Stochastic Systems

arXiv: arXiv:0000.0000

TUNING APPROXIMATE DYNAMIC PROGRAMMING POLICIES FOR AMBULANCE REDEPLOYMENT VIA DIRECT SEARCH

By Matthew S. Maxwell, Shane G. Henderson and Huseyin Topaloglu

Cornell University

In this paper we consider approximate dynamic programming methods for ambulance redeployment. We first demonstrate through simple examples how typical value function fitting techniques, such as approximate policy iteration and linear programming, may not be able to locate a high-quality policy even when the value function approximation architecture is rich enough to provide the optimal policy. To make up for this potential shortcoming, we show how to use direct search methods to tune the parameters in a value function approximation architecture so as to obtain high-quality policies. Direct search is computationally intensive. We therefore use a post-decision state dynamic programming formulation of ambulance redeployment that, together with direct search, requires far less computation with no noticeable performance loss. We provide further theoretical support for the post-decision state formulation of the ambulance-deployment problem by showing that this formulation can be obtained through a limiting argument on the original dynamic programming formulation.

1. Introduction. Emergency medical service (EMS) providers are tasked with staffing and positioning emergency vehicles to supply a region with emergency medical care and ensure short emergency response times. Large operating costs and increasing numbers of emergency calls (henceforth "calls") make this a demanding task. One method commonly used to reduce response times to calls is known as ambulance redeployment. Ambulance redeployment, also known as move-up or system-status management, is the strategy of relocating idle ambulances in real time to minimize response times for future calls. When making ambulance redeployment decisions it is necessary to take into account the current state of the system, which may include the position and operating status of ambulances in the fleet, the number and nature of calls that are queued for service and external factors such as traffic jams and weather conditions. Furthermore, the system evolves under uncertainty and the probability distributions for where and when future calls are likely to arrive influence how the ambulances should be deployed. Dynamic programming (DP) provides an attractive framework to build models for ambulance redeployment as it allows us to capture the randomness in the

evolution of the system and to make decisions that depend on the system state. However, DP-based ambulance redeployment models require keeping track of a large amount of information to represent the state of the system. This requirement yields high-dimensional state variables and DP models often end up being computationally intractable.

In this paper, we investigate the use of approximate dynamic programming (ADP) techniques for building tractable ambulance redeployment models. Our approach models the ambulance redeployment problem as a Markov decision process (MDP) and defines a parameterized approximation J_r of the DP value function J. The approximation J_r is constructed as a linear combination of B "basis" functions that are weighed by using the set of coefficients $r = (r_1, \ldots, r_B) \in \Re^B$. Thus, each set of coefficients r provides a different approximation to the value function. We can use the approximate value function J_r to construct an approximate ambulance redeployment policy, where the decision we make at any point in time is chosen as the one that minimizes the sum of the immediate expected cost and the expected value function approximation at the next decision epoch. The main question of interest is how to choose the set of parameters r so that the ambulance redeployment decisions from the approximate policy perform well and the decisions can be computed quickly. We make contributions on both how to choose the set of parameters r and how to speed up the computation times for decisions.

There are several methods to choose the set of parameters r. One method is to collect cost trajectories of the system under a reasonable policy and choose the parameters r so that the value function approximation J_r provides a good fit to the collected cost trajectories. This idea forms the foundation underlying temporal learning-based methods and approximate policy iteration [12, 33]. There is also some work that directly solves a regression problem to fit J_r to cost trajectories of the system [45, 25]. Another approach is to use the linear programming (LP) representation of the MDP formulation of the ambulance redeployment problem. In this LP, there is one decision variable corresponding to the value function in each state of the system. So, the number of decision variables can be very large or even infinite, but we can get around this difficulty by replacing the value functions in the LP with the approximation J_r. In this case, the parameters r become the decision variables in the LP, which may not be too many for a reasonable choice of the set of basis functions. Solving the LP yields a set of parameters r, which, in turn, characterizes a value function approximation J_r. In this paper, we demonstrate an interesting shortcoming of both the regression-based and LP-based methods for choosing a set of parameters r.

In particular, we consider the case where there exists a set of parameters r so that the approximate policy that we obtain by using the value function approximation J_r is an optimal policy. In other words, by choosing the parameters r appropriately, it is possible to recover an optimal policy. However, we demonstrate that both the regression-based method and the LP-based method may fail to identify such a value of r and may end up with suboptimal policies. This is of some concern, because although an optimal policy is within the reach of our approximation architecture, the regression-based and LP-based approaches may miss this policy.

To overcome this potential pitfall, we propose using direct search methods to choose a set of parameters r. In particular, the important metric in practical applications is the performance of the policy obtained by using the value function approximation J_r. Each $r \in \Re^B$ yields a value function approximation J_r, which, in turn, provides an approximate policy with a corresponding total expected cost. For a given initial state of the system s_0, we use $\hat{J}_r(s_0)$ to denote the total expected cost incurred by the approximate policy characterized by the set of parameters r. We note that $\hat{J}_r(s_0)$ can be estimated by simulating the approximate policy for multiple sample paths. In this case, we can use direct search methods to solve the optimization problem $\min_{r \in \Re^B} \hat{J}_r(s_0)$ and obtain a set of parameters r that characterizes an approximate policy with desirable performance. Since we can compute $\hat{J}_r(s_0)$ for any $r \in \Re^B$ by using simulation, direct search methods that use only function values are particularly suitable for solving the last optimization problem. In this paper, we experiment with direct optimization methods based on simplex search [30] and successive quadratic approximations [16]. Another possibility is to use the cross-entropy method; see [42] for an application to the game of Tetris. We report significant improvements when compared with a more standard, regression-based method for choosing the set of parameters r.

One potential drawback of using direct search methods to choose a set of parameters is that we need to compute $\hat{J}_r(s_0)$ for every value of r that we visit during the course of the search process. Computing $\hat{J}_r(s_0)$ requires simulating the approximate policy characterized by the parameters r. On the other hand, simulating the approximate policy characterized by the parameters r requires making decisions sequentially over time, where each decision is chosen as the one that minimizes the sum of the immediate expected cost and the expected value function approximation at the next decision epoch. Due to the necessity of estimating expectations to implement the decisions of the approximate policy, the computational requirements of direct search can be quite large. To overcome this difficulty, we give an alternative MDP

formulation of the ambulance redeployment problem that measures the state of the system immediately after making a redeployment decision. We refer to this formulation as the post-decision state formulation. The useful feature of the post-decision state formulation is that the value function approximation is computed by using the post-decision state and the post-decision state is a deterministic function of the state and decision at the current epoch. Thus, the post-decision state formulation avoids the necessity to compute expectations when simulating the policy characterized by a value function approximation. This flexibility allows us to improve the computation times for direct search methods by an order of magnitude, with no noticeable performance loss.

To further motivate the post-decision state formulation and provide theoretical support for it, we consider an alternative DP formulation where the next artificial decision epoch is taken to be the earlier of the next true decision epoch in the system and a deterministic time that is τ time units into the future. This definition of a decision epoch yields a correct DP formulation for the ambulance redeployment problem for any choice of $\tau > 0$. The standard DP formulation in discrete time arises when $\tau = \infty$. As τ shrinks towards 0, the simulation effort required to determine decisions is reduced, but we obtain less and less information on the expected cost accrued until the next true decision epoch, so we rely more and more on the value-function approximation. We show that in the limit as $\tau \to 0$, we arrive at the post-decision state formulation, so that the post-decision state formulation can be viewed as the limit of a series of DP formulations designed to reduce computational effort. This interpretation of the post-decision state formulation as a limit should hold in some generality. Section 7 discusses the kinds of systems in which we expect this to be the case.

The idea of measuring the state of the system right after a decision appears in [46, 41, 33]. Similar to us, [33] motivates the use of the post-decision state variable through ADP. In that work, the post-decision state variable is used to reduce the number of dimensions of the state variable. A state variable with a smaller number of dimensions suggests a less complicated value function to approximate. For our application setting, the post-decision state variable does not necessarily change the number of dimensions of the state variable, but it allows us to bypass computing expectations when simulating the behavior of an approximate policy. [31], [39], [44] and [46] use the post-decision state variable in batch service, dynamic assignment, fleet management and inventory control problems.

The ambulance redeployment literature contains a number of different approaches for computing redeployment policies. One approach is to formulate

an integer program that models ambulance redeployment and to solve this integer program in real-time whenever a redeployment decision is required [20, 23, 29, 35]. This approach is computationally intensive and sometimes requires a parallel computing environment, possibly along with heuristic solution methods, to obtain a redeployment decision in real time. Another approach is to solve an integer program in a preparatory phase which determines a "lookup table," or the desired ambulance locations given the number of currently available ambulances [3, 21]. Real-time redeployment is managed by dispatchers who attempt to position ambulances according to the prescribed locations. Other approaches attempt to directly incorporate the randomness of calls into the model formulation. For example, [7, 8, 9] formulate the ambulance redeployment problem as a DP. This approach was revisited more recently in [48] in an attempt to gain insight into the problem. As mentioned above, the difficulty of working with DP formulations is that they are intractable for realistic problem sizes and computing optimal policies is possible only in simplified situations such as those having only one or two ambulances. The work of [4, 5] incorporates random evolution of the system heuristically through the construction of a "preparedness function" that attempts to evaluate future states based on their ability to handle incoming calls. Recently, [47] has developed another approach that involves solving integer programs in real time. The integer programs require certain coefficients that are chosen in a "tuning phase" that uses simulation optimization, in the same manner developed herein. Indeed, the developments in [47] were partially motivated by the work presented here.

There has been earlier work on obtaining ambulance redeployment policies using ADP [26, 27, 28, 36]. The ADP approach is flexible enough to deal with the random evolution of the system directly and does not have the computational burden of the integer programming methods. Similar to the lookup table approach, ADP algorithms require a preparatory tuning process which may be computationally expensive, but after this initial computation most ADP policies are able to operate in real-time situations without computational concerns. The contribution in this paper relative to this earlier work is that we show how to use direct search, in conjunction with a post-decision state formulation, to dramatically reduce the computation required for tuning, making it possible to design effective ADP policies for large metropolitan regions.

ADP methods only approximate the true value function, and so the resulting ADP policies are not necessarily optimal. Nevertheless, given a suitable approximation architecture, ADP policies have been shown to perform well in problems that would be intractable otherwise. Examples of ADP appli-

cations include inventory control [46], inventory routing [1], option pricing [45], backgammon [43], dynamic fleet management [44], and network revenue management [2, 19].

One difficult aspect of using ADP is the choice of basis functions. An effective choice of basis functions generally requires expert opinion, accurate intuition of the MDP dynamics, and significant trial and error. There are no generally accepted rules to pick suitable basis functions and they must be chosen manually to represent the key features of the system. On the other hand, there are numerous algorithmic methods for tuning the parameter vector r to obtain good policies given a set of basis functions, e.g., approximate policy iteration [12], temporal-difference learning [40], least-squares temporal-difference learning [14, 13], linear programming [37, 15], and smoothed approximate linear programming [17].

In the remainder of the paper, Section 2 introduces ADP and describes the construction of ADP policies. Section 3 uses a sample problem to describe theoretical limitations of common parameter tuning methods that are based on regression and the LP formulation. Section 4 describes the ambulance redeployment problem in detail, formulates ambulance redeployment as an MDP, and gives an ADP approach for ambulance redeployment. Section 5 illustrates how the limitations in Section 3 can have significant negative effects on policy performance and how direct search methods are able to provide better-performing policies than traditional tuning approaches for ambulance redeployment. Section 6 introduces the post-decision state formulation and shows that, coupled with direct search, the post-decision formulation of ambulance redeployment is an efficient method for tuning ADP parameters in high-dimensional problems. Section 7 concludes.

2. Approximate Dynamic Programming. Finding an optimal policy in an MDP depends upon the computation of the DP value function J. Unfortunately, computing the value function J is intractable for many MDPs. One common approach to overcome this difficulty is to approximate J with a parameterized formulation that is easy to compute. Section 2.1 explains how these approximations are used to create ADP policies and Section 2.2 describes two general approaches used to tune the ADP approximation parameters. In this section, and also in Section 3 we focus on infinite-horizon, discounted-cost problems for simplicity and clarity, but the key ideas generalize to other formulations including, for example, cost up to absorption, as discussed briefly in Section 3.

2.1. *ADP Policies.* A discrete-time MDP is defined by a state space, a control space, system dynamics, transition costs, and an objective function.

