Approximate Dynamic Programming for Ambulance Redeployment

Mateo Restrepo
Center for Applied Mathematics
Cornell University, Ithaca, NY 14853, USA
mr324@cornell.edu

Shane G. Henderson, Huseyin Topaloglu
School of Operations Research and Information Engineering
Cornell University, Ithaca, NY 14853, USA
{sgh9,ht88}@cornell.edu

May 5, 2008

Abstract

We present an approximate dynamic programming approach for making ambulance redeployment decisions in an emergency medical service system. The primary decision is where we should redeploy idle ambulances so as to maximize the number of calls reached within a given delay threshold. We begin by formulating this problem as a dynamic program. To deal with the high-dimensional and uncountable state space in the dynamic program, we construct approximations to the value functions that are parameterized by a small number of parameters. We tune the parameters of the value function approximations using simulated cost trajectories of the system. Computational experiments demonstrate the performance of the approach on emergency medical service systems in two metropolitan areas. We report practically significant improvements in performance relative to benchmark static policies.

1. Introduction

Rising costs of medical equipment, increasing call volumes and worsening traffic conditions are putting emergency medical service (EMS) managers under pressure to meet performance goals set by regulators or contracts. Ambulance redeployment is one possible approach that can potentially help EMS managers. Ambulance redeployment, also known as relocation, move up, system-status management or dynamic repositioning, refers to any strategy by which a dispatcher repositions idle ambulances to compensate for others that are busy, and hence, unavailable. The increasing availability of geographic information systems and the increasing affordability of computing power have finally created ideal conditions for bringing real-time ambulance redeployment approaches to fruitful implementation.

In this paper, we present an approximate dynamic programming (ADP) approach for making real-time ambulance redeployment decisions. We begin by formulating the ambulance redeployment problem as a dynamic program. This dynamic program involves a high-dimensional and uncountable state space and we address this difficulty by constructing approximations to the value functions that are parameterized by a small number of parameters. We tune the parameters of the value function approximations through an iterative and simulation-based method. Each iteration of the simulation-based method consists of two steps. In the first step, we simulate the trajectory of the greedy policy induced by the current value function approximation and collect cost trajectories of the system. In the second step, we tune the parameters of the value function approximation by solving a regression problem that fits the value function approximation to the collected cost trajectories. This yields a new set of parameters that characterize new value function approximations, and so, we can go back and repeat the same two steps. In this respect, the idea we use closely resembles the classical policy iteration algorithm in the Markov decision process literature. In particular, the first (second) step is analogous to the policy-evaluation (policy-improvement) step of the policy iteration algorithm.

There are two streams of literature that are related to our work. The first one is the literature on ADP. A generic approach for ADP involves using value function approximations of the form $\sum_{p=1}^{P} r_p \, \phi_p(\cdot)$, where $\{r_p : p = 1, \ldots, P\}$ are tuneable parameters and $\{\phi_p(\cdot) : p = 1, \ldots, P\}$ are fixed basis functions; see Bertsekas and Tsitsiklis (1996), Powell (2007). There are a number of methods to tune the parameters $\{r_p : p = 1, \ldots, P\}$ so that $\sum_{p=1}^{P} r_p \, \phi_p(\cdot)$ yields a good approximation to the value function. For example, temporal difference learning and Q-learning use stochastic approximation ideas in conjunction with simulated trajectories of the system to iteratively tune the parameters; see Sutton (1988), Watkins and Dayan (1992), Tsitsiklis (1994), Bertsekas and Tsitsiklis (1996), Tsitsiklis and Van Roy (1997), Si, Barto, Powell, and Wunsch II (2004). On the other hand, the linear-programming approach for ADP finds a good set of values for the parameters by solving a large linear program whose decision variables are $\{r_p : p = 1, \ldots, P\}$; see Schweitzer and Seidmann (1985), de Farias and Van Roy (2003), Adelman and Mersereau (2007). Both classes of approaches are aimed at tuning the parameters $\{r_p : p = 1, \ldots, P\}$ so that $\sum_{p=1}^{P} r_p \, \phi_p(\cdot)$ yields a good approximation to the value function. The choice of the basis functions $\{\phi_p(\cdot) : p = 1, \ldots, P\}$, on the other hand, is regarded as more of an art form, requiring substantial knowledge of the problem structure. Applications of ADP include inventory control (Van Roy, Bertsekas, Lee, and Tsitsiklis, 1997), inventory routing (Adelman, 2004), option pricing (Tsitsiklis and Van Roy, 2001), game playing (Yan, Diaconis, Rusmevichientong, and Van Roy, 2005; Farias and Van Roy, 2006), dynamic fleet management (Topaloglu and Powell, 2006), and network revenue management (Adelman, 2007; Farias and Van Roy, 2007).

