3. Set $(\beta^0,\dots,\beta^{m-1})=(\hat\beta^0,\dots,\hat\beta^{m-1})$, increase n by 1, and go to step 1.

The algorithm stops when n equals a predetermined number $n^*>1$, the total number of iterations. Note that in step 1, we do *not* store $v(s)$ or the action rule $f^{(n)}(s)$ for every state $s\in\mathcal{S}$. Instead, we obtain the action $f^{(n)}(s)$ *on-the-go* from Equation (15) whenever a state s is visited during the simulation.

Remarks. The success of the ADP algorithm depends on (1) choosing suitable basis functions to approximate v^h and (2) iteratively updating the coefficients associated with the basis functions. For (1), we leave the thorough justification of the basis function choice to Section 6. For (2), to deal with the time dependency, we modify the standard TD learning procedure to approximately solve the time-decomposed Poisson Equation (12); see online supplement, section A.1.

It is worth pointing out that our ADP algorithm is scalable in the system size. The complexities of step 1 (policy improvement) and 2 (policy evaluation) are both $O(MJ^2)$, where M is the number of simulated days in Step 1, and J is the number of pools. See section 2 of the online supplement of Dai and Shi (2018) for a detailed complexity analysis.

6. Basis Functions

To derive the basis functions, we study a special *midnight MDP* to gain insights into its relative value function and to support our choice of the midnight basis functions. Once the basis functions for the midnight epoch are chosen, we derive those for other epochs via a backward induction.

Specifically, the midnight MDP we consider is a special case of the MDP model developed in Section 3 by setting $m = 1$, with the only decision epoch every day being set at the midnight epoch. Because we generate new discharges at each midnight (following from the two-time-scale service time setting introduced in Section 3.1), it is easy to check that this MDP is time-homogeneous and the customer count $x = (x_1, \ldots, x_J)$ is enough to capture the state. We use $v_{\text{mid}}(x)$ to denote the associated optimal relative value function, and propose the following approximation:

$$v_{\text{mid}}(x) \approx v_1^F(x) + V_s\big(\sum_j x_j\big). \tag{16}$$

Here, v_1^F is obtained from a fluid control problem and approximates the *overflow cost* part in $v_{\text{mid}}(x)$, whereas $V_s(\cdot)$ denotes the relative value function from an *integrated single-pool* system and approximates the *holding cost* part in $v_{\text{mid}}(x)$. Below, in Sections 6.1 and 6.2, we specify the basis functions for different epochs motivated from approximation (16). In Section 6.3, we explain the rationale for approximation (16). In Section 6.4, we compare the long-run average costs from using our proposed basis functions and those from exact analysis in a two-pool setting.

6.1. Basis Functions for the Midnight Epoch

For a general multiepoch MDP with $m > 1$, at the midnight epoch $h = 0$, the to-be-discharged counts from the past day are zero because all patients should have been discharged by that time, and the new to-be-discharged counts for the next day are generated based on the customer counts $x = (x_1, \ldots, x_J)$ at this midnight epoch. Thus, the midnight relative value function, including the optimal one $v^{0,*}$, only depends on x, which is similar to v_{mid} in the sense that both value functions depend on x only. Though $v^{0,*}$ does not necessarily equal to v_{mid} except when $m = 1$, we assume that they have a similar structure and can be approximated by Equation (16).

To facilitate finding basis functions for other epochs, we need an even more parsimonious form to approximate $v^{0,*}$ than Equation (16). The reason is that if we directly use Equation (16), to obtain the value functions for other epochs via a backward induction, we need to prestore not only $v_1^F(x)$ for all possible x, but also the approximate value function for each epoch and for each possible state. In other words, we will meet the same computational challenge for conventional value iteration. Based on the piecewise linear structure of $v_1^F(x)$, we instead propose the following basis functions for the midnight epoch:

$$V_s\big(\sum x\big), x_j^2, x_j, \quad j = 1, \ldots, J, \tag{17}$$

and, correspondingly, the parsimonious approximation

$$v^{0,*}(x) \approx \beta_s^0 V_s\Big(\sum_j x_j\Big) + \sum_{j=1}^{J} (\beta_{2j}^0 x_j^2 + \beta_{1j}^0 x_j) + \beta_{00}^0. \tag{18}$$

We provide more justifications on the choice of these basis functions in section 5.1 of the online supplement of Dai and Shi (2018). In particular, we show that starting from approximation (18) and performing the backward induction m times (for the m epochs of a day), we get an "updated" midnight value function. This updated value function has the same form as Equation (18) if the optimal policy at midnight is work-conserving, thus providing some consistency for using Equation (18).

6.2. Basis Functions for Different Time Epochs

For epoch $m - 1$, the optimal relative value function satisfies

$$\gamma + v^{m-1}(s) = \min_f \Big\{ g(s, f) + \sum_{s'} p(s' | s, f) v^0(s') \Big\}, \quad s \in \mathcal{S}^{m-1}, \tag{19}$$

where v^0 is the optimal value function for the midnight epoch, the epoch following epoch $m - 1$. For notational convenience, we omit the * symbol from the optimal relative valuation function.