A state $s \in \mathcal{S}$ contains enough information such that the future evolution of the MDP is independent of the past evolution given the current state. The state space $\mathcal{S}$ is the set of all possible states that may be realized by the MDP. The notation S_k denotes the kth state of the MDP evolution. For each $s \in \mathcal{S}$ we define a control space $\mathcal{X}(s)$ that dictates the available actions in state s. The control space for the MDP is defined as $\mathcal{X} = \bigcup_{s\in\mathcal{S}} \mathcal{X}(s)$ and we assume $|\mathcal{X}(s)| < \infty$ for all $s \in \mathcal{S}$. Let U_{k+1} denote a vector of iid Uniform$(0,1)$ random variables (with appropriate dimension) used to generate all random behavior between states S_k and S_{k+1}. We denote the MDP system dynamics as $S_{k+1} = f(S_k, x, U_{k+1})$ where the next state S_{k+1} results from being in state S_k, choosing action $x \in \mathcal{X}(S_k)$, and having random effects dictated by U_{k+1}.

Let $c(S_k, x, U_{k+1})$ denote the (possibly random) transition cost associated with being in state S_k and choosing action $x \in \mathcal{X}(S_k)$. One common objective function for MDPs is to minimize the expected sum of the discounted transition costs from a given starting state S_0, i.e.,

$$\min_{\pi} \mathbb{E}\left[\sum_{k=0}^{\infty} \alpha^k c\left(S_k, \pi(S_k), U_{k+1}\right)\right], \tag{2.1}$$

where a policy π is a mapping from $\mathcal{S}$ to $\mathcal{X}$ such that $\pi(s) \in \mathcal{X}(s)$ for all $s \in \mathcal{S}$ and $\alpha \in (0,1)$ is the discount factor. This expectation is finite if, e.g., c is bounded. The value function for policy π starting in state S_0 is defined as

$$J^{\pi}(S_0) = \mathbb{E}\left[\sum_{k=0}^{\infty} \alpha^k c(S_k, \pi(S_k), U_{k+1})\right].$$

For an optimal policy π^* we define the value function $J = J^{\pi^*}$. From the DP optimality principle we know that

$$J(S_k) = \min_{x\in\mathcal{X}(S_k)} \mathbb{E}\left[c(S_k, x, U_{k+1}) + \alpha J\left(f(S_k, x, U_{k+1})\right)\right] \qquad \forall S_k \in \mathcal{S}, \tag{2.2}$$

and that choosing an action $x \in \mathcal{X}(S_k)$ achieving the minimum in the right-hand side of (2.2) for every state $S_k \in \mathcal{S}$ yields an optimal policy [11]. Unfortunately, the complexity of computing J often increases drastically in the size of $\mathcal{S}$; hence most MDPs with large state spaces are intractable to solve via DP.

One approach to overcome these computational difficulties is to approximate J with a simpler function and to use the approximation in lieu of the value function when computing policy decisions. This approach is known as ADP. For example, given "basis functions" $\phi_1, \ldots, \phi_B$ mapping $\mathcal{S}$ to $\mathbb{R}$ we

define a linear approximation architecture $J_r(\cdot) = \sum_{b=1}^{B} r_b \phi_b(\cdot)$ where the subscript r denotes the vector $(r_1, \ldots, r_B)$ of tunable parameters. Given an approximation J_r for J we define the quantity

$$L_r(S_k) = \min_{x \in \mathcal{X}(S_k)} \mathbb{E}\left[c(S_k, x, U_{k+1}) + \alpha J_r\left(f(S_k, x, U_{k+1})\right)\right] \quad \forall S_k \in \mathcal{S}, \tag{2.3}$$

and the ADP policy with respect to L_r chooses an action $x \in \mathcal{X}(S_k)$ achieving the minimum in the right-hand side. In this manner, L_r defines an ADP policy. This ADP policy is also known as the greedy policy with respect to J_r; however, we use the former terminology to distinguish between ADP policies that use the same approximation architecture J_r but are formulated differently.

For any given state $s \in \mathcal{S}$ there may be multiple actions achieving the minimum on the right-hand side of (2.2) or (2.3). In DP any action achieving the minimum may be chosen without loss of optimality. However, in an ADP context these actions may have very different consequences. Nevertheless, it is common to regard all actions achieving $L_r(s)$ as equivalent choices since they are equivalent insofar as the approximation architecture is able to differentiate. Because the vector of tunable coefficients r ultimately affects actions chosen and overall performance these coefficients are referred to as ADP policy parameters. These coefficients are generally tuned in a preparatory phase before an ADP policy is implemented.

For further information, see [34] for MDPs, [10] for DP, and [12, 33] for ADP.

2.2. *Tuning ADP policies.* Theoretical results show that if $J_r(s) \approx J(s)$ for all $s \in \mathcal{S}$ then the performance of the greedy policy with respect to J_r is not too far from the optimal policy performance. For example, [12, Proposition 6.1] considers infinite-horizon discounted problems with finite state and action spaces and proves that $\|J_r^* - J\|_\infty \leq 2\alpha\|J_r - J\|_\infty/(1-\alpha)$, where $\alpha \in (0,1)$ is the discount factor, and J_r^* is the *true* value function associated with the greedy policy induced by the approximation J_r. Thus, one can get arbitrarily close to the optimal policy with an arbitrarily accurate approximation for the true value function, and owing to the finite number of potential policies, an optimal policy is actually attained when $\|J_r - J\|_\infty$ is sufficiently small.

The standard method for tuning approximation architectures therefore tries to ensure that $J_r \approx J$. In this sense, value function fitting methods attempt to find coefficients r that solve

$$\min_r \|J_r - J\|, \tag{2.4}$$

for some distance measure $\|\cdot\|$. These methods tune the coefficients based upon approximating the true value function with the hope of getting good policy performance as a result.

The objective function in (2.4) does not represent the true goal. Instead, we are interested in the performance of the policy represented by the coefficients r. Direct search attempts to optimize this true goal, i.e., to solve

$$\min_r \mathbb{E}\left[\sum_{k=0}^{\infty} \alpha^k c\left(S_k, \pi_r(S_k), U_{k+1}\right)\right], \tag{2.5}$$

where π_r denotes the ADP policy with respect to, say, L_r. Thus the direct search method tunes the policy coefficients of the value function approximation J_r based solely upon the performance of the ADP policy using J_r.

Why might there be differences in the performance of policies where the coefficients are chosen in this manner? The choice of action in a fixed state depends on the *relative* values of the value function approximation J_r in the current and potential future states, and not on the individual values themselves. Thus, an approximate value function J_r bearing little resemblance to the actual value function J can still induce a good policy, and there might be large differences in the performance of policies obtained from (2.4) and (2.5). A striking example involves the game of Tetris [42], where the cross-entropy method was used to improve average game scores by an order of magnitude over value function fitting methods.

The key observation is that value function fitting methods may not always return the coefficients corresponding to the best policy performance. This can happen even when the optimal policy arises from a particular choice of r, as we see next.

3. Limitations of Common ADP Tuning Approaches. Two general approaches for value function fitting are regression and linear programming. In this section we show through simple examples that even when a choice of coefficients r induces the optimal policy, neither tuning method may be able to recover it. The reason is that the fitting criterion is not typically aligned with the performance of the greedy policy induced by the approximate value function.

3.1. *Limitations of Regression-Based Approaches.* Given an initial policy π_0, regression-based methods take a noisy estimate, $\hat{J}^{\pi_0}(s)$, of the value function for π_0 starting at state s for each $s \in \mathcal{S}$ (or perhaps only a subset of $\mathcal{S}$). A new set of policy parameters $r^*_{\pi_0,p}$ is calculated via regression, i.e.,

$$r^*_{\pi_0,p} = \operatorname*{argmin}_r \left\{ \sum_{s\in\mathcal{S}} \left| \hat{J}^{\pi_0}(s) - J_r(s) \right|^p \right\}, \tag{3.1}$$

where $1 \le p \le \infty$ indicates the p-norm used in the regression. Commonly in ADP, the least-squares regression is formulated recursively and used in conjunction with simulation to update policy parameters after each sampled state transition; see [12, 33].

Usually the regression-based tuning is iterated with the hope of finding a set of parameters inducing a policy with good performance. For example, one standard tuning approach called approximate policy iteration begins with an initial policy π_0 and then simulates state trajectories using π_0 that are used to compute the coefficients $r^*_{\pi_0,2}$ via (3.1). In the next iteration π_1 is set to be the greedy policy with respect to the approximation architecture using the coefficients from the initial iteration, i.e., $J_{r^*_{\pi_0,2}}$, and π_1 is used to simulate state trajectories and compute the coefficients for the next iteration $r^*_{\pi_1,2}$. This process is repeated until the computational budget is exhausted or a suitable policy is obtained.

Regression-based tuning methods are appealing because they are easily understood, easily implemented, and fast to compute, but they may have drawbacks.

Consider the MDP shown in Figure 1 where the objective is to minimize the discounted sum of the transition costs starting from state 3, where $\alpha \in (0, 1]$ is the discount factor. (When $\alpha = 1$ there is no discounting and we are instead minimizing expected total cost up to absorption in State 0.) In this MDP there are only two deterministic policies. Let π_1 (π_2) denote the policy that opts to transition to State 1 (State 2) from State 3. Consider the case where $\alpha > 3/4$. In this case π_2 is the optimal policy.

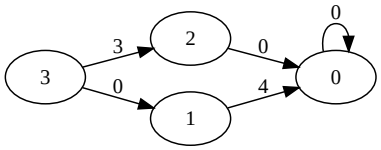

FIG 1. *Example MDP*

Let $\phi_1(s) = s$ and consider the approximation architecture $J_r(s) = r_1\phi_1(s)$. For the approximation architecture to induce the policy π_2 we must have $3 + \alpha J_r(2) = 3 + 2\alpha r_1 < \alpha r_1 = 0 + \alpha J_r(1)$ which implies that $r_1 < -3/\alpha$.

Thus, the approximation architecture J_r is rich enough to induce an optimal policy provided that r_1 is chosen correctly.

Now consider a p-norm regression as in (3.1) with $1 \leq p < \infty$. Under the policy π_1 we have that

$$\begin{aligned}\sum_{s=0}^{3} |J^{\pi_1}(s) - r_1 s|^p &= |0-0|^p + |4-r_1|^p + |0-2r_1|^p + |4\alpha - 3r_1|^p \\ &= |4-r_1|^p + 2^p |r_1|^p + |4\alpha - 3r_1|^p . \end{aligned} \tag{3.2}$$

For any $r_1 < 0$, each term of (3.2) can be decreased by increasing r_1 to 0. Hence the set of coefficients returned by the regression satisfies $r^* \geq 0$, and the induced policy is the suboptimal policy π_1 for any $1 \leq p < \infty$. A similar analysis holds when starting from the policy π_2, so in either case we will induce the suboptimal policy π_2. The same argument works when $p = \infty$.

Thus, in this example regression with respect to any p-norm will always return a sub-optimal policy. The same result holds when any subset of $\mathcal{S}$ is used in the regression (with the exception of $\{0\}$ for which any $r_1 \in \mathbb{R}$ is a valid regression solution). Furthermore, given any discount factor $\alpha \in (0, 3/4]$ we can create a similar example where suboptimal policies result. Additionally, with minor modifications to the MDP in Figure 1, we can create MDPs where the suboptimal policy is arbitrarily worse than the optimal policy. A similar example for an undiscounted MDP is given in [27].

3.2. *Limitations of LP-Based Approaches.* LP-based methods are based on the LP formulation of exact dynamic programs and have the form

$$\begin{aligned} \max_{r} \quad & \nu^T \bar{J}_r \\ \text{s.t.} \quad & \mathbb{E}\left[c(S_k, x, U_{k+1}) + \alpha J_r(f(S_k, x, U_{k+1}))\right] \geq J_r(S_k) \quad \forall S_k \in \mathcal{S},\ x \in \mathcal{X}(S_k) \end{aligned} \tag{3.3}$$

for minimizing the expected α-discounted transition costs, where ν is a (column) vector with positive components and $\bar{J}_r$ denotes a vector containing $J_r(S_k)$ for all $S_k \in \mathcal{S}$, see [37, 15]. This LP formulation is most useful when the expectation in the constraints can be computed exactly or estimated easily. Also, due to the number of constraints in this LP, it is often necessary to use techniques such as constraint sampling to make the LP tractable.

Again consider the MDP shown in Figure 1 with discount factor $\alpha \in (3/4, 1)$. Let $\phi_1(s) = 1$ and $\phi_2(s) = s$ and consider the approximation architecture $J_r(s) = r_1\phi_1(s) + r_2\phi_2(s)$. For the approximation architecture to induce the optimal policy π_2 we must have $3 + \alpha J_r(2) = 3 + \alpha r_1 + 2\alpha r_2 < \alpha r_1 + \alpha r_2 = 0 + \alpha J_r(1)$ which implies that $r_2 < -3/\alpha$.