The second stream of literature that is related to our work is the literature on ambulance redeployment. One class of models involves solving integer programs in real time whenever an ambulance redeployment decision needs to be made; see Kolesar and Walker (1974), Gendreau, Laporte, and Semet (2001), Brotcorne, Laporte, and Semet (2003), Gendreau, Laporte, and Semet (2006), Nair and Miller-Hooks (2006), Richards (2007). The objective function in these integer programs generally involves a combination of backup coverage for future calls and relocation cost of ambulances. These models are usually computationally intensive, since they require solving an optimization problem every time a decision is made. As a result, a parallel computing environment is often (but not always) used to implement a working real-time system. A second class of models is based on solving integer programs in a preparatory phase. This approach provides a lookup table describing, for each number of available ambulances, where those ambulances should be deployed. Dispatchers attempt to dispatch so as to keep the ambulance configuration close to the one suggested by the lookup table; see Ingolfsson (2006), Goldberg (2007). A third class of models attempts to capture the randomness in the system explicitly, either through a dynamic programming formulation or through heuristic approaches. Berman (1981a), Berman (1981b) and Berman (1981c) represent the first papers that provide a dynamic programming approach for the ambulance redeployment problem. However, these papers follow an exact dynamic programming formulation and, as is often the case, this formulation is tractable only in oversimplified versions of the problem with few vehicles and small transportation networks. Andersson (2005) and Andersson and Vaerband (2007) make the ambulance redeployment decision by using a "preparedness" function that essentially measures the capability of a certain ambulance configuration to cover future calls. The preparedness function is similar in spirit to the value function in a dynamic program, measuring the impact of current decisions on the future evolution of the system. However, the way the preparedness function is constructed is heuristic in nature.

When compared with the three classes of models described above, our approach provides a number of advantages. In contrast to the models that are based on integer programs, our approach captures the random evolution of the system over time since it is based on a dynamic programming formulation of the ambulance redeployment problem. Furthermore, the decisions made by our approach in real time can be computed very quickly as this requires solving a simple optimization problem that minimizes the sum of the immediate cost and the value function approximation. In lookup table approaches, there may be more than one way to redeploy the ambulances so that the ambulance configuration over the transportation network matches the configuration suggested by the lookup table. Therefore, table lookup approaches still leave some aspects of dispatch decisions to subjective interpretation by dispatchers. Our approach, on the other hand, can fully automate the decision-making process. In traditional dynamic-programming approaches, one is usually limited to very small problem instances, whereas ADP can be used on problem instances with realistic dimensions. Our approach allows working with a variety of objective functions, such as the number of calls that are not served within a threshold time standard or the total response time for the calls. Furthermore, our approach allows the possibility of constraining the frequency and destinations of ambulance relocations. This is important since a relocation scheme should balance improvements in service levels with the additional demands imposed on ambulance crews.

In summary, we make the following research contributions. 1) We develop a tractable ADP approach for the ambulance redeployment problem. Our approach employs value function approximations of the form $\sum_{p=1}^{P} r_p \, \phi_p(\cdot)$ and uses sampled cost trajectories of the system to tune the parameters $\{r_p : p = 1, \ldots, P\}$. Since it is based on the dynamic programming formulation of the problem, our approach is able to capture the random evolution of the system over time. 2) We develop a set of basis functions $\{\phi_p(\cdot) : p = 1, \ldots, P\}$ that yield good value function approximations for the ambulance redeployment problem. This opens up the possibility of using other ADP approaches, such as temporal difference learning and the linear-programming approach. 3) We provide computational experiments on EMS systems in two metropolitan areas. Our results indicate that ADP has the potential to obtain high-quality redeployment policies in real systems. They also show that our approach compares favorably with benchmark policies that are similar to those used in practice.

The remainder of this paper is organized as follows. In Section 2, we present a dynamic programing formulation for the ambulance redeployment problem. In Section 3, we describe our ADP approach. In Section 4, we discuss the basis functions that we use in our value function approximations. In Sections 5 and 6, we report computational results for two metropolitan areas. We conclude in Section 7 and point out some directions for further research.