We plug Equation (18) into Equation (19) to get an approximate form for $v^{m-1}(s)$. By repeating the procedure, we derive the approximate form of v^h for each epoch $h = m - 2, \ldots, 1$, respectively, using the approximate form of v^{h+1}; see Section B.4 of the online supplement for detailed algebra. These approximate forms suggest the following basis functions:

$$V_s^h(x, y), x_j^2, x_j, y_j^2, y_j, x_j y_j, \quad j = 1, \ldots, J, \tag{20}$$

for epoch $h = m - 1, \ldots, 1$, where

$$V_s^h(x, y) = \sum_{i=0}^{\infty} \mathbb{P}(A^h = i) V_s\Big(\sum_j x_j + i - \sum_j y_j\Big), \tag{21}$$

is the expectation over the one-dimensional single-pool value function $V_s(\cdot)$ with respect to the random variable A^h, and A^h follows a Poisson distribution with mean $\sum_j (\int_{t_h}^1 \lambda_j(s) ds)$. In other words, A^h denotes the total number of arrivals from all J classes that arrive between the current epoch h and the most recent midnight epoch. One thing worth mentioning is that, using our proposed basis functions (20), the cost-to-go function $\sum_{s' \in \mathcal{S}^{h+1}} p(s' | s, f) \phi^{h+1}(s') \beta^{h+1}$ in Equation (15) can be calculated efficiently for any given action f, which allows us to obtain the optimal action quickly in step 1 of the ADP algorithm. In particular, the one-dimensional property of $V_s(\cdot)$ allows an easy evaluation of the expectation; see online supplement, section A.2 for an example.

6.3. Rationale for Approximation (16)

In this section, we focus on the special midnight MDP with $m=1$ and explain the rationale for approximation (16). We assume that the customers are homogeneous in the sense that the discharge rate $\mu_j=\mu$ and the unit holding cost $C_j=C$ for $j=1,\ldots,J$. We also assume that the state space is finite (by a state truncation of the original MDP) and the Markov chain generated from the optimal policy f^* is irreducible, such that the following representation for the relative value function $v_{\text{mid}}(x)$ is well defined:

$$v_{\text{mid}}(x)=\mathbb{E}_x\left[\sum_{k=0}^{\infty}\left(\sum_{i,j}B_{ij}f^*_{ij}-\gamma^*_v\right)\right]+\mathbb{E}_x\left[\sum_{k=0}^{\infty}\left(\sum_{j=1}^{J}C(X_j(k)-N_j)^+-\gamma^*_q\right)\right],\tag{22}$$

where γ^*_q and γ^*_v are the optimal long-run average holding cost and overflow cost, respectively, with $\gamma^*=\gamma_q+\gamma_v$, and $\mathbb{E}_x$ denotes the expectation conditional on $X_j(0)=x_j$ for $j=1,\ldots,J$. See Puterman (1994) for details. Note that the assumptions of homogeneous customers and finite state space are only made here to justify Approximation (16); our modeling framework and developed ADP algorithm do not need these assumptions.

Comparing Equation (22) with Equation (16) shows that the approximation is based on

$$\mathbb{E}_x\left[\sum_{k=0}^{\infty}\left(\sum_{i,j}B_{ij}f^*_{ij}-\gamma^*_v\right)\right]\approx v_1^F(x),\tag{23}$$

$$\mathbb{E}_x\left[\sum_{k=0}^{\infty}\left(\sum_{j=1}^{J}C(X_j(k)-N_j)^+-\gamma^*_q\right)\right]\approx V_s\Big(\sum_j x_j\Big).\tag{24}$$

The explanations, first for Equation (24) and then for Equation (23), are as follows.

Work-Conserving Policy and Single-Pool System. We say a policy f is a work-conserving policy if

$$\sum_{j=1}^{J}(X_j^f(k)-N_j)^+=\left(\sum_{j=1}^{J}X_j^f(k)-\sum_{j=1}^{J}N_j\right)^+,\quad k=0,1,\ldots,\tag{25}$$

where $X_j^f(k)$ denotes the postaction state under policy f at midnight of day k. In other words, after taking the action at each midnight, no server should be idle when any of the J pools has a positive queue length. Condition (25) also requires the system to have *full connectivity* between the customer classes and servers—that is, a customer can be assigned to any of the pools via either primary or overflow assignments. Full connectivity is more restrictive than the conventional complete resource pooling condition for heavy traffic analysis; see, for example, Bell and Williams (2005). Under a work-conserving policy f, the J-pool system operates as an integrated *single-pool* system. The following proposition says that we can use the relative value function of this single-pool system, $V_s(\cdot)$, to capture the holding cost in the J-pool system.

Proposition 2. *Assume f is a policy that satisfies Equation (25). We have*

$$\mathbb{E}_x\left[\sum_{k=0}^{\infty}\left(\sum_{j=1}^{J}C(X_j^f(k)-N_j)^+-\gamma_q^f\right)\right]=V_s\Big(\sum_j x_j\Big).\tag{26}$$

Here, $V_s(\cdot)$ denotes the relative value function of a single-pool system using the same unit holding cost C, and we show it has a similar representation as Equation (22). This single-pool system has a total of $N=\sum_j N_j$ servers, serving a single class of customers whose daily arrival follows a Poisson distribution with mean $\sum_j\Lambda_j$; each customer in service is discharged with probability μ each day. The proof is based on a coupling argument between the J-pool system and the single-pool system; see section 1.4 of the online supplement in Dai and Shi (2018) for a sketch of the proof.

Note that the left side of Equation (26) is not necessarily equal to that of Equation (24), because the latter corresponds to the holding cost under the optimal policy f^*. Approximation (24) acquires an equality when f^* is indeed work-conserving. Our numerical experiments suggest that when B is not significantly larger than C, f^* is work-conserving or close to work-conserving in the midnight MDP. This observation is also consistent with the current practice in our partner hospital—that is, aggressive overflow during the night and early morning; see discussions in Section 1.2.