Now consider what happens when we tune the coefficients using the LP-based formulation. The LP formulation for the MDP in Figure 1 can be written as

$$\max_{r} \quad \begin{bmatrix}\nu_1 & \nu_2 & \nu_3 & \nu_4\end{bmatrix} \begin{bmatrix} r_1 \\ r_1 + r_2 \\ r_1 + 2r_2 \\ r_1 + 3r_2 \end{bmatrix}$$

$$\text{s.t.} \quad (1-\alpha)r_1 \leq 0 \tag{3.4}$$

$$(1-\alpha)r_1 + r_2 \leq 4 \tag{3.5}$$

$$(1-\alpha)r_1 + 2r_2 \leq 0 \tag{3.6}$$

$$(1-\alpha)r_1 + (3-\alpha)r_2 \leq 0 \tag{3.7}$$

$$(1-\alpha)r_1 + (3-2\alpha)r_2 \leq 3. \tag{3.8}$$

Let $q_\alpha = (1-\alpha)r_1$, and rewrite (3.5)-(3.8) as

$$\begin{aligned} r_2 &\leq 4 - q_\alpha \\ r_2 &\leq \frac{-q_\alpha}{2} \\ r_2 &\leq \frac{-q_\alpha}{3-\alpha} \\ r_2 &\leq \frac{3-q_\alpha}{3-2\alpha}. \end{aligned}$$

Since $q_\alpha \leq 0$ by (3.4) we know that the feasible region for (r_1, r_2) contains at least $\{(r_1, r_2) : r_1 \leq 0, r_2 \leq 0\}$. Given any feasible $r = (r_1, r_2)$ inducing an optimal ADP policy (i.e., $r_1 \leq 0$ and $r_2 < -3/\alpha$) we have that $r' = (r_1, 0)$ is also feasible. The objective function value for r' is greater than that of r by $-(\nu_2 + 2\nu_3 + 3\nu_4)r_2 > 0$. Consequently, coefficients inducing an optimal ADP policy will not induce an optimal LP solution, and hence will never be returned via an LP-based tuning approach.

Given any $\alpha \in (0, 1)$, one can find a value for the transition cost from state 1 to state 0 in the example MDP so that LP-based tuning again fails to recover the optimal policy. Additionally, the difference between the optimal policy performance and that obtained via LP-based tuning can be made arbitrarily large by increasing this transition cost. Discounting is not essential. Indeed, when $\alpha = 1$ so that we are minimizing expected (undiscounted) cost till absorption, we drop the constant (first) basis function, and impose the boundary condition $J_r(0) = 0$ in the tuning LP. Again an optimal policy is obtained for a particular choice of r but the LP formulation will not return it.

The MDP in Figure 1 is of course a simple example that is easily solved exactly, but it is typical of ADP applications where the approximation architecture does not encompass the true value function. Furthermore, since the example MDP is deterministic the inability to properly tune the policy coefficients is a direct consequence of the tuning methods themselves, as opposed to sampling-based errors that may arise when dealing with random state transitions.

In Sections 4 and 5 we show that similar issues arise in a realistic application of ADP, namely ambulance redeployment.

4. Ambulance Redeployment. To model EMS operations we divide the region under consideration into an appropriate-sized grid and assume that calls in each grid cell arrive according to independent time-inhomogeneous Poisson processes and are independently distributed uniformly throughout the cell. When a call arrives, the dispatcher assigns the closest available ambulance to the call. If there are no ambulances available the call is placed on a waiting list and served first-come first-served as ambulances become available.

In reality the closest ambulance is not always the one dispatched to a call, but it is a close approximation of the policy followed in most EMS systems. Furthermore, in reality calls are prioritized into "levels of care" but we assume a single level of care.

If the ambulance assigned to a call was available at base at the time of the call, then we assume it takes 45 seconds for the crew to reach their vehicle and be ready to leave. This delay is called *turn-out* time or *chute* time. If the assigned ambulance was not idle at a base, then it does not incur turn-out time.

After being assigned to a call the ambulance travels to the call scene. We assume deterministic travel times along the shortest path of a representative street network. The distance to calls off the road network are calculated using the Manhattan distance from the nearest node. Although not used in our model, random travel times could be incorporated with little modification.

Paramedics provide preliminary care to the patient at the scene. We model this "on scene" time as an exponentially distributed random variable having a mean of 12 minutes. In approximately 25% of the cases the patient does not need to be transported to a hospital and the ambulance becomes free at the call scene. Otherwise, the patient is transported to a hospital and transferred to the hospital staff. The destination hospital is chosen from the empirical distribution of hospital destinations given the location of the emergency, and the transfer time at the hospital is modeled as a Weibull-distributed random

variable with a mean of 30 minutes and standard deviation of 13 minutes.

Our ADP policy makes redeployment decisions when an ambulance becomes available after completing care for the patient, either at the call scene or the hospital. Thus, when an ambulance becomes available, the redeployment policy is used to calculate the desired redeployment base for the newly free ambulance. After the redeployment decision is made, the ambulance travels to this destination to wait for future calls; however, if a call arrives before the ambulance has reached its destination it may still be assigned to the call (provided it is closest to the call). Our goal is to tune the ADP policy so that we maximize the fraction of calls with response times (time from receipt of the call to the time when the ambulance stops at the destination) under a time threshold which is typically 8 or 9 minutes. We only consider redeploying ambulances to bases, but other locations such as convenient street intersections and hospitals could be included as well. If there are calls on the waiting list when an ambulance becomes available, the ambulance is immediately assigned to the first call on the waiting list and no redeployment decision is made.

Section 4.1 gives an MDP formulation for this problem and Section 4.2 presents the ADP policy for ambulance redeployment including an explanation of the basis functions used within the approximation architecture and the methods used to compute the policy.

4.1. *Ambulance Redeployment as an MDP.* We model the ambulance redeployment problem as a queuing system within the generalized semi-Markov decision process framework [22]. In this setup calls are customers and ambulances are servers. The service time for a call includes the response time, time at scene, and transport and transfer time to a hospital if needed.

4.1.1. *State Space.* Let N denote the number of ambulances, and let c_i for $i = 1, \ldots, N$ denote the location of the call being served by ambulance i, with $c_i = \emptyset$ if ambulance i is available. Let a_i denote the location of ambulance i at the time it last responded to or completed serving a call. In this queuing system, the service time distribution for a call depends upon both c_i and a_i. Let r_i denote the redeployment base for ambulance i, i.e., the location of the base to which ambulance i will position itself once ambulance i becomes available. If ambulance i is serving a call, then the value of r_i does not impact the system dynamics until the ambulance becomes available. Control actions in this MDP will set these redeployment values as ambulances become available. Let w_i for $i = 1, \ldots, M$ denote the location of the ith call on the waiting list, with $w_i = \emptyset$ if there is no ith call waiting. If a call arrives and there are already M calls on the waiting list we assume

the call is handled by another agency. This assumption has very little practical implication since one could take M to be quite large to capture EMS operations having no alternative supporting agencies.

The events in this discrete-event dynamic system (DEDS) are e_0 indicating that a call arrival has occurred, and e_i for $i = 1, \ldots, N$ indicating that ambulance i has completed serving its assigned call. Let t denote the current simulation time, t_0 denote the time since the last call arrival, and t_i for $i = 1, \ldots, N$ denote the elapsed time since ambulance i was either assigned to a call or finished serving a call.

Let $C = (c_1, \ldots, c_N)$, $A = (a_1, \ldots, a_N)$, $R = (r_1, \ldots, r_N)$, $W = (w_1, \ldots, w_M)$, and $T = (t, t_0, \ldots, t_N)$. Thus the state of the DEDS can be denoted as $s = (C, A, R, W, e, T)$ where $e \in \{e_0, \ldots, e_N\}$. Let $C(s)$, $A(s)$, $R(s)$, $W(s)$, $e(s)$, and $T(s)$ denote the C, A, R, W, e, and T components of state s. Additionally, let $c_i(s)$, $a_i(s)$, $r_i(s)$, and $w_i(s)$ denote the ith component of $C(s)$, $A(s)$, $R(s)$, and $W(s)$, let $t(s)$ denote the t component of $T(s)$, and let $t_i(s)$ denote the t_i component of $T(s)$.

Ambulance dispatchers typically have more information at their disposal than that contained within this state space representation. For example, dispatchers often know the status of the busy ambulances–whether they are treating a patient at the call scene, transporting a patient to a hospital, or at a hospital transferring a patient. Dispatchers use this information in making redeployment decisions. For example, if a call arises near a hospital at which some ambulance has nearly completed patient transfer, the dispatcher may not assign an ambulance to the call and instead wait for the ambulance at the hospital to become available. In this sense, the state space representation is a simplified model of dispatcher information and the simulated dynamics are a simplification of ambulance redeployment dynamics (as is typical for simulation).

4.1.2. *Control Space.* We call state s a decision state if there is a redeployment decision to be made in that state, i.e., if an ambulance just became available, $e(s) = e_i$ and $t_i(s) = 0$ for some $1 \leq i \leq N$, and there are no calls on the waiting list, $w_1(s) = \emptyset$. For decision states $\mathcal{X}(s)$ is the set of potential locations to which we may direct the newly-freed ambulance. We consider this set to be a predetermined set of ambulance bases in the proximity of an ambulance's home base. For "non-decision states" $\mathcal{X}(s) = \{\emptyset\}$ indicating a "do-nothing" action. We assume that $|\mathcal{X}(s)|$ is finite (and not too large) for all $s \in \mathcal{S}$ so that the right-hand side of (2.3) can be estimated via Monte Carlo for each $x \in \mathcal{X}(s)$ within real-time computation constraints.

In practice, more complicated redeployment actions are often taken. For

example, when one ambulance becomes free, multiple idle ambulances may be asked to relocate along with the newly freed ambulance. Such policies could be implemented within our framework, but we do not do so for three reasons. First, as the complexity of potential redeployment options increases, the time required to choose the decision increases as well. Second, although more complex redeployment decisions may be able to increase performance over single ambulance redeployments, we believe the potential improvement to be marginal. Third, ambulance crews often consider redeployments to be, at best, a nuisance. It is therefore important to not over-burden crews with redeployments. Our strategy of only repositioning newly free ambulances involves minimal disruption, since the crews are already "on the road." It is straightforward to consider redeployment at other instants, such as just after an ambulance has been dispatched, or at regular time intervals through the day. Simulation results in this vein are given in [28].

4.1.3. *System Dynamics.* We denote the MDP system dynamics as $S_{k+1} = f(S_k, x, U_{k+1})$ where the next state S_{k+1} is a function of the current state S_k, the chosen action $x \in \mathcal{X}(S_k)$, and random effects dictated by U_{k+1}. Define the post-decision state as the immediate state resulting from being in state S_k and applying the effects of action x to the state (before the passage of time). We denote the post-decision state resulting from state S_k and choosing action $x \in \mathcal{X}(S_k)$ as $S_k^+(x)$. When the control taken in state S_k is implicitly understood we use the more concise notation S_k^+. The pre-decision state S_k and post-decision state $S_k^+(x)$ are equal when S_k is a non-decision state. For any decision state S_k recall that the action x corresponds to dispatching the ambulance that became available to Base x. The post decision state $S_k^+(x)$ is therefore equal to S_k, except that if i is the index of the ambulance that just became free (so that $e(S_k) = e_i$), then the redeployment base, i.e., destination, of Ambulance i is $r_i(S_k^+(x)) = x$.

A more complete discussion of post-decision states and associated ADP representations can be found in [33].

Given $S_0^+ \in \mathcal{S}$ with $t(S_0^+) = 0$ we use the following algorithm to construct the DEDS:

1. **(Initialization)** Set the event counter $k = 0$.
2. **(Residual time generation)** For each event $e_i \in E(S_k^+)$ generate a residual time z_i conditional upon the current state S_k^+ where $E(S_k^+) = \{e_0\} \cup \{e_i : c_i(S_k^+) \neq \emptyset \text{ for } i = 1, \ldots, N\}$ denotes the set of active events for state S_k^+, i.e., the set of events that may cause a transition out of state S_k^+. Without loss of generality we assume z_i is generated by inversion from the ith component of U_{k+1} which we

denote $U_{k+1}(i)$. Thus $z_i = F_i^{-1}\big(S_k^+, U_{k+1}(i)\big)$ where $F_i^{-1}\big(S_k^+, \cdot\big)$ is the quantile function for the residual event time for event e_i in state S_k^+.