2. Ambulance Redeployment as a Markov Decision Process

In this section, we present a dynamic programming formulation of the ambulance redeployment problem. As will shortly be clear, our model involves an uncountable state space. For an excellent account of the basic terminology, notation and fundamental results regarding dynamic programming in uncountable states spaces, we refer the reader to Bertsekas and Shreve (1978).

2.1. State Space

Two main components in the state of an EMS system at a given time are the vectors $\mathbf{A} = (\mathbf{a}_1, \ldots, \mathbf{a}_N)$ and $\mathbf{C} = (\mathbf{c}_1, \ldots, \mathbf{c}_M)$ containing information about the state of each ambulance and each waiting call in the system. To simplify the presentation, we assume that we do not keep more than M waiting calls in the system, possibly by diverting them to another EMS system. This is not really a restriction from a practical perspective since M can be quite large. The state of each ambulance is given by a tuple $\mathbf{a} = (\sigma_a, o_a, d_a, t_a)$, where σ_a is the status of the ambulance, o_a and d_a are respectively origin and destination locations of the ambulance and t_a is the starting time of the ambulance movement. To serve a call, an ambulance first moves to the call scene and provides service at the scene for a certain amount of time. Following this, the ambulance possibly transports the patient to a hospital, and after spending some time at the hospital, the ambulance becomes free to serve another call. Therefore, the status of an ambulance σ_a can be "off shift," "idle at base," "going to call," "at call scene," "going to hospital," "at hospital" or "returning to base." If the ambulance is stationary at location o, then we have $o_a = d_a = o$. If the ambulance is in motion, then t_a corresponds to the starting time of this movement. Otherwise, t_a corresponds to the starting time of the current phase in the service cycle. For example, if the status of the ambulance is "at hospital," then t_a corresponds to the time at which the ambulance arrived at the hospital. This time is kept in the state of an ambulance to be able to give a Markov formulation for the non-Markovian elements in the system, such as nonexponentially distributed service times and deterministic travel times. Similarly, for a call, we have $\mathbf{c} = (\sigma_c, o_c, t_c, p_c)$, where σ_c is the status of the call, o_c is the location of the call, t_c is the time at which the call arrived into the system and p_c is the priority level of the call. The status of a call σ_c can be "assigned to an ambulance" or "queued for service."

We model the dynamics of the system as an event-driven process. Events are triggered by changes in the status of the ambulances or by call arrivals. Therefore, the possible event types in the system are "call arrives," "ambulance goes off shift," "ambulance comes on shift," "ambulance departs for call scene," "ambulance arrives at call scene," "ambulance leaves call scene," "ambulance arrives at hospital," "ambulance leaves hospital" and "ambulance arrives at base." We assume that we can make decisions (for any ambulance, and not just the one involved in the event) only at the times of these events. This comes at the cost of some loss of optimality, since making decisions between the times of the events may improve performance. From a practical perspective, however, events are frequent enough to allow ample decision-making opportunities.

By restricting our attention to the times of events, we visualize the system as jumping from one event time to another. Therefore, we can use the tuple $s = (\tau, e, \mathbf{A}, \mathbf{C})$ to represent the state of the system, where τ corresponds to the current time, e corresponds to the current event type, and $\mathbf{A}$ and $\mathbf{C}$ respectively correspond to the state of the ambulances and the calls. In this case, the state trajectory of the system can be written as $\{s_k : k = 1, 2, \ldots\}$, where s_k is the state of the system just after the kth event occurs. Note that the time is rolled into our state variable. Throughout the paper, we use $\tau(s)$ and $e(s)$ to respectively denote the time and the event type when the state of the system is s. In other words, $\tau(s)$ and $e(s)$ are the first two components of the tuple $s = (\tau, e, \mathbf{A}, \mathbf{C})$.

2.2. Controls

We assume that calls are served in decreasing order of priority, and within a given priority level are served in first-in-first-out order. We further assume that the closest available ambulance is dispatched to a call. This is not an exact representation of reality, but is close enough for our purposes: We will show that ADP yields an effective redeployment strategy under this dispatch policy, and we expect that it will continue to do so for more realistic dispatch policies.