Fluid Model for the Overflow Cost. To get $v_1^F(x)$ that is specified below in Equation (30), we consider the fluid model associated with the J-pool system and an optimal fluid control problem to minimize the long-run average cost, defined as

$$\lim_{n\to\infty}\frac{1}{n}\sum_{k=0}^{n}\left(\sum_{i,j}B_{ij}\bar{f}_{ij}(k)+\sum_{j=1}^{J}C(\bar{x}_j(k)-N_j)^+\right),\tag{27}$$

where $\bar{x}(k)=(\bar{x}_1(k),\ldots,\bar{x}_J(k))$ denotes the post-action fluid customer count at midnight of each day k, and $\bar{f}(k)=\{\bar{f}_{ij}(k)\}$ denotes the fluid action and is defined in a similar way as f in Section 3.2.2, except that $\bar{f}_{ij}$ values can be nonintegers. This fluid control problem can be viewed as the *deterministic* version of the midnight MDP. Similar to Equation (22), we represent the optimal relative value function in this fluid control problem as

$$\nu^F(x) = \sum_{k=0}^{\infty}\Big(\sum_{i,j} B_{ij}\bar{f}^*_{ij}(k) - \bar{\gamma}^*_v\Big) + \sum_{k=0}^{\infty}\Big(\sum_{j=1}^{J} C(\bar{x}^*_j(k) - N_j)^+ - \bar{\gamma}^*_q\Big), \tag{28}$$

where $\{\bar{f}^*_{ij}(k)\}$ values are the optimal actions solved from Equation (27), $\{\bar{x}^*_j(k)\}$ values form the optimal system "path" when starting from $\bar{x}(0) = x$ and implementing the optimal fluid actions, and $\bar{\gamma}^*_v$ and $\bar{\gamma}^*_q$ denote the optimal long-run average (fluid) holding and overflow costs, respectively. We can efficiently solve $\nu^F(x)$ by a linear programming formulation. See section 3 of the online supplement of Dai and Shi (2018) for details, including the transition dynamics for the fluid model.

Similar to the results for a certain class of queueing networks, one can prove

$$\lim_{||x||\to\infty} \frac{|\nu^F(x) - \nu_{\text{mid}}(x)|}{||x||^2} = 0, \tag{29}$$

see, for example, theorem 7 in Meyn (2000). In other words, ν^F serves as a good approximation for ν_{mid} when the system starts from a more congested state. Comparing Equation (28) with Equation (22), the first part of ν^F, denoted as

$$\nu^F_1(x) = \sum_{k=0}^{\infty}\Big(\sum_{i,j} B_{ij}\bar{f}^*_{ij}(k) - \bar{\gamma}^*_v\Big), \tag{30}$$

provides an approximation to the overflow cost part in ν_{mid}. This is why we propose Equation (23).

Comparison with Fluid Basis Functions. One can directly use ν^F, instead of Equation (16), to approximate ν_{mid}. However, this approximation is quite rough. In particular, when $\rho_j = \Lambda_j/(\mu_j N_j) < 1$ and the starting state satisfies $x_j < N_j$ for all j, the fluid holding and overflow cost are both 0, and thus, $\nu^F(x) = 0$ for all such x, which is obviously not the case for $v_{\text{mid}}(x)$. Indeed, our approximation (16) is a refinement to ν^F by replacing its second part, $\Sigma_{k=0}^{\infty}(\Sigma_{j=1}^{J} C(\bar{x}^*_j(k) - N_j)^+ - \bar{\gamma}^*_q)$, with $V_s(\Sigma_j x_j)$. See Figures 1 and 2 in the online appendix for examples of the remarkably good approximation quality of Equation (16) in a two-pool setting. More importantly, we use Equation (16) to guide the choice of basis functions. These basis functions require fewer coefficients to be estimated comparing to "fluid" basis functions developed from ν^F as suggested by Moallemi et al. (2008) and Veatch (2005). Following their approach, the basis functions are quadratic functions with all the interaction terms and the size is $O(J^2)$, whereas the size of our proposed basis functions is $O(J)$. Moreover, we find that using the fluid basis functions often leads to cost and policy oscillation—that is, the chattering issue mentioned in Bertsekas (2012, section 6.4), whereas using our proposed basis functions tends to have a more stable performance; see section 6.1 of the online supplement of Dai and Shi (2018) for an example.

6.4. Performance of Basis Functions in a Two-Pool Setting

To demonstrate the accuracy of the ADP algorithm, ideally we would like to compare the long-run average cost from the ADP algorithm with the "true" optimal value solved from the conventional policy iteration. Because of the curse-of-dimensionality for conventional PI, we can only perform such comparison in a two-pool setting; for a general multipool setting, we compare the ADP policy with a few benchmark policies in Section 7. Considering that our ADP is simulation based, we run simulation experiments (with a common underlying random number generator) to compare the long-run average costs.

Table 1 reports performance of the ADP algorithm in a two-pool, eight-epoch setting, where the homogeneous customer assumption used to justify approximation (16) holds. For comparison purposes, we report performance from using our proposed basis functions and that from using a set of naive functions $(x_i, y_i, x_i y_i)$, $i = 1, 2$. We also conduct experiments in nonhomogeneous settings where μ, B, or C are different for the two classes of patients; see section 5.2 of the online supplement in Dai and Shi (2018). These numerical results show that the ADP algorithm, along with the proposed basis functions, produces overflow policies that lead to near-optimal performance. The optimality gap, defined as $\frac{\text{ADP cost} - \text{optimal cost}}{\text{optimal cost}}$, is less than 5% among all tested experiments.