3. **(Select next event)** Let E_{k+1} be an event with minimal residual time from step 2, and let Δ_{k+1} denote that residual time.
4. **(Select next state)** Generate S_{k+1} conditional upon S_k^+, E_{k+1}, and Δ_{k+1} via U_{k+1}.
5. **(Update clocks)** Set $t(S_{k+1}) = t(S_k) + \Delta_{k+1}$. Set $t_i(S_{k+1}) = 0$ where i is the index of event E_{k+1}. Additionally, set $t_i(S_{k+1}) = 0$ if there is a call arrival in state S_{k+1}, i.e., $E_{k+1} = e_0$, and ambulance i is assigned to respond to the arriving call. For all other clocks set $t_i(S_{k+1}) = t_i(S_k) + \Delta_{k+1}$.
6. **(Redeployment Decision)** Let $S_{k+1}^+ = S_{k+1}$. If S_{k+1} is a decision state then choose a desired ambulance base for redeployment, $x \in \mathcal{X}(S_{k+1})$, for server i and set $R_i(S_{k+1}^+) = x$, where i denotes the server that just became idle, i.e., the index i such that $E_{k+1} = e_i$.
7. **(Repeat)** Set $k = k + 1$ and repeat from Step 2.

Given this DEDS formulation, we define the continuous-time queuing process $S(t)$ of the DEDS for $t \geq 0$ by defining

$$
\begin{aligned}
C(t) &= \sum_{k=0}^{\infty} C(S_k^+) \mathbb{1}_{\{t(S_k) \leq t < t(S_{k+1})\}} \\
A(t) &= \sum_{k=0}^{\infty} A(S_k^+) \mathbb{1}_{\{t(S_k) \leq t < t(S_{k+1})\}} \\
R(t) &= \sum_{k=0}^{\infty} R(S_k^+) \mathbb{1}_{\{t(S_k) \leq t < t(S_{k+1})\}} \\
W(t) &= \sum_{k=0}^{\infty} W(S_k^+) \mathbb{1}_{\{t(S_k) \leq t < t(S_{k+1})\}} \\
e(t) &= \sum_{k=0}^{\infty} e(S_k^+) \mathbb{1}_{\{t(S_k) \leq t < t(S_{k+1})\}}, \text{ and} \\
T(t) &= \left(t, \sum_{k=0}^{\infty} \big(t_0(S_k^+) + t - t(S_k)\big) \mathbb{1}_{\{t(S_k) \leq t < t(S_{k+1})\}}, \ldots, \right. \\
&\qquad \left. \sum_{k=0}^{\infty} \big(t_N(S_k^+) + t - t(S_k)\big) \mathbb{1}_{\{t(S_k) \leq t < t(S_{k+1})\}} \right).
\end{aligned}
$$

By this construction $S(t)$ is piecewise constant in the C, A, R, W, and e components and piecewise linearly increasing in the T component, with

jumps occurring only at event times. Additionally, $S(t)$ is right continuous with left limits.

4.1.4. *Transition Costs and Objective Function.* Given a state S_k, an action $x \in \mathcal{X}(S_k)$, and U_{k+1} we incur a transition cost of $c(S_k, x, U_{k+1})$. Let D be a given threshold time of 8 minutes for satisfactory response times. The transition cost is 1 if an ambulance is assigned to a call which it cannot reach within D minutes and 0 otherwise. In other words,

$$c(S_k, x, U_{k+1}) = \begin{cases} 1 & \text{if } \exists\ i \text{ s.t. } t_i(S_{k+1}) = 0,\ c_i(S_{k+1}) \neq \emptyset, \text{ and} \\ & \quad d(a_i(S_{k+1}), c_i(S_{k+1})) > D \\ 0 & \text{otherwise,} \end{cases}$$

where $t_i(S_{k+1}) = 0$ and $c_i(S_{k+1}) \neq \emptyset$ together indicate that ambulance i has just started responding to a call and $d(a_i(S_{k+1}), c_i(S_{k+1}))$ is the travel time between ambulance i's location at the time of the arrival, $a_i(S_{k+1})$, and the call location $c_i(S_{k+1})$ (including any turn-out time).

We are interested in the expected number of "late calls," or calls not responded to within the time threshold D, over a finite planning horizon T which is usually between one and two weeks. Thus, given the initial state S_0, we use the objective function in (2.1) where $c\left(S_k, \pi(S_k), U_{k+1}\right) = 0$ for all S_k such that $t(S_k) > T$. Discounting future late calls has no clear significance or meaning in an ambulance redeployment context, so $\alpha = 1$ in our experiments.

Our transition cost is chosen both for simplicity and because EMS provider contracts are usually written in terms of the percentage of calls responded to within the given threshold. With this transition-cost definition, there is no incentive for marginal reductions in response times that do not cross the threshold. The work in [18, 6] instead optimizes for "maximum survivability." Although we do not incorporate survivability into our work, the ADP framework is versatile enough to do so. Furthermore, plots of the full response-time distribution in [26] show that ADP policies using the given transition costs reduce the full response-time distribution, and not just those response times that are close to the threshold, at least in the experiments reported there.

4.2. *ADP Policy for Ambulance Redeployment.* To define an ADP policy for ambulance redeployment using the form of (2.3) we must first define the basis functions $\phi_1, \ldots, \phi_B$ and the policy parameters r that form the approximation architecture J_r (Section 4.2.1). By assumption $|\mathcal{X}(s)|$ is not too large for any $s \in \mathcal{S}$, so we can compute the minimization in (2.3) by

calculating the value of the right-hand side for each $x \in \mathcal{X}(s)$ and taking the minimum. We cannot, however, compute the expectation on the right-hand side in closed form, so we estimate it using Monte Carlo simulation (Section 4.2.2).

4.2.1. *Erlang Basis Function.* Our approximation architecture decouples the city into smaller, independent regions each containing only a single ambulance base. Each region is modeled as an Erlang loss system having exponential interarrival time distributions and general service time distributions. The basis functions in our approximation architecture represent the Erlang loss for each of these smaller regions. Thus B represents both the number of basis functions in our approximation architecture and the number of ambulance bases.

Let Γ denote the collection of grid cells into which the call arrival process is discretized. Define $l(\gamma)$ for $\gamma \in \Gamma$ to be the centroid of cell γ and $l(b)$ for $b = 1, \ldots, B$ to be the location of base b. Let $\Gamma_b = \{\gamma \in \Gamma \ : \ d(l(\gamma), l(b)) \leq d(l(\gamma), l(b')) \text{ for all } b' = 1, \ldots, B\}$ denote the set of grid cells that have centroids closer to base b than to any other base.

Define $\lambda(\gamma, t)$ to be the call arrival rate in cell γ at time t and $\Lambda(t) = \sum_{\gamma \in \Gamma} \lambda(\gamma, t)$ to be the total call arrival rate at time t. Let the arrival rate to Γ_b be $\lambda_b(s) = \sum_{\gamma \in \Gamma_b} \lambda(\gamma, t(s))$, and let $n_b(s) = \sum_{i=1}^{N} \mathbb{1}_{\{c_i(s)=\emptyset\}} \mathbb{1}_{\{r_i(s)=l(b)\}}$ denote the number of ambulances either idle at base b or redeploying to base b in state s.

Thus for $b = 1, \ldots, B$, we define the basis function for base b in state s to be the Erlang loss for an $M/G/n_b(s)/n_b(s)$ queue with arrival rate $\lambda_b(s)$ and service rate μ_b weighted according to how likely call arrivals are within Γ_b:

$$\phi_b(s) = \frac{\lambda_b(s)}{\Lambda(t(s))} \frac{(\lambda_b(s)/\mu_b)^{n_b(s)}/n_b(s)!}{\sum_{k=0}^{n_b(s)} (\lambda_b(s)/\mu_b)^k/k!}.$$

The average service time $1/\mu_b$ is a sum of the average response time, scene time, hospital transport time, and hospital transfer time for calls arriving in Γ_b. The scene time and hospital transfer time are generated from distributions with known means. The average response time and hospital transport time are estimated via simulation as the average response time from base b to calls arriving in Γ_b and the average transport time from calls arriving in Γ_b to their destination hospitals respectively. Repeating this procedure for $b = 1, \ldots, B$ we approximate the service rates $\mu_1, \ldots, \mu_b$ prior to running the DEDS and include them as input to the simulation.

The Erlang loss is the steady state proportion of arrivals refused service in a queue having limited queuing capacity. This quantity is relevant for

ambulance redeployment because it can be viewed as an approximation for the steady state proportion of calls that cannot be served by ambulances stationed at the closest base. Such calls must either be placed on a waiting list or served by an ambulance stationed further away. In either situation the response time for the call is likely to increase significantly and the majority of such calls will not be served within the given threshold. For this reason the Erlang loss calculation is related to the value function for a particular state, i.e., the proportion of late calls resulting from being in a given state.

As is common in ADP applications, the basis functions $\phi_1, \ldots, \phi_B$ are not intended to represent the value function exactly. Our approximation architecture ignores the state dependent service rates as well as the more complex dynamics involved when ambulances serve calls outside their respective area. Nevertheless, the basis functions in combination with the tuning coefficients $r_1, \ldots, r_B$ are effective for designing policies that perform well. For example, this approximation architecture is able to significantly improve upon the ambulance redeployment policies in [28] and [26].

4.2.2. *Simulation-Based ADP Policy.* For a given decision state $S_k \in \mathcal{S}$, the ADP policy selects a redeployment base by choosing the redeployment base $x \in \mathcal{X}(S_k)$ that minimizes the right-hand side of (2.3). We cannot compute the expectation in the right-hand side of (2.3) analytically, so we estimate it through Monte Carlo simulation. Let $U_{k+1}^{(1)}, \ldots, U_{k+1}^{(G)}$ denote G iid uniform random vectors of appropriate dimension. We approximate the ADP policy (recall that we use an undiscounted formulation on a finite time horizon) as

$$\operatorname*{argmin}_{x \in \mathcal{X}(S_k)} \frac{1}{G} \sum_{g=1}^{G} \left(c(S_k, x, U_{k+1}^{(g)}) + J_r \left(f(S_k, x, U_{k+1}^{(g)}) \right) \right). \tag{4.1}$$

We call this approach simulation-based ADP due to the use of Monte Carlo simulation, as opposed to other methods that do not need to estimate the expectation via simulation (e.g., the post-decision state policy in Section 6). Techniques such as ranking and selection and common random numbers can also be used to select the minimum instead of using naïve Monte Carlo sampling for each $x \in \mathcal{X}(S_k)$.

5. Simulation Optimization Tuning Results. The limitations of value function fitting methods as illustrated in Section 3 extend beyond sample problems such as that depicted in Figure 1. We consider the problem of tuning the simulation-based ambulance redeployment ADP policy for use in Edmonton, Alberta, Canada. Edmonton is the 5th largest city in

Canada with a population over 700,000. We model the ambulance operations of Edmonton using a discrete event simulation having 16 ambulances, 11 ambulance bases, and 5 hospitals. The call arrival model has a flat arrival rate of 6 calls/hour with a higher density of calls in the metro areas during the day and a higher density of calls in rural areas in the evening and early morning. The travel model used in the simulation is a deterministic shortest-path calculation on a network consisting of major roads in the Edmonton area.

Before proceeding we stress that since the call arrival process, ambulance redeployment policies, and travel network are all stylized, so our simulation results should not be interpreted as indicative of actual performance in Edmonton, nor should the results in Section 6.3 be interpreted in that way. Rather, these computational results showcase the performance of different ADP policies in realistic but not real scenarios with realistic but not real dynamics.

Figure 2 shows the performance of three different tuning methods for the simulation-based ADP policy in Section 4.2.2 using the value function approximation architecture given in Section 4.2.1. Each point in the graph represents an unbiased estimate of policy performance for the ADP policy with respect to L_r for a given set of coefficients r. The coefficients used in each iteration are dictated by the tuning method (based upon the results of previous iterations), and policy performance is estimated from 20 replications of a simulated two-week period of ambulance operations using the specified policy. The estimation of performance at one set of coefficients is considered to be one function evaluation, and the sequence of these function evaluations are indicated along the x-axis. The y-axis gives the estimated percentage of late calls for each policy.

The least-squares method is a value-function fitting method described for infinite-horizon discounted costs in Section 3.1, and we use a version of that method for finite-horizon (two-week) problems here. The Nelder-Mead method is a black box unconstrained local optimization heuristic for deterministic functions [30], and the Noisy UOBYQA algorithm is a derivative-free unconstrained local optimization algorithm based on quadratic approximation adapted from [32] for use with noisy data [16]. Each function evaluation used the same random number seed and simulations were synchronized via the substream functionality of the RngStream random number generator [24].

We chose the initial policy coefficients used by the tuning methods via a "static" policy, or a policy where every ambulance is assigned a home base to which the ambulance returns each time it becomes available. We selected

the static policy π_0 yielding the best performance as estimated through Monte Carlo simulation over a large set of static policies. We then used π_0 to generate 30 two-week sample paths and collected the noisy estimates of the value function for π_0, $\hat{J}^{\pi_0}(s)$, for every s in the sample paths. Given the values of $\hat{J}^{\pi_0}(\cdot)$ from the simulation, we used (3.1) to calculate the policy coefficients $r^*_{\pi_0,2}$ (e.g., a single regression-based iteration) and these coefficients were used as the initial policy coefficients for the tuning methods. We also use the static policy π_0 as a benchmark policy to evaluate the potential benefits of using a redeployment policy over a policy which does not redeploy ambulances dynamically. We do not use discounting, i.e., $\alpha = 1$.