7. Numerical Results for a Five-Pool Model

In this section, we evaluate the ADP algorithm in a five-pool simulation model populated with parameters estimated from real hospital data. We refer to this simulation model as the *extended model* because it incorporates more realistic features than the *basic model* we introduced in Section 3. In Section 7.1, we specify this extended simulation model. In Section 7.2, we compare the long-run average costs from the ADP policy and three naive policies under a variety of system conditions. We also summarize insights gained from the numerical results there.

7.1. Extended Simulation Model

The extended model has five parallel server pools, serving patients from five medical specialties: General

Table 1. Homogeneous Two-Pool MDP: Eight Decision Epochs

	$B = 6$	$B = 12$	$B = 30$
Using optimal value functions	22.41 ± 0.08	27.02 ± 0.13	37.21 ± 0.08
ADP (proposed basis) optimality gap	22.78 ± 0.06; (2%)	27.57 ± 0.12; (2%)	38.70 ± 0.06; (6%)
ADP (naive basis) optimality gap	23.94 ± 0.12; (7%)	29.86 ± 0.13; (11%)	41.48 ± 0.14; (11%)

Notes. We set $N_1 = 28$, $N_2 = 32$, $\Lambda_1 = \Lambda_2 = 6.25$, $\mu_1 = \mu_2 = 1/4$, $C_1 = C_2 = 3.0$, and $B = B_{12} = B_{21}$. The number after the ± sign is the half-width of the corresponding 95% confidence interval. Numbers following the semi-colons in second and third rows are the corresponding optimality gap.

Medicine (GeMed), Surgery (Surg), Orthopedic (Ortho), Cardiology (Card), and Other Medicine (OtMed), with the last one including Gastroenterology and Neurology. The five chosen specialties account for 75% of the total inpatient admissions and 85% of admissions from ED. We exclude three specialties (Oncology, Renal, and Respiratory) from the extended model because of their small volume and because their patients usually require specialized rooms and equipment. Below, we specify (1) features of the extended model that deviate from those in the basic model; (2) parameter estimation; and (3) model validation. Note that the basic and extended models are essentially two different (though closely connected) models, whereas the ADP algorithm is developed under the framework of the basic model. To obtain an “ADP policy” for the extended model from the ADP algorithm, we need to establish certain mapping between the basic and extended models; see online appendix, section B.3 for details.

Features of the Extended Model. Within each of the five specialties, we further differentiate patients by their admission sources and classify them into: (1) ED patients—patients admitted from ED, (2) EL patients—elective admissions, and (3) TR patients—internal transfers from other hospital units, such as ICU or operating rooms. For each specialty, we generate arrivals from each admission source separately and use three separate buffers, unlike the basic model, to hold waiting patients from the three sources—that is, the extended model has 15 arrival streams and 15 buffers. We enforce a priority rule for the primary bed assignment: when a primary bed becomes available, EL patients have the highest priority, followed by ED and then TR patients. This priority setting is same as that in Powell et al. (2011) and Shi et al. (2016); see the latter for the rationale. We have also tested other priority settings such as giving TR patients higher priority than ED patients, and found that the ADP performance is not very sensitive to the priority settings. We specify the overflow assignment in online appendix, section B.3. Note that hospital managers may not strictly follow one particular priority rule in practice. Our modeling framework and ADP algorithm are meant for decision support, and are flexible to allow “overruling” if managers decide to use a different priority setting. If such an overruling happens, the model simply moves to the next decision epoch by updating states using the actual decision and realized arrivals and discharges.

Empirical study shows that the arrival processes of ED patients follow nonhomogeneous Poisson processes, but not for EL or TR patients (Shi et al. 2014). Thus, in the extended model we generate the arrivals of EL or TR patients by using a two-step procedure: (1) generate a total number of A_k arrivals from an empirical distribution on day k, and (2) generate an arrival time on day k for each of these A_k patients according to an empirical bed-request time distribution. The service time of each patient follows Equation (2); however, we allow the LOS distributions to be both *specialty*- and *admission-source*-dependent, relaxing the pool-dependent assumptions made in the basic model. Note that in the baseline scenario, we still assume that the LOS distributions are geometric; in Section 7.2, we demonstrate results from using empirical LOS distributions. All other modeling features not mentioned here remain the same as in the basic model.

Parameter Estimation. We populate the extended simulation model with a 2010 data set from our partner

Table 2. Parameter Settings on the Daily-Scale for the Five-Pool Model

	GeMed			Surg			Ortho			Card			OtMed			Total
Λ_j	16.92	0.31	0.66	10.76	4.39	5.07	7.94	2.34	2.61	10.99	3.87	4.39	11.75	1.75	1.21	84.96
Average LOS	5.24	5.47	5.94	3.25	4.71	5.71	4.65	6.15	5.03	4.01	4.15	4.36	3.86	3.69	5.36	4.48
N_j		85			104			115			67			54		425
ρ_j		111%			81%			56%			118%			108%		90%

Notes. We use a 2010 data set, and the empirical statistics are documented in Shi et al. (2014). In the first two rows (Λ_j and average LOS), each of the three entries reports the averages for ED, EL, and TR patients, respectively. In the last row, ρ_j denotes the implied utilization, defined as $\Lambda_j/(\bar{\mu}_j N_j)$, where $1/\bar{\mu}_j$ is the average LOS across all admission sources for specialty j.

hospital. Table 2 reports the daily-scale parameters. For N_j, because the number of beds were changing during 2010, we use the average number of beds allocated for each specialty and fine tune the values to match the simulation output with various empirical performance. Note that the actual hospital bed configuration was suboptimal—certain specialties (e.g., GeMed and Card) were allocated with insufficient beds, and their implied utilization values exceed 100%. In Section 7.2, we test a hypothetical "balanced" setting, where the bed configuration is done reasonably well, with the implied utilization of each server pool being $<100\%$.