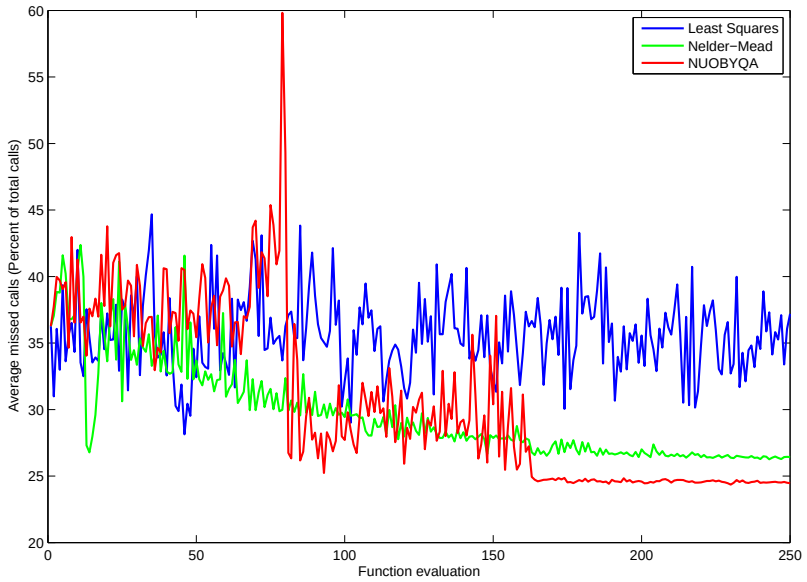

FIG 2. *Results for three methods for tuning ADP coefficients in Edmonton, Canada. Each curve plots the estimated percentage of calls with response times above the threshold for 250 policies that are obtained within a tuning procedure.*

Each iteration along the x-axis of Figure 2 yields a different set of coefficients r, and one can pick the r that gives the best performance. Although each point is an unbiased estimate of performance using the associated coefficients, selecting the coefficients with the best performance introduces a selection bias. Consequently, each "minimizing" policy was reevaluated using independent random number streams to estimate their performance. The performance of these policies, expressed as 95% confidence intervals, are

28.6% ± .1% for least squares, 26.7% ± .1% for Nelder-Mead, and 24.8% ± .1% for NUOBYQA. The performance of the inital static policy was 28.4% ± .1%.

The least squares method found reasonable policies very quickly, but the performance of these policies did not significantly decrease with further tuning. Overall, the least squares method was unable to find a policy with better performance than the initial static policy. The Nelder-Mead method also found a policy with good performance early on; however, ultimately this method converges to a local minimum having inferior performance to the policies found by the NUOBYQA method. The best policies were found with the NUOBYQA method and provide a 3.5% decrease in the percentage of late calls as compared to the initial static policy.

A city-wide decrease of just 1% of late calls represents a practically significant improvement in the context of ambulance redeployment. Simulation experiments not reported here indicate that to obtain a similar improvement using a static policy, one would have to purchase, maintain, and staff an additional ambulance, 24 hours a day and 7 days a week. Hence a reduction of 3.5% is highly significant.

These results may appear to contradict those of [28], where regression successfully identified policies that out-performed the best known static polices. The key difference here is a different approximation architecture. Repeating the same analysis in this paper with the approximation architecture used in [28] shows that all three methods find policies having about a 2-3% improvement over the static policy, consistent with the results of [28].

The time taken for the simulation-based ADP algorithm to make a single redeployment decision is about .07 seconds on a 2.66 Ghz processor, which is easily fast enough for real-time decisions. However, the computational burden of the (off-line) *tuning* procedure is large. Each function evaluation consists of 20 two-week replications of ambulance operations and requires about 1 hour of computation. The total tuning process required nearly 12 days of computation for each method. Since ADP parameters only need to be tuned once before implementing an ADP policy, this is still practically feasible, but these tuning methods are not likely to scale well for larger cities.

There may be situations where the dimension of r or other factors make direct search computationally prohibitive and regression- or LP-based methods must be used. Perhaps because of these extreme cases we believe that simulation-optimization methods are underutilized. This situation persists even though examples such as that in Section 3 and [27] show that there are limitations of standard methods not present in simulation-optimization methods and the gains from using simulation-optimization techniques can

be enormous; see e.g., [42]. In Section 6 we show how the computational disadvantages of direct search can be dramatically reduced in the context of ambulance redeployment in a major metropolitan region.

6. Post-Decision State Formulation. Suppose that for any state S_k we can rewrite the immediate cost function $c(S_k, x, U_{k+1})$ and the system dynamics function $f(S_k, x, U_{k+1})$ in terms of the post-decision state $S_k^+(x)$ as $c(S_k^+(x), U_{k+1})$ and $f(S_k^+(x), U_{k+1})$ respectively. In contrast to Sections 2 and 3 where we worked in discrete time, and in Sections 4 and 5 where we did not discount, here we allow discounting in continuous time. For notational simplicity we denote the discount factor on the interval between the kth event S_k and the $(k+j)$th event S_{k+j}, which is of length $t(S_{k+j}) - t(S_k)$, as $\alpha_j = \alpha^{t(S_{k+j})-t(S_k)}$, where $\alpha \in (0, 1]$.

Let the post-decision value function be defined as

$$\tilde{J}(S_k^+(x)) = \mathbb{E}\left[c(S_k^+(x), U_{k+1}) + \alpha_1 J(f(S_k^+(x), U_{k+1}))\right]. \tag{6.1}$$

Then we have that $J(S_k) = \min_{x \in \mathcal{X}(S_k)} \tilde{J}(S_k^+(x))$.

Let $\tilde{J}_r$ denote a linear approximation architecture for the post-decision state where r represents the set of coefficients, and define

$$\tilde{L}_r(s) = \min_{x \in \mathcal{X}(s)} \tilde{J}_r(s^+(x)) \qquad \forall\ s \in \mathcal{S}. \tag{6.2}$$

The ADP policy with respect to $\tilde{L}_r$ is called the post-decision state ADP policy. The computational benefits of (6.2) over (2.3) are that the expectation operator is contained within the approximation $\tilde{J}_r$. Consequently, this formulation trades the computational burden of Monte Carlo simulation with a heavier reliance on the (post-decision) value function approximation. We elect to use the same basis functions for both the pre- and post-decision state approximations J_r and $\tilde{J}_r$, albeit with unique coefficients.

In Section 6.1 we define a generalization of the ADP policy, in Section 6.2 we show that the post-decision state policy is a limiting case of this generalization, and in Section 6.3 we provide computational results for a direct search tuning method using the post-decision state policy.

6.1. *Truncated Microsimulation Policy.* An ADP policy in state S_k uses (2.3) to choose an action $x \in \mathcal{X}(S_k)$ based upon the expected value of the sum of the transition cost to S_{k+1} and the value function at state S_{k+1} given action x. If the state S_{k+1} is similar to S_k there may be little new information contained in state S_{k+1}. For example, if S_{k+1} corresponds to a call arrival occurring almost immediately after a service completion, then state S_{k+1}

does not contain very much information on the evolution of the system after state S_k. For situations where S_{k+1} corresponds to a non-decision state (no decision is required), (2.3) can be extended to capture more information on the evolution of the system by evaluating the expectation over the time interval not just until the next *state*, but until the next *decision* state.

Let $Q = Q(k, S_k, x)$ be a random variable indicating the number of non-decision states between state S_k and the next decision state given that we choose decision x in the kth state S_k. Then the value function for S_k can be rewritten as

$$J(S_k) = \min_{x \in \mathcal{X}(S_k)} \mathbb{E}\left[\sum_{j=0}^{Q} \alpha_j c(S_{k+j}, x_j, U_{k+j+1}) + \alpha_{Q+1} J\left(f(S_{k+Q}, x_Q, U_{k+Q+1})\right)\right] \quad \forall\, S_k \in \mathcal{S}, \tag{6.3}$$

where $x_0 = x$ and $x_j = \emptyset$ for $1 \leq j < Q$ (see Appendix A). Using this formulation we define

$$L_{r,\infty}(S_k) = \min_{x \in \mathcal{X}(S_k)} \mathbb{E}\left[\sum_{j=0}^{Q} \alpha_j c(S_{k+j}, x_j, U_{k+j+1}) + \alpha_{Q+1} J_r\left(f(S_{k+Q}, x_Q, U_{k+Q+1})\right)\right] \quad \forall\, S_k \in \mathcal{S},$$

and call the ADP policy with respect to $L_{r,\infty}$ the "microsimulation ADP policy." The idea behind the microsimulation ADP policy is that instead of estimating the expectation through Monte Carlo samples of the next state as in (4.1), we use short simulations or "microsimulations" to sample transition costs and future state evolution up until a decision state is reached. One practical drawback of this policy is that it is not known how long each microsimulation must run before reaching a decision state. When large values of Q are observed, the computation may be too large for real-time implementation even if the standard ADP policy can be computed rapidly. To overcome this drawback we select a deterministic time $\tau > 0$ and truncate each microsimulation at τ if it has not already stopped due to reaching a decision state.

The truncation time may occur at a non-event time, so we appeal to the continuous-time version of the DEDS (see Section 4.1.3). Suppose that the kth (pre-decision) state is S_k at time $t(S_k)$ and we take action x at that instant. Let $S_k^+(x) = S(t(S_k))$ denote the post-decision state immediately

after decision x is taken from pre-decision state S_k. Also, let $S_k(\tau, x)$ denote the (random) state, $S(t(S_k) + \tau)$, at time $t(S_k) + \tau$. Finally, let $S_k^+(\tau, x)$ denote the deterministic state that arises at time $t(S_k) + \tau$ when no events occur in the time interval $(t(S_k), t(S_k) + \tau]$.

Let $\gamma_1 = t(S_{k+1}) - t(S_k)$ and $\gamma_{Q+1} = t(S_{Q+1}) - t(S_k)$ denote the time until the next event and the time until the next decision event respectively. Let $Q_\tau = Q_\tau(k, S_k, x)$ be a random variable giving the number of non-decision states between time $t(S_k)$ and the earlier of the next decision state or the threshold time $t(S_k) + \tau$, given that we choose decision x in state S_k. Then the truncated microsimulation value function for all $S_k \in \mathcal{S}$ can be expressed (see Appendix B) as

$$
\begin{aligned}
J(S_k) = &\min_{x \in \mathcal{X}(S_k)} P(\tau \le \gamma_1) \alpha^\tau \tilde{J}(S_k^+(\tau, x)) \\
&+ \mathbb{E}\left[\sum_{j=0}^{Q_\tau} \alpha_j c(S_{k+j}, x_j, U_{k+j+1}) + \alpha^\tau \tilde{J}(S_k(\tau, x)); \gamma_1 < \tau \le \gamma_{Q+1} \right] \\
&+ \mathbb{E}\left[\sum_{j=0}^{Q} \alpha_j c(S_{k+j}, x_j, U_{k+j+1}) + \alpha_{Q+1} J(f(S_{k+Q}, x_Q, U_{k+Q+1})); \gamma_{Q+1} < \tau \right].
\end{aligned}
$$

For all $S_k \in \mathcal{S}$, let the approximate value function be

(6.4)

$$
\begin{aligned}
L_{r,r',\tau}(S_k) = &\min_{x \in \mathcal{X}(S_k)} P(\tau \le \gamma_1) \alpha^\tau \tilde{J}_r(S_k^+(\tau, x)) \\
&+ \mathbb{E}\left[\sum_{j=0}^{Q_\tau} \alpha_j c(S_{k+j}, x_j, U_{k+j+1}) + \alpha^\tau \tilde{J}_r(S_k(\tau, x)); \gamma_1 < \tau \le \gamma_{Q+1} \right] \\
&+ \mathbb{E}\left[\sum_{j=0}^{Q} \alpha_j c(S_{k+j}, x_j, U_{k+j+1}) + \alpha_{Q+1} J_{r'}(f(S_{k+Q}, x_Q, U_{k+Q+1})); \gamma_{Q+1} < \tau \right].
\end{aligned}
$$

The ADP policy with respect to $L_{r,r',\tau}$ is called the truncated microsimulation policy. Two different sets of coefficients r and r' must be used with this policy to account for the approximations of $\tilde{J}$ and J respectively.

6.2. *Limiting Behavior of the Truncated Microsimulation Value Function Approximation.* The truncated microsimulation policy attempts to balance the more precise estimation of longer microsimulations with the computational effort required by such simulations. Since the computational effort required to compute the truncated microsimulation policy generally increases

with the threshold τ it is natural to ask how the policy performs with reduced computational effort, i.e., as τ goes to zero. We show below that the truncated microsimulation policy converges to the post-decision state policy as $\tau \to 0$.

The key idea is that the truncated microsimulation value-function approximation $L_{r,r',\tau}$ should be very close to that of the post-decision value-function approximation $\tilde{L}_r$ when τ is small because the system has had little time to change between the two evaluation points. In essence, the first term in the right-hand side of (6.4) will dominate as $\tau \to 0$. To make this precise, we show pathwise convergence as $\tau \to 0$, and then take expectations through to the limit. The pathwise argument relies on the notion that $t(S_k) + \tau < t(S_{k+1})$ for sufficiently small τ, i.e., that the event times are strictly increasing, almost surely.

There are, however, some states where the event times are not strictly increasing, even if the interarrival and service-time distributions are non-atomic. Indeed, if a clock $t_i(S_k)$ has reached the maximal value of the support of the associated distribution, then an immediate event will occur. Let $\mathcal{B}$ denote the set of states in $\mathcal{S}$ where this happens. The proof of Proposition 1 below is in Appendix C.

PROPOSITION 1. *Assume the interarrival distributions and service-time distributions of the DEDS are non-atomic. If* $P(S_0 \notin \mathcal{B}) = 1$ *then* $P(\exists k : t(S_{k+1}) = t(S_k)) = 0$, *i.e., if* S_0 *is not in* $\mathcal{B}$ *w.p.1 then the probability of any two events occurring at the same time is zero.*

Under the conditions of Proposition 1 the event times are strictly increasing, and so, by bounded convergence, $\lim_{\tau \downarrow 0} P(t(S_k) + \tau < t(S_{k+1})) = 1$ for all $S_k \in \mathcal{S}$.

Assuming J, $\tilde{J}$, and c are bounded, we can bound the two expectations in $L_{r,r',\tau}$ (see (6.4)) by a function of $\mathbb{E}[Q]$, the expected number of non-decision events in $(t(S_k), t(S-k)+\tau)$. We further assume that the expected number of arrivals on any finite interval is finite. The number of events on any interval can be bounded by twice the number of arrivals (one event for the arrival and one event for the service completion) plus a finite constant depending on the initial state. Thus the two expectations in $L_{r,r',\tau}$ must be finite. This is sufficient to state and prove our final result, the proof of which is given in Appendix C. The assumption that J, $\tilde{J}$ and c are bounded can be relaxed, but it is sufficient for our ambulance deployment formulation and it simplifies the proofs.

THEOREM 2. *Assume that $P(S_0 \notin \mathcal{B}) = 1$, that $\tilde{J}_r$, for a fixed r, and c are bounded, that $\tilde{J}_r(\cdot)$ is continuous in $T(\cdot)$ (for any fixed remaining state components), and that the discount factor $\alpha \in (0, 1]$. Then for any bounded ADP approximation architecture $J_{r'}$*

$$\lim_{\tau \downarrow 0} L_{r,r',\tau}(s) = \tilde{L}_r(s) \qquad \forall s \in \mathcal{S}.$$

Thus the function defining the post-decision state policy is the limit of the function defining the truncated microsimulation policy as the truncation time goes to zero (for any bounded ADP approximation architecture).

The key arguments in the proof of Theorem 2 are as follows. Since $L_{r,r',\tau}$ is a minimum over a finite number of convergent sequences we can interchange the order of the limit and the minimum in $L_{r,r',\tau}$. By Proposition 1 and the fact that the expectations of $L_{r,r',\tau}$ are finite,

$$\begin{aligned}
\lim_{\tau \downarrow 0} L_{r,r',\tau}(S_k) &= \min_{x \in \mathcal{X}(S_k)} \lim_{\tau \downarrow 0} \tilde{J}_r(S_k^+(\tau, x)) \\
&= \min_{x \in \mathcal{X}(S_k)} \tilde{J}_r(\lim_{\tau \downarrow 0} S_k^+(\tau, x)) \quad \text{since } \tilde{J}_r \text{ is continuous} \\
&= \min_{x \in \mathcal{X}(S_k)} \tilde{J}_r(S_k^+(x)) \quad \text{by the construction in Section 4.1.3.}
\end{aligned}$$

Thus as the simulation threshold time goes to zero the function defining the truncated microsimulation policy converges to the function defining the post-decision state policy. In this sense we can view the post-decision state policy as a limit of the truncated microsimulation policy, further motivating the post-decision state policy. This interpretation of the post-decision state policy as a limit should hold under very general conditions that are not specific to the ambulance redeployment problem. The essential ingredients are continuity of the value-function approximation, right-continuity of paths, finite action space, and uniform integrability.

The post-decision state ADP policy has the overwhelming computational advantage of not needing to perform Monte Carlo microsimulations to approximate expectations. Given an approximation architecture $\tilde{J}_r$, the only computation required to make a decision is computing the minimum of $\tilde{J}_r$ over a finite (and reasonably small) set. Consequently, tuning post-decision state policies with direct search methods becomes computationally feasible even with much higher dimensional problems than those of Section 5.

6.3. *Computational Results.* Melbourne is the second largest city in Australia with about 4 million residents. Our model has 87 ambulance bases,

22 hospitals, and up to 97 ambulances. The call-arrival model divides Melbourne into a 100×100 grid with time-dependent arrival rates. Based on the geography of Melbourne roughly 80% of these grid cells have no demand. For the remaining 20%, we estimated a typical week of call-arrival rates from historic call data. Throughout the week the daily call arrivals for the entire city vary from about 300 calls/day to 750 calls/day with an average of approximately 570 calls/day. To correspond to the fluctuating demand, ambulance availability is controlled through daily shifts similar to those used in practice with as few as 65 ambulances available during low-demand times and up to 97 ambulances available during peak times. The demand model we use for Melbourne is more realistic than that used for Edmonton, but as with Edmonton, the call-arrival volume and patterns used in our simulation differ significantly from the true demand. By design then, the model we use is realistic but not real, and our results have no reflection on actual ambulance operations in Melbourne.

The travel model used in the simulation is a deterministic shortest-path calculation on a detailed road network with 4,955 nodes and 11,876 arcs. To reduce computation time, the shortest paths between any two nodes are precomputed and stored. To further reduce computation time we perform one-week simulations for Melbourne as compared with the two-week simulations of Edmonton.

The ADP policies used in this section are defined by the post-decision state ADP formulation defined in (6.2) with no discounting ($\alpha = 1$). The parameterized approximation architecture $\tilde{J}_r$ used by the post-decision state ADP has the same form as the approximation architecture J_r (as described in Section 4.2.1), but the coefficients for these two approximation architectures are distinct.

Figure 3 shows the results of the least squares and Nelder-Mead tuning methods for Melbourne using the post-decision state ADP formulation. Since this approximation architecture has one basis function per ambulance base there are 87 basis functions used in the approximation architecture, and the ADP policy parameter tuning problem is an 87 dimensional problem. In this high-dimensional space the NUOBYQA method is computationally infeasible and hence is not included in the results. As in Figure 2, each point along the x-axis represents a single function evaluation at a given set of coefficients, and the y-axis gives the average percentage of missed calls over 30 independent replications of the one-week period. As in Section 5, the performance of the minimizing policy was reevaluated with independent random number streams to eliminate selection bias. The static policy used to generate the initial policy for the tuning approach has 26.7% $\pm$ .1% late

calls. The best policy found by the least squares based fitting procedure has 31.3% ± .1% late calls which is significantly worse than the initial static policy. The Nelder-Mead search method was able to find a policy with 25.8% ± .1% late calls, a practically significant improvement over the static policy of about 0.9%.

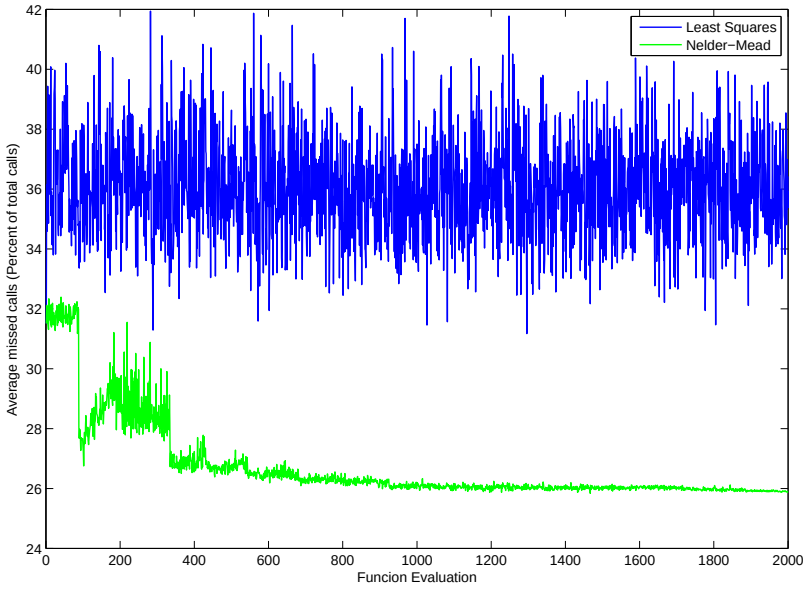

FIG 3. *Results for two methods for tuning ADP coefficients in Melbourne, Australia. Each curve plots the estimated percentage of calls with response times above the threshold for 2000 policies that are obtained within a tuning procedure.*

The direct search tuning in Figure 3 required approximately 12.5 hours of computation time on a 2.66 Ghz processor whereas one function evaluation of the simulation-based ADP policy on Melbourne requires over 5.5 hours. Without the post-decision state formulation such an extensive tuning process would have required over one year of computation time given the same processing capacities. Thus a direct search tuning method coupled with a post-decision state formulation is able to decrease the computational effort used to tune the ADP policy while simultaneously finding superior policies.

7. Conclusion. Ambulance redeployment policies can improve EMS response times within current resources. By altering the positioning of ambulances in real-time EMS providers are better able to respond to gaps in coverage that occur due to inherent uncertainties involved in emergency

services. ADP is an appealing framework for implementing ambulance redeployment because it directly accounts for the randomness in the system while maintaining computational feasibility for real-time decision support.

Unfortunately, standard tuning approaches for ADP policies have limitations that may prevent the resulting ADP policies from performing as well as possible within a given approximation architecture. These limitations were shown both theoretically for a stylized problem and empirically for two different ambulance redeployment scenarios. In both of these scenarios direct search methods were able to find ADP policies with significantly better performance than the best policies found otherwise.

The benefit of direct-search tuning is that it tunes ADP policy parameters based directly on the performance resulting from those parameters, rather than indirectly through value-function fitting. This benefit, however, comes with the cost of higher computational requirements than typically required for other approaches. Using a post-decision state representation the computational burden for tuning associated ADP policies is dramatically reduced.

The post-decision state formulation of a problem may seem foreign and perhaps forced at first, but we show that in our ambulance redeployment setting, the post-decision state ADP policy is actually the limit of a general truncated microsimulation policy which is based on the standard simulation-based ADP formulation. As such, the post-decision state ADP policy can be viewed as the limiting policy of simulation-based ADP policies when the computational budget for simulations goes to zero. While we proved this result only in the present context, the result should hold more generally. Our proof essentially consists of the following steps:

1. Show that as τ, the length of the micro simulations, decreases to 0, the state of the continuous-time Markov process at the end of the micro simulation converges to the post-decision state. It is sufficient that the sample paths following decision epochs are right continuous, as was the case here.
2. Show that the expected value of the "cost-to-go" from time τ onwards converges to the expected value of the cost-to-go immediately after the decision epoch. In general, this will follow if the value function approximation is continuous in the state variables so that pathwise convergence is attained as $\tau \to 0$, and if the appropriate quantities are uniformly integrable. In our setting these quantities were bounded, so uniform integrability was immediate.
3. Ensure that the expected value of the cost-to-go from the post-decision state can be readily computed. In our setting, the post-decision state

was deterministic given the pre-decision state and action, so no averaging was required.

4. Ensure that the optimal action with respect to this post-decision state formulation can be readily computed. In our setting the action space is finite and easily enumerated, so we optimize by complete enumeration.