For the hourly scale parameters, we estimate the hourly arrival proportions $\lambda_j(t)/\Lambda_j$ from the corresponding empirical patterns for each admission source. Because the five specialties have similar hourly arrival patterns, we pool the data to estimate $\lambda_j(t)/\Lambda_j$ for each class j. Thus, the five customer classes have "synchronized" arrival patterns. Figure 4(a) illustrates the ED arrival pattern; online supplement, Section B.5 shows the EL and TR arrival patterns (as well as the daily arrival distributions). However, the discharge patterns are not synchronized. As suggested from the data, we use a late discharge time distribution h_{dis} for customers departed from the Card and OtMed server pools (see the dashed curve in Figure 4(b)) and use the average discharge time distribution for all other pools (see the solid curve in Figure 4(b)).

Model Validation and Computational Settings. We perform a detailed model validation in online appendix, section B.2. We show that this extended simulation model, populated with the parameters described above, produces performance curves that are close to the empirical ones. Computational settings for the simulation experiments are detailed in online appendix, section B.1.

7.2. Comparing the ADP Policy with Naive Policies

In this section, we compare the ADP policy against three naive policies—empirical, full-sharing, and midnight policies—introduced in Section 1.2. We focus on the average cost comparison using simulation experiments. We consider an eight-epoch setting where the overflow decision is made every three hours throughout each day. We assume that the unit holding cost $C_j = 6.0$ is the same for each class j and that the overflow cost for each class follows the same cost scheme, where the cost of assigning a patient to a preferred overflow ward is five less than that to a secondary overflow ward. This difference in the overflow costs reflects the preference for using the preferred overflow ward. We use the cost pair—for example, $B = (15, 20)$—to denote the overflow cost to the preferred and secondary wards, respectively, across all classes, and we vary the cost pair from $(15, 20)$ to $(40, 45)$ in the experiments. Our simulation model and algorithm definitely allow nonhomogeneous holding costs or overflow cost pairs; see section 6.6 of the online supplement of Dai and Shi 2018 for additional experiments.

Table 3 summarizes the long-run average daily costs under different overflow cost pairs. The results reported in the introduction correspond to the *baseline scenario* with $B = (30, 35)$. The ADP policy generally outperforms the three naive policies by 10%–20%, sometimes by more than 30%. We also observe that when the overflow costs are lower, the ADP policy behaves more like the full-sharing policy, whereas when the overflow costs are higher, the ADP policy behaves more like the empirical or midnight policy. These observations are expected because it is beneficial to pool beds when the cost of pooling is cheaper, and vice versa. We observe similar phenomena when using nonhomogeneous holding costs and overflow cost pairs with larger or smaller gaps.

Table 4 reports the long-run average costs of the four policies under a variety of system conditions. Below, we discuss the findings and our insights.

Impact of System Load. We observe that when the system load increases, the benefit gained from the ADP policy, measured by the relative difference in the long-run average cost, decreases. To explain this observation, recall that the naive policies perform a full resource

Table 3. Long-Run Average Daily Costs of the ADP Policy and Three Naive Policies

$C = 6.0$	ADP policy	Empirical	Full-sharing	Midnight
$B = (15, 20)$	231.29	290.05 (25.4%)	259.96 (12.4%)	436.92 (88.9%)
$B = (20, 25)$	284.34	347.09 (22.1%)	325.58 (14.5%)	482.24 (69.6%)
$B = (25, 30)$	337.35	404.13 (19.8%)	391.20 (16.0%)	527.57 (56.4%)
$B = (30, 35)$	390.36	461.16 (18.1%)	456.82 (17.0%)	572.90 (46.8%)
$B = (35, 40)$	443.30	518.20 (16.9%)	522.45 (17.9%)	618.22 (39.5%)
$B = (40, 45)$	496.23	575.23 (15.9%)	588.07 (18.5%)	663.55 (33.7%)

Notes. Number inside parentheses in columns 2–4 denotes the relative difference in the long-run average cost between the ADP policy and the column's naive policy. The half-width of the 95% confidence interval for the reported average cost is between 0.1 and 0.5 and is omitted from the table.

sharing at least once a day (e.g., at the midnight epoch for the midnight policy). Thus, the ADP policy mainly gains its advantage by using better overflow decisions during the day. However, in a more heavily loaded system, fewer beds are available during the day and, thus, less opportunity for overflow assignments. In addition, in a more heavily loaded system, the overall congestion level is mainly driven by *daily-scale* parameters—for example, average LOS and capacity—whereas overflow decisions made during the day mainly affect *hourly* performance and have less impact on overall congestion; see Dai and Shi (2017) for a more rigorous argument on this daily–hourly difference using a two-time-scale framework. Correspondingly, when the system load decreases, we observe a larger benefit gained from the ADP policy. This benefit reaches the largest when the system is around 84% utilized, where the relative difference between the ADP policy and the best-performed naive policy increases to 28%. This benefit starts to decrease as the system load further decreases, because overflow assignments occur less often when each pool has more capacity to handle its patient demand. When each pool has sufficiently low utilization ($<75\%$ in our experiments), all reasonable overflow policies perform similarly because overflow is rarely needed; see section 6.6 of the online supplement for the details.