APPENDIX A: MICROSIMULATION VALUE FUNCTION DERIVATION

We use the notation $E[X; A]$ to denote the expected value of the random variable X on the event A, i.e., $E[XI_A]$, where $I_A(\omega) = 1$ if $\omega \in A$ and 0 otherwise. For all $S_k \in \mathcal{S}$,

$$
\begin{aligned}
J(S_k) &= \min_{x \in \mathcal{X}(S_k)} \mathbb{E}\left[c(S_k, x, U_{k+1}) + \alpha_1 J(f(S_k, x, U_{k+1}))\right] \\
&= \min_{x \in \mathcal{X}(S_k)} \mathbb{E}\left[c(S_k, x, U_{k+1}) + \alpha_1 J(f(S_k, x, U_{k+1})); Q = 0\right] \\
&+ \mathbb{E}\left[c(S_k, x, U_{k+1}) + \alpha_1 J(f(S_k, x, U_{k+1})); Q = 1\right] + \cdots \\
&= \min_{x \in \mathcal{X}(S_k)} \mathbb{E}\left[c(S_k, x, U_{k+1})) + \alpha_1 J(f(S_k, x, U_{k+1})); Q = 0\right] \\
&+ \mathbb{E}[c(S_k, x, U_{k+1}) + \alpha_1 c(S_{k+1}, \emptyset, U_{k+2}) + \alpha_2 J(f(S_{k+1}, \emptyset, U_{k+2})); Q = 1] + \cdots \\
&= \min_{x \in \mathcal{X}(S_k)} \sum_{q=0}^{\infty} \mathbb{E}\left[\sum_{j=0}^{q} \alpha_j c(S_{k+j}, x_j, U_{k+j+1}) + \alpha_{q+1} J(f(S_{k+q}, x_q, U_{k+q+1})); Q = q\right] \\
&= \min_{x \in \mathcal{X}(S_k)} \mathbb{E}\left[\sum_{j=0}^{Q} \alpha_j c(S_{k+j}, x_j, U_{k+j+1}) + \alpha_{Q+1} J(f(S_{k+Q}, x_Q, U_{k+Q+1}))\right],
\end{aligned}
$$

where $x_0 = x$ and $x_j = \emptyset$ for $j \geq 1$.

APPENDIX B: TRUNCATED MICROSIMULATION VALUE FUNCTION DERIVATION

For all $S_k \in \mathcal{S}$

$$
\begin{aligned}
J(S_k) &= \min_{x \in \mathcal{X}(S_k)} \mathbb{E}\left[c(S_k, x, U_{k+1}) + \alpha_1 J(f(S_k, x, U_{k+1}))\right] \\
&= \min_{x \in \mathcal{X}(S_k)} P(\tau \leq \gamma_1) \mathbb{E}\left[c(S_k, x, U_{k+1}) + \alpha_1 J(f(S_k, x, U_{k+1})) \middle| \tau \leq \gamma_1\right] \\
&\quad + \mathbb{E}\left[c(S_k, x, U_{k+1}) + \alpha_1 J(f(S_k, x, U_{k+1})); \gamma_1 < \tau \leq \gamma_{Q+1}\right] \\
&\quad + \mathbb{E}\left[c(S_k, x, U_{k+1}) + \alpha_1 J(f(S_k, x, U_{k+1})); \gamma_{Q+1} < \tau\right].
\end{aligned}
$$

Consider the conditional expectation above. There is zero cost associated with the transition from the pre-decision state S_k to the post-decision state

$S_k^+(x)$, and on to state $S_k(\tau, x)$. We then incur, by definition, the cost $c(S_k(\tau, x), U_{k+1})$ from then till state S_{k+1} is reached, which must be discounted by α^τ. Finally, we have the discounted value function at state S_{k+1} (by definition of $f(\cdot, \cdot)$). Thus,

$$\begin{aligned}
&\mathbb{E}\left[c(S_k, x, U_{k+1}) + \alpha_1 J(f(S_k, x, U_{k+1}))\middle|\tau \le \gamma_1\right] \\
&= \mathbb{E}\left[\alpha^\tau c(S_k(\tau, x), U_{k+1}) + \alpha^\tau J(f(S_k(\tau, x), U_{k+1}))\middle|\tau \le \gamma_1\right] \\
&= \alpha^\tau \mathbb{E}\left[\tilde{J}(S_k(\tau, x))\middle|\tau \le \gamma_1\right] \qquad \text{by (6.1)} \\
&= \alpha^\tau \tilde{J}(S_k^+(\tau, x)),
\end{aligned}$$

where the last equality holds because on the event $\tau \le \gamma_1$, $S_k(\tau, x) = S_k^+(\tau, x)$ and $\tilde{J}(S_k^+(\tau, x))$ is a deterministic function of S_k, τ, and x.

Similarly,

$$\begin{aligned}
&\mathbb{E}\left[c(S_k, x, U_{k+1}) + \alpha_1 J(f(S_k, x, U_{k+1})); \gamma_1 < \tau \le \gamma_{Q+1}\right] \\
&\quad = \mathbb{E}\left[\textstyle\sum_{j=0}^{Q_\tau} \alpha_j c(S_{k+j}, x_j, U_{k+j+1}) + \alpha^\tau \tilde{J}(S_k(\tau, x)); \gamma_1 < \tau \le \gamma_{Q+1}\right].
\end{aligned}$$

Thus, for all $S_k \in \mathcal{S}$,

$$\begin{aligned}
J(S_k) = &\min_{x \in \mathcal{X}(S_k)} P(\tau \le \gamma_1)\alpha^\tau \tilde{J}(S_k^+(\tau, x)) \\
&+ \mathbb{E}\left[\sum_{j=0}^{Q_\tau} \alpha_j c(S_{k+j}, x_j, U_{k+j+1}) + \alpha^\tau \tilde{J}(S_k(\tau, x)); \gamma_1 < \tau \le \gamma_{Q+1}\right] \\
&+ \mathbb{E}\left[\sum_{j=0}^{Q} \alpha_j c(S_{k+j}, x_j, U_{k+j+1}) + \alpha_{Q+1} J(f(S_{k+Q}, x_Q, U_{k+Q+1})); \gamma_{Q+1} < \tau\right]
\end{aligned}$$

where the last line follows from (6.3).

APPENDIX C: PROOF OF THEOREM 2

We first prove Proposition 1. Let $I(s) = \{i : e_i \in E(s)\}$ denote the index set of active events $E(s)$. For any $s \in \mathcal{S}$ let $M_i(s)$ for $e_i \in E(s)$ denote the (possibly infinite) supremum of the support of the residual time distribution for event e_i. Recall that $\mathcal{B}$ is the set of states where an event occurs immediately. More formally, define

$$\mathcal{B} = \{s \in \mathcal{S} : \exists\ e_i \in E(s) \text{ where } M_i(s) = 0\}.$$

LEMMA 3. *If* $P(S_k \notin \mathcal{B}) = 1$ *then*

$$P(t(S_{k+1}) = t(S_k)) = 0,$$

i.e., if S_k *is not in* $\mathcal{B}$ *w.p.1, then the probability of getting an immediate event on the next transition is zero.*

PROOF. Recall that $F_i^{-1}(S_k, \cdot)$ denotes the quantile function for the residual event time for event e_i in state S_k. We have

$$\begin{aligned} P(t(S_{k+1}) = t(S_k)) &= P(t(S_{k+1}) = t(S_k) | S_k \notin \mathcal{B}) P(S_k \notin \mathcal{B}) \\ &\quad + P(t(S_{k+1}) = t(S_k) | S_k \in \mathcal{B}) P(S_k \in \mathcal{B}) \\ &= P(t(S_{k+1}) = t(S_k) | S_k \notin \mathcal{B}) \qquad \text{by assumption} \\ &= P\left(\min_{i \in I(S_k)} F_i^{-1}(S_k, U_{k+1}(i)) = 0 \middle| S_k \notin \mathcal{B} \right). \end{aligned}$$

Conditional upon $I(S_k) = I$, we have

$$\begin{aligned} &P\Big(\min_{i \in I} F_i^{-1}(S_k, U_{k+1}(i)) = 0 \Big| S_k \notin \mathcal{B}, I(S_k) = I \Big) \\ &\qquad \leq \sum_{i \in I} P\left(F_i^{-1}(S_k, U_{k+1}(i)) = 0 \middle| S_k \notin \mathcal{B}, I(S_k) = I \right) \\ &\qquad = \sum_{i \in I} P\left(U_{k+1}(i) = 0 \right) \\ &\qquad = 0, \end{aligned}$$

where the second to last equality holds due to the fact that $S_k \notin \mathcal{B}$ which implies that for each $i \in I$ the residual time distribution is non-atomic and hence $F_i^{-1}(S_k, U_{k+1}(i)) = 0$ if and only if $U_{k+1}(i) = 0$.

Since there are a finite number of realizations of $I(\cdot)$ we have that

$$P\left(\min_{i \in I(S_k)} F_i^{-1}(S_k, U_{k+1}(i)) = 0 \middle| S_k \notin \mathcal{B} \right) = 0,$$

and the desired result holds. □

LEMMA 4. *If* $P(S_k \notin \mathcal{B}) = 1$ *then*

$$P(S_{k+1} \notin \mathcal{B}) = 1,$$

i.e., if S_k *is not in* $\mathcal{B}$ *w.p.1 then* S_{k+1} *is not in* $\mathcal{B}$ *w.p.1.*

PROOF. We have

$$\begin{aligned}P(S_{k+1} \in \mathcal{B}) &= P(S_{k+1} \in \mathcal{B}|S_k \notin \mathcal{B})P(S_k \notin \mathcal{B}) + P(S_{k+1} \in \mathcal{B}|S_k \in \mathcal{B})P(S_k \in \mathcal{B}) \\ &= P(S_{k+1} \in \mathcal{B}|S_k \notin \mathcal{B}) \qquad \text{by assumption} \\ &= P(\exists\, i \in I(S_{k+1}) \text{ such that } M_i(S_{k+1}) = 0|S_k \notin \mathcal{B}) \\ &\leq P\left(\exists\, i \in I(S_k) \text{ such that } F_i^{-1}\big(S_k, U_{k+1}(i)\big) = M_i(S_k)|S_k \notin \mathcal{B}\right).\end{aligned}$$

The third equality holds by definition of $\mathcal{B}$ and the last line holds because a necessary condition to satisfy $M_i(S_{k+1}) = 0$ is that $F_i^{-1}\big(S_k, U_{k+1}(i)\big)$ be equal to $M_i(S_k)$. Conditional upon $I(S_k) = I$, we have

$$\begin{aligned}&P\left(\exists\, i \in I \text{ such that } F_i^{-1}\big(S_k, U_{k+1}(i)\big) = M_i(S_k)|S_k \notin \mathcal{B}, I(S_k) = I\right) \\ &\qquad \leq \sum_{i \in I} P\left(F_i^{-1}\big(S_k, U_{k+1}(i)\big) = M_i(S_k)|S_k \notin \mathcal{B}, I(S_k) = I\right) \\ &\qquad = \sum_{i \in I} P\left(U_{k+1}(i) = 1\right) \\ &\qquad = 0,\end{aligned}$$

where the second to last equality holds due to the fact that $S_k \notin \mathcal{B}$ which implies that for each $i \in I$ the residual time distribution is non-atomic and hence $F_i^{-1}\big(S_k, U_{k+1}(i)\big) = M_i(S_k)$ if and only if $U_{k+1}(i) = 1$.

Since there are a finite number of realizations of $I(\cdot)$ we have that

$$P\left(\exists\, i \in I(S_k) \text{ such that } F_i^{-1}\big(S_k, U_{k+1}(i)\big) = M_i(S_k)|S_k \notin \mathcal{B}\right) = 0,$$

and the desired result holds. □

PROOF OF PROPOSITION 1. Given $P(S_0 \notin \mathcal{B}) = 1$ and Lemma 4 we know that $P(S_k \notin \mathcal{B}) = 1$ for all k by induction. Thus by Lemma 3 we know that $P(t(S_{k+1}) = t(S_k)) = 0$ for all k.

The number of events is bounded by twice the number of arrivals (one for arrival and one for service completion) plus a constant factor depending on the initial state S_0. Since the number of arrivals are countable the number of events are also countable, and we have that

$$\begin{aligned}P\left(\exists k : t(S_{k+1}) = t(S_k)\right) &\leq \sum_{k=0}^{\infty} P\left(t(S_{k+1}) = t(S_k)\right) \\ &= 0.\end{aligned}$$

□

We now turn to the proof of Theorem 2.

LEMMA 5. *If the value function approximation is bounded, i.e.,* $|J_{r'}(s)| \leq H_J < \infty$ *and the costs are bounded, i.e.,* $|c(s,x,\cdot)| \leq H_c < \infty$ *for all* $s \in \mathcal{S}$, $x \in \mathcal{X}(s)$, *then*

$$\lim_{\tau \downarrow 0} \mathbb{E}\Bigg[\Big| \sum_{j=0}^{Q} \alpha_j c(S_{k+j}, x_j, U_{k+j+1}) + \alpha_{Q+1} J_{r'}(f(S_{k+Q}, x_Q, U_{k+Q+1}))\Big| \, ; \gamma_{Q+1} < \tau\Bigg] = 0,$$

where $x_0 \in \mathcal{X}(S_k)$ *and* $x_j = \emptyset$ *for* $1 \leq j \leq Q$.