Impact of Bed Allocations. Comparing to the baseline setting, we observe a smaller benefit gained from the ADP policy in the balanced setting. The reason is that when the bed allocation is done more appropriately with each pool being able to serve its patients without overflow, fewer overflow assignments are needed, especially for specialties which have $>100\%$ utilization in the baseline setting—for example, GeMed and Card. Indeed, in a perfectly balanced setting, where each pool has the same utilization (90%), the relative difference between the ADP policy and the empirical policy (the best performer of the naive policies) reduces to only 6%.

Impact of LOS Distributions. In one set of experiments, we change the geometric LOS distributions to empirical distributions; see online appendix, section B.5. To implement the ADP algorithm in this system with empirical LOS distributions, we generate actions from Equation (15) and pretend that the LOS distributions are geometric—that is, we use the same transition probabilities $p(s'|s,f)$ as in the geometric setting to evaluate the cost-to-go function. The coefficients used in the algorithm are trained in the corresponding setting with geometric LOS distributions. Referring again to Table 4, the last row shows that the ADP still outperforms the naive policies, with a similar magnitude of improvements as we observed in the geometric setting (baseline scenario). A potential reason for this good performance is that the empirical LOS distributions are not far from geometric distributions, especially on the tail side; see the plots in online appendix, section B.5 for an example. Caution should be exercised when applying our ADP algorithm to other hospital settings with different nongeometric LOS distributions. We leave to future research to comprehensively examine the performance of our ADP under other LOS distributions and to improve the modeling framework and algorithm to accommodate general LOS distributions.

Other Sensitivity Analyses. We have performed additional sensitivity analyses, including using different discharge time distributions or priority settings and approximating gender effect by scaling down the system size by half. The ADP performance is robust in the sense that it gains a similar magnitude of benefit over the naive policies, as we see in this section, and our main insights remain similar; see section 6.6 of the online supplement for more details.

8. Conclusions and Future Work

In this paper, we model hospital inpatient flow as a multiclass, multipool parallel queueing network and

Table 4. Long-Run Average Daily Costs of the ADP Policy and Other Naive Policies

Same $C = 6.0$	ADP policy	Empirical	Full-sharing	Midnight
Baseline	390.36	461.16 (18.1%)	456.82 (17.0%)	572.90 (46.8%)
Increase load	471.99	534.44 (13.2%)	532.23 (12.8%)	645.89 (36.8%)
Decrease load	340.55	412.41 (21.1%)	407.51 (19.7%)	518.81 (52.3%)
Balanced allocation	166.77	188.09 (12.8%)	216.94 (30.1%)	199.73 (19.8%)
Empirical LOS	389.63	460.35 (18.2%)	455.98 (17.0%)	571.97 (46.8%)

Notes. The four scenarios reported in rows 2–5 use the same parameter setting as in the baseline scenario except one of the following: (1) increasing load with $\{N_j\} = \{83, 102, 111, 66, 53\}$; (2) decreasing load with $\{N_j\} = \{87, 106, 119, 68, 55\}$; (3) more balanced bed allocation with $\{N_j\} = \{103, 97, 78, 82, 65\}$; (4) using empirical LOS distributions for patients of each specialty and admission source. Number inside parentheses in columns 2–4 denotes the relative difference in the long-run average cost between the ADP policy and the column's naive policy. The half-width of the 95% confidence interval for the reported average cost is between 0.15 and 0.5 and is omitted from the table.

formulate the inpatient overflow decision problem as a discrete-time, infinite-horizon average cost MDP. The MDP explicitly incorporates time-varying features into both the patient arrival and discharge processes. To tackle the curse-of-dimensionality, we develop a simulation-based ADP algorithm, where the relative value functions are approximated by carefully selected basis functions. We demonstrate, via extensive numerical experiments, that the ADP algorithm is efficient in finding good overflow policies in relatively realistic settings and can help hospital managers to devise operational strategies to achieve the desired performance. Sensitivity analyses also suggest that the ADP algorithm obtains larger benefit when a hospital is moderately loaded and bed allocation is not perfectly balanced.

The work presented in this paper has several limitations. First, because of the complexity of hospital operations, our developed queueing models, even the extended simulation model, cannot fully replicate the real system. For example, a patient may be transferred back to a primary ward if initially being assigned to a nonprimary one. Although such transfers are not frequent ($<20\%$ among overflow patients) in our partner hospital, they have been promoted in some hospitals (Thompson et al. 2009). Our model does not capture transfers between general wards and other hospital units such as ICU (Helm and Oyen 2014); activities in these units could impact patients' stay in the general wards. We assume discrete decision epochs, whereas real bed assignments may occur in a more dynamic fashion. Nevertheless, the main purpose of this paper is to develop a sophisticated algorithm for a challenging overflow problem, where the underlying models are highly relevant to hospital operations. We leave for future research to make the queueing models more realistic and to adapt the ADP algorithm to new models. In particular, explicitly incorporating general LOS distributions and subclasses within a medical specialty (to capture patient gender, admission source, etc.) in the modeling framework could potentially give better decisions. We would also like to alert readers that the five-pool simulation model extensively used in this paper is populated with data from one particular hospital. Different hospitals may have different characteristics—for example, LOS distributions could be very different from geometric distributions, and patient–bed configuration could be different from that in Figure 1. Implementing the ADP algorithm in a real hospital environment needs substantial efforts and "customization" in calibrating the decision models.