PROOF. By the boundedness assumptions,

$$\Big| \sum_{j=0}^{Q} \alpha_j c(S_{k+j}, x_j, U_{k+j+1}) + \alpha_{Q+1} J_{r'}(f(S_{k+Q}, x_Q, U_{k+Q+1}))\Big| \leq \sum_{j=0}^{Q} H_c + H_J.$$

The quantity Q is bounded by the total number of events in $(t(S_k), t(S_k)+\tau)$ which is bounded by twice the number of arrivals in $(t(S_k), t(S_k)+\tau)$ (one for arrival, one for service completion) plus a constant number of events depending upon the state S_k. The expected number of arrivals in any finite time period is finite, so the dominating random variable $H_c(Q+1) + H_J$ has finite expectation. Dominated convergence completes the proof, since $\gamma_{Q+1} \geq \tau$ eventually. □

Using exactly the same technique we can show that if $\tilde{J}_r$ and the costs are both bounded, then

$$\lim_{\tau \downarrow 0} \mathbb{E}\left[\Big| \sum_{j=0}^{Q_\tau} \alpha_j c(S_{k+j}, x_j, U_{k+j+1}) + \alpha^\tau \tilde{J}_r(S_k(\tau, x))\Big|; \gamma_1 < \tau \leq \gamma_{Q+1}\right] = 0,$$

where $x_0 \in \mathcal{X}(S_k)$ and $x_j = \emptyset$ for $1 \leq j \leq Q_\tau$.

PROOF OF THEOREM 2. We have that

(C.1)

$$\begin{aligned}
\lim_{\tau\downarrow 0} L_{r,r',\tau}(S_k) &= \lim_{\tau\downarrow 0}\Bigg(\min_{x\in\mathcal{X}(S_k)} P(\tau\le\gamma_1)\alpha^\tau \tilde{J}_r(S_k^+(\tau,x))\\
&+\mathbb{E}\Bigg[\sum_{j=0}^{Q_\tau}\alpha_j c(S_{k+j},x_j,U_{k+j+1})+\alpha^\tau \tilde{J}_r(S_k(\tau,x));\gamma_1<\tau\le\gamma_{Q+1}\Bigg]\\
&+\mathbb{E}\Bigg[\sum_{j=0}^{Q}\alpha_j c(S_{k+j},x_j,U_{k+j+1})+\alpha_{Q+1}J_{r'}(f(S_{k+Q},x_Q,U_{k+Q+1}));\gamma_{Q+1}<\tau\Bigg]\Bigg)\\
&=\min_{x\in\mathcal{X}(S_k)}\lim_{\tau\downarrow 0}\Bigg(P(\tau\le\gamma_1)\alpha^\tau \tilde{J}_r(S_k^+(\tau,x))\\
&+\mathbb{E}\Bigg[\sum_{j=0}^{Q_\tau}\alpha_j c(S_{k+j},x_j,U_{k+j+1})+\alpha^\tau \tilde{J}_r(S_k(\tau,x));\gamma_1<\tau\le\gamma_{Q+1}\Bigg]\\
&+\mathbb{E}\Bigg[\sum_{j=0}^{Q}\alpha_j c(S_{k+j},x_j,U_{k+j+1})+\alpha_{Q+1}J_{r'}(f(S_{k+Q},x_Q,U_{k+Q+1}));\gamma_{Q+1}<\tau\Bigg]\Bigg).
\end{aligned}$$

The interchange of the minimum and limit above is justified because we have a minimization over a finite number of convergent sequences and hence the sequence of the minimums converges uniformly. Interchange of the minimum and limit for uniformly convergent sequences is justified in, for example, [38, Proposition 5].

From Proposition 1, $\lim_{\tau\downarrow 0} P(\tau\le\gamma_1)=1$ and we have already shown that the remaining terms converge to 0. Thus we have that

$$\begin{aligned}
\lim_{\tau\downarrow 0} L_{r,r',\tau}(S_k) &= \min_{x\in\mathcal{X}(S_k)}\lim_{\tau\downarrow 0}\tilde{J}_r(S_k^+(\tau,x))\\
&=\min_{x\in\mathcal{X}(S_k)}\tilde{J}_r\Big(\lim_{\tau\downarrow 0}S_k^+(\tau,x)\Big) \qquad \text{(C.2)}\\
&=\min_{x\in\mathcal{X}(S_k)}\tilde{J}_r\left(S_k^+(x)\right) \qquad \text{(C.3)}\\
&=\tilde{L}_r(S_k),
\end{aligned}$$

where (C.2) follows because the T component of $S_k^+(\tau,x)$ is the only component that varies as $\tau\downarrow 0$ (see Section 4.1.3) and $\tilde{J}_r(\cdot)$ is continuous in $T(\cdot)$ by assumption, and (C.3) follows from the definition of the continuous-time system dynamics in Section 4.1.3. □

ACKNOWLEDGMENTS

We thank the associate editor for extremely helpful comments that improved the paper. This research was supported in part by NSF Grants CMMI 0758441 and CMMI 1200315.

REFERENCES

[1] D. Adelman. A price-directed approach to stochastic inventory routing. *Operations Research*, 52(4):499–514, 2004.

[2] D. Adelman. Dynamic bid-prices in revenue management. *Operations Research*, 55(4):647–661, 2007.

[3] R. Alanis, A. Ingolfsson, and B. Kolfal. A Markov chain model for an EMS system with repositioning, 2010.

[4] T. Andersson. *Decision support tools for dynamic fleet management.* PhD thesis, Department of Science and Technology, Linkoepings Universitet, Norrkoeping, Sweden, 2005.

[5] T. Andersson and P. Vaerband. Decision support tools for ambulance dispatch and relocation. *Journal of the Operational Research Society*, 58:195–201, 2007.

[6] D. Bandar, M. E. Mayorga, and L. A. McLay. Optimal dispatching strategies for emergency vehicles to increase patient survivability. *To appear, International Journal of Operational Research*, 2012.

[7] O. Berman. Dynamic repositioning of indistinguishable service units on transportation networks. *Transportation Science*, 15(2), 1981.

[8] O. Berman. Repositioning of distinguishable urban service units on networks. *Computers and Operations Research*, 8:105–118, 1981.

[9] O. Berman. Repositioning of two distinguishable service vehicles on networks. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11(3), 1981.

[10] D. Bertsekas. *Dynamic Programming and Optimal Control.* Athena Scientific, Nashua, NH, 2005.

[11] D. Bertsekas and S. Shreve. *Stochastic Optimal Control: The Discrete Time Case.* Academic Press, New York, 1978.

[12] D. Bertsekas and J. Tsitsiklis. *Neuro-Dynamic Programming.* Athena Scientific, Belmont, Massachusetts, 1996.

[13] J. A. Boyan. Technical update: Least-squares temporal difference learning. *Machine Learning*, 49(2):233–246, 2002.

[14] S. J. Bradtke, A. G. Barto, and P. Kaelbling. Linear least-squares algorithms for temporal difference learning. In *Machine Learning*, pages 22–33, 1996.

[15] D. P. de Farias and B. Van Roy. The linear programming approach to approximate dynamic programming. *Operations Research*, 51:2003, 2001.

[16] G. Deng and M. C. Ferris. Adaptation of the UOBYQA algorithm for noisy functions. In *WSC '06: Proceedings of the 38th conference on Winter simulation*, pages 312–319. Winter Simulation Conference, 2006.

[17] V. V. Desai, V. F. Farias, and C. C. Moallemi. Approximate dynamic programming via a smoothed linear program. *To appear in Operations Research*, 2012.

[18] E. Erkut, A. Ingolfsson, and G. Erdoğan. Ambulance deployment for maximum survival. *Naval Research Logistics*, 55(1):42–58, 2007.

[19] V. F. Farias and B. Van Roy. An approximate dynamic programming approach to network revenue management. Technical report, Stanford University, Department of Electrical Engineering, 2007.

[20] M. Gendreau, G. Laporte, and S. Semet. A dynamic model and parallel tabu search heuristic for real time ambulance relocation. *Parallel Computing*, 27:1641–1653, 2001.

[21] M. Gendreau, G. Laporte, and S. Semet. The maximal expected coverage relocation problem for emergency vehicles. *Journal of the Operational Research Society*, 57:22–28, 2006.

[22] P. W. Glynn. A GSMP formalism for discrete event systems. *Proceedings of the IEEE*, 77(1), 1989.

[23] P. Kolesar and W. E. Walker. An algorithm for the dynamic relocation of fire companies. *Operations Research*, 22(2):249–274, 1974.

[24] P. L'Ecuyer, R. Simard, E. J. Chen, and W. D. Kelton. An object-oriented random-number package with many long streams and substreams. *Operations Research*, 50(6):1073–1075, 2002.

[25] F. A. Longstaff and E. S. Schwartz. Valuing American options by simulation: A simple least-squares approach. *The Review of Financial Studies*, 14(1):113–147, 2001.

[26] M. S. Maxwell, S. G. Henderson, and H. Topaloglu. Ambulance redeployment: An approximate dynamic programming approach. In M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, editors, *Proceedings of the 2009 Winter Simulation Conference*, pages 1850–1860, Piscataway, New Jersey, 2009. Institute of Electrical and Electronics Engineers, Inc.

[27] M. S. Maxwell, S. G. Henderson, and H. Topaloglu. Identifying effective policies in approximate dynamic programming: Beyond regression. In B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, editors, *Proceedings of the 2010 Winter Simulation Conference*, Piscataway, New Jersey, 2010. Institute of Electrical and Electronics Engineers, Inc.

[28] M. S. Maxwell, M. Restrepo, S. G. Henderson, and H. Topaloglu. Approximate dynamic programming for ambulance redeployment. *INFORMS Journal on Computing*, 22(2):266–281, 2010.

[29] R. Nair and E. Miller-Hooks. Evaluation of relocation strategies for emergency medical service vehicles. *Transportation Research Record: Journal of the Transportation Research Board*, 2137:63–73, 2009.

[30] J. Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[31] K. Papadaki and W. B. Powell. An adaptive dynamic programming algorithm for a stochastic multiproduct batch dispatch problem. *Naval Research Logistics*, 50(7):742–769, 2003.

[32] M. Powell. UOBYQA: unconstrained optimization by quadratic approximation. *Mathematical Programming*, 92(3):555–582, 2002.

[33] W. B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. John Wiley & Sons, Hoboken, NJ, 2007.

[34] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Hoboken, NJ, 2005.

[35] D. Richards. *Optimised Ambulance Redeployment Strategies*. PhD thesis, The University of Auckland, Auckland, New Zealand, 2007.

[36] V. Schmid. Solving the dynamic ambulance relocation and dispatching problem using approximate dynamic programming. *European Journal of Operational Research*, 219:611–621, 2012.

[37] P. J. Schweitzer and A. Seidmann. Generalized polynomial approximations in Markovian decision processes. *Journal of Mathematical Analysis and Applications*, 110(2):568 – 582, 1985.

[38] A. Shapiro. Monte Carlo sampling methods. In A. Ruszczynski and A. Shapiro, editors, *Stochastic Programming*, volume 10 of *Handbooks in Operations Research and Management Science*, pages 353 – 425. Elsevier, 2003.

[39] M. Z. Spivey and W. B. Powell. The dynamic assignment problem. *Transportation Science*, 38(4):399–419, 2004.

[40] R. S. Sutton. Learning to predict by the methods of temporal differences. In *Machine Learning*, pages 9–44. Kluwer Academic Publishers, 1988.

[41] R. S. Sutton and A. G. Barto. *Reinforcement Learning*. The MIT Press, Cambridge, MA, 1998.

[42] I. Szita and A. Lörincz. Learning tetris using the noisy cross-entropy method. *Neural Computation.*, 18(12):2936–2941, 2006.

[43] G. Tesauro. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6(2):215–219, 1994.

[44] H. Topaloglu and W. B. Powell. Dynamic programming approximations for stochastic, time-staged integer multicommodity flow problems. *INFORMS Journal on Computing*, 18(1):31–42, 2006.

[45] J. Tsitsiklis and B. Van Roy. Regression methods for pricing complex American-style options. *IEEE Transactions on Neural Networks*, 12(4):694–703, 2001.

[46] B. Van Roy, D. P. Bertsekas, Y. Lee, and J. N. Tsitsiklis. A neuro dynamic programming approach to retailer inventory management. In *Proceedings of the IEEE Conference on Decision and Control*, 1997.

[47] L. Zhang. *Simulation Optimisation and Markov Models for Dynamic Ambulance Redeployment*. PhD thesis, The University of Auckland, Auckland, New Zealand, 2012.

[48] L. Zhang, A. Mason, and A. Philpott. Optimization of a single ambulance move up. Preprint, 2010.

School of Operations Research and Information Engineering
Ithaca NY 14853
E-mail: msm57@cornell.edu
sgh9@cornell.edu
ht88@cornell.edu