Second, we demonstrate the performance of the ADP algorithm mainly through numerical experiments. An important future work is to establish performance bounds on this algorithm. Because our setting is novel (long-run average cost problem with time-varying arrival and departure patterns), one would need to develop new methodologies to establish such bounds, potentially based on existing frameworks. One possibility is to use the information relaxation framework established in Brown et al. (2010) and Brown and Haugh (2017), which has gained success in various finite- and infinite-horizon discounted cost settings.

Third, although in this paper we mainly use B_{ij} and C_j as tuning parameters, estimating these cost parameters is a nontrivial task and remains a viable topic in empirical research. Thompson et al. (2009) propose a method to interactively learn the cost structures by letting the decision makers to make choices under hypothetical scenarios. Park et al. (2019) use the ED physician's behavior—when to take a new patient from the waiting room—to identify the holding cost; a similar technique could be borrowed to our inpatient overflow setting.

Acknowledgments
The authors thank the associate editor and two anonymous reviewers for constructive comments to improve this paper. The authors thank Anton Braverman, Jiekun Feng, and Shuangchi He for improving the exposition of the paper.

References
Adelman D, Mersereau AJ (2008) Relaxations of weakly coupled stochastic dynamic programs. *Oper. Res.* 56(3):712–727.

Armony M, Israelit S, Mandelbaum A, Marmor Y, Tseytlin Y, Yom-Tov G (2015) Patient flow in hospitals: A data-based queueing perspective. *Stochastic Systems* 5(1):146–194.

Armony M, Ward AR (2010) Fair dynamic routing in large-scale heterogeneous-server systems. *Oper. Res.* 58(3):624–637.

Baron O, Berman O, Krass D, Wang J (2017) Strategic idleness and dynamic scheduling in an open-shop service network: Case study and analysis. *Manufacturing Service Oper. Management* 19(1):52–71.

Bell SL, Williams RJ (2005) Dynamic scheduling of a parallel server system in heavy traffic with complete resource pooling: Asymptotic optimality of a threshold policy. *Electronic J. Probab.* 10(3):1044–1115.

Bertsekas DP (2012) *Dynamic Programming and Optimal Control: Approximate Dynamic Programming*, vol. II (Athena Scientific, Belmont, MA).

Best TJ, Sandkç B, Eisenstein DD, Meltzer DO (2015) Managing hospital inpatient bed capacity through partitioning care into focused wings. *Manufacturing Service Oper. Management* 17(2): 157–176.

Brown DB, Haugh MB (2017) Information relaxation bounds for infinite horizon Markov decision processes. *Oper. Res.* 65(5): 1355–1379.

Brown DB, Smith JE, Sun P (2010) Information relaxations and duality in stochastic dynamic programs. *Oper. Res.* 58(4):785–801.

Chan CW, Dong J, Green LV (2017) Queues with time-varying arrivals and inspections with applications to hospital discharge policies. *Oper. Res.* 65(2):469–495.

Dai JG, Shi P (2017) A two-time-scale approach to time-varying queues in hospital inpatient flow management. *Oper. Res.* 65(2): 514–536.

Dai JG, Shi P (2018) Online supplement for "Inpatient overflow: An approximate dynamic programming approach." Working paper, Purdue University, West Lafayette, IN.

Dai JG, Tezcan T (2008) Optimal control of parallel server systems with many servers in heavy traffic. *Queueing Systems* 59(2):95–134.

de Farias DP, Roy BV (2003) The linear programming approach to approximate dynamic programming. *Oper. Res.* 51(6):850–865.

Dong J, Perry O (2017) Queueing models for patient-flow dynamics in inpatient wards. Working paper, Columbia University, New York.

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

Gesenway D (2010) Having problems finding your patients?. Accessed September 4, 2018, http://www.todayshospitalist.com/Having-problems-finding-your-patients/.

Gurvich I, Whitt W (2009a) Queue-and-idleness-ratio controls in many-server service systems. *Math. Oper. Res.* 34(2):363–396.

Gurvich I, Whitt W (2009b) Scheduling flexible servers with convex delay costs in many-server service systems. *Manufacturing Service Oper. Management* 11(2):237–253.

Gurvich I, Whitt W (2010) Service-level differentiation in many-server service systems via queue-ratio routing. *Oper. Res.* 58(2):316–328.

Han S, He S, Oh HC (2016) Models for hospital inpatient operations: A data driven optimization approach for reducing ED boarding times. *Presentation at INFORMS 2016* (INFORMS, Catonsville, MD).

He S, Sim M, Zhang M (2019) Data-driven patient scheduling in emergency departments: A hybrid robust-stochastic approach. *Management Sci.*, ePub ahead of print May 1, https://doi.org/10.1287/mnsc.2018.3145.

Helm JE, Oyen MPV (2014) Design and optimization methods for elective hospital admissions. *Oper. Res.* 62(6):1265–1282.

Hoot NR, Aronsky D (2008) Systematic review of emergency department crowding: Causes, effects, and solutions. *Ann Emerg Med* 52(2):126–136.

Huang J, Carmeli B, Mandelbaum A (2015) Control of patient flow in emergency departments, or multiclass queues with deadlines and feedback. *Oper. Res.* 63(4):892–908.

Huang Q, Thind A, Dreyer J, Zaric G (2010) The impact of delays to admission from the emergency department on inpatient outcomes. *BMC Emergency Medicine* 10(1):16.

Khademi A, Saure DR, Schaefer AJ, Braithwaite RS, Roberts MS (2015) The price of nonabandonment: HIV in resource-limited settings. *Manufacturing Service Oper. Management* 17(4):554–570.

Kilinc D, Saghafian S, Traub SJ (2016) Dynamic assignment of patients to primary and secondary inpatient units: Is patience a virtue. Working paper, Harvard University, Cambridge, MA.

Koole G, Pot A (2005) Approximate dynamic programming in multi-skill call centers. *Proc. Winter Simulation Conf. (IEEE, New York).*

Mandelbaum A, Momcilovic P, Tseytlin Y (2012) On fair routing from emergency departments to hospital wards: QED queues with heterogeneous servers. *Management Sci.* 58(7):1273–1291.

Maxwell MS, Henderson SG, Topaloglu H (2013) Tuning approximate dynamic programming policies for ambulance redeployment via direct search. *Stochastics Systems* 3(2):322–361.

Maxwell MS, Restrepo M, Henderson SG, Topaloglu H (2010) Approximate dynamic programming for ambulance redeployment. *INFORMS J. Comput.* 22(2):266–281.

Meyn SP (2000) Feedback regulation for sequencing and routing in multiclass queueing networks. *SIAM J. Control Optim.* 40(3):741–776

Moallemi CC, Kumar S, Roy BV (2008) Approximate and data-driven dynamic programming for queueing networks. Working paper, Stanford University, Stanford, CA.

National University Hospital (2011) *BMU Training Guide: Inpatient Operations* (National University Hospital Inpatient Department, Singapore).

Pang G, Yao DD (2018) Heavy-traffic limits for a many-server queueing network with switchover. *Adv. Appl. Probab.* 45(3):645–672.

Park E, Ding Y, Nagarajan M, Grafstein E (2019) Patient prioritization in emergency department triage systems: An empirical study of Canadian triage and acuity scale (CTAS). *Manufacturing Service Oper. Management*, ePub ahead of print April 8, https://doi.org/10.1287/msom.2018.0719.

Perry O, Whitt W (2009) Responding to unexpected overloads in large-scale service systems. *Management Sci.* 55(8):1353–1367.

Perry O, Whittv W (2011a) A fluid approximation for service systems responding to unexpected overloads. *Oper. Res.* 59(5):1159–1170.

Perry O, Whitt W (2011b) An ODE for an overloaded X model involving a stochastic averaging principle. *Stochastics Systems* 1(1): 59–108.

Pines JM, Batt RJ, Hilton JA, Terwiesch C (2011) The financial consequences of lost demand and reducing boarding in hospital emergency departments. *Ann. Emergency Medicine* 58(4):331–340.

Powell ES, Khare RK, Venkatesh AK, Roo BDV, Adams JG, Reinhardt G (2011) The relationship between inpatient discharge timing and emergency department boarding. *J. Emergency Medicine* 42(2): 186–196.

Powell WB (2011) *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. *Wiley Series in Probability and Statistics* (Wiley-Interscience, Hoboken, NJ).

Puterman ML (1994) *Markov Decision Processes: Discrete Stochastic Dynamic Programming* (John Wiley, New York).

Rabin E, Kocher K, McClelland M, Pines J, Hwang U, Rathlev N, Asplin B, Trueger NS, Weber E (2012) Solutions to emergency department 'boarding' and crowding are underused and may need to be legislated. *Health Affairs* 31(8):1757–1766.

Roubos D, Bhulai S (2010) Approximate dynamic programming techniques for the control of time-varying queuing systems applied to call centers with abandonments and retrials. *Probab. Engrg. Inform. Sci.* 24(1):27–45.

Samiedaluie S, Kucukyazici B, Verter V, Zhang D (2017) Managing patient admissions in a neurology ward. *Oper. Res.* 65(3): 635–656.

Shi P, Chou MC, Dai JG, Ding D, Sim J (2016) Models and insights for hospital inpatient operations: Time-dependent ED boarding time. *Management Sci.* 62(1):1–28.

Shi P, Dai JG, Ding D, (James) Ang SK, Chou M, Jin X, Sim J (2014) Patient flow from emergency department to inpatient wards: Empirical observations from a Singaporean hospital. Working paper, Purdue University, West Lafayette, IN.

Singer AJ, Jr. Thode HC, Viccellio P, Pines JM (2011) The association between length of emergency department boarding and mortality. *Academic Emergency Medicine* 18(12):1324–1329.

Song H, Tucker A, Graue R, Moravick S, Yang J (2018) Capacity pooling in hospitals: The hidden consequences of off-service placement. Working paper, University of Pennsylvania, Philadelphia.

Stolyar AL, Tezcan T (2010) Control of systems with flexible multi-server pools: A shadow routing approach. *Queueing Systems* 66(1):1–51.

Stolyar AL, Tezcan T (2011) Shadow-routing based control of flexible multiserver pools in overload. *Oper. Res.* 59(6):1427–1444.

Sutton RS (1988) Learning to predict by the methods of temporal differences. *Machine Learn* 3(1):9–44.

Teow K, El-Darzi E, Foo C, Jin X, Sim J (2011) Intelligent analysis of acute bed overflow in a tertiary hospital in Singapore. *J. Medical Systems* 36(3):1873–1882.

Tezcan T, Dai JG (2010) Dynamic control of *N*-systems with many servers: Asymptotic optimality of a static priority policy in heavy traffic. *Oper. Res.* 58(1):94–110.

Thompson S, Nunez M, Garfinkel R, Dean MD (2009) Efficient short-term allocation and reallocation of patients to floors of a hospital during demand surges. *Oper. Res.* 57(2):261–273.

Veatch MH (2005) Approximate dynamic programming for networks: Fluid models and constraint reduction. Working paper, Gordon College, Wenham, MA.

Watkins CJCH, Dayan P (1992) Technical note: Q-learning. *Machine Learn.* 8(3):279–292.