Let $\mathcal{R}(s)$ denote the set of ambulances that are available for redeployment when the state of the system is s. Let $u_{ab}(s) = 1$ if we redeploy ambulance a to base b when the state of the system is s, and 0 otherwise. Letting $\mathcal{B}$ be the set of ambulance bases, we can capture the potential reallocation decisions in the binary matrices $u(s) = \{u_{ab}(s) : a \in \mathcal{R}(s),\ b \in \mathcal{B}\}$. The set of feasible decisions is then

$$\mathcal{U}(s) = \Big\{ u(s) \in \{0,1\}^{|\mathcal{R}(s)| \times |\mathcal{B}|} : \sum_{b \in \mathcal{B}} u_{ab}(s) \leq 1 \quad \forall\, a \in \mathcal{R}(s) \Big\}.$$

The constraints in this definition simply state that each ambulance that is considered for redeployment can be redeployed to at most one base. An important assumption is that the cardinality of $\mathcal{U}(s)$ is small so that an optimization problem over this feasible set can be solved by enumerating over all feasible solutions.

We can use different definitions of $\mathcal{R}(s)$ to permit different amounts of relocation. For example, all available ambulances are considered for relocation if $\mathcal{R}(s)$ denotes the set of ambulances that are either idle at base or returning to base. But this may lead to impractically many relocations, so we can restrict $\mathcal{R}(s)$, for example to $\varnothing$ unless the event $e(s)$ corresponds to an ambulance becoming free after serving a call, in which case we can take $\mathcal{R}(s)$ equal to the singleton set corresponding to the newly freed ambulance. Here the ambulance crews are "on the road" when considered for relocation so there is no additional disruption to the crews relative to the standard "return to base" policy.

2.3. Fundamental Dynamics

Call arrivals are generated across the region $R \subset \mathbb{R}^2$ according to a Poisson point process with a known arrival intensity $C = \{C(t,x,y) : t \geq 0,\ (x,y) \in R\}$. As mentioned above, we have a fixed policy for serving calls, but our general approach does not depend on the particular form of this policy. If there are no available ambulances to serve a call, then the call is placed into a waiting queue. An ambulance serving a call proceeds through a sequence of events, including arriving at the scene, treating the patient, transporting and handing over the patient at the hospital. The main source of uncertainty in this call service cycle are the times spent between events. We assume that probability distributions for all of the event durations are known.

Besides these sources of randomness, the major ingredient governing the dynamics is the decision-making of dispatchers. As a result, the complete trajectory of the system is given by $\{(s_k, u_k) : k = 1, 2, \ldots\}$, where s_k is the state of the system at the time of the kth event and u_k is the decision (if any) made by the dispatcher when the state of the system is s_k. We capture the dynamics of the system symbolically by

$$s_{k+1} = f(s_k, u_k, X_k(s_k, u_k)),$$

where $X_k(s_k, u_k)$ is a random element of an appropriate space encapsulating all the sources of randomness described above. We do not attempt to give a more explicit description of the transition function $f(\cdot,\cdot,\cdot)$, since this does not add anything of value to our treatment. It suffices to point out that the preceding discussion and the fact that we can simulate the evolution of the system over time imply that an appropriate $f(\cdot,\cdot,\cdot)$ exists.

2.4. Transition Costs

Along with a transition from state s_k to s_{k+1} through decision u_k, we incur a cost $g(s_k, u_k, s_{k+1})$. In our particular implementation, we use the cost function

$$g(s_k, u_k, s_{k+1}) = \begin{cases} 1 & \text{if the event } e(s_{k+1}) \text{ corresponds to an ambulance arrival at a call scene,} \\ & \text{the call in question is urgent and the response time exceeds } \Delta \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Here Δ is a fixed threshold response time that is on the order of 10 minutes. Therefore, the cost function (1) allows us to count the number of high priority calls whose response times exceed a threshold response time. We are interested in the performance of the system over the finite planning horizon $[0, T]$, so let $g(s, \cdot, \cdot) = 0$ whenever $\tau(s) > T$. In our implementation, T corresponds to two weeks. We have chosen this particular cost function because of its simplicity, and also because most performance benchmarks in the EMS industry are formulated in terms of the percentage of calls that are reached within a time standard. Our approach allows other cost functions without affecting the general treatment.

2.5. Objective Function and Optimality Equation

A policy is a mapping from the state space to the action space, prescribing which action to take for each possible state of the system. Throughout the paper, we use $\mu(s)$ to denote the action prescribed by policy μ when the state of the system is s. If we follow policy μ, then the state trajectory of the system $\{s_k^\mu : k = 1, 2, \ldots\}$ evolves according to $s_{k+1}^\mu = f(s_k^\mu, \mu(s_k^\mu), X_k(s_k^\mu, \mu(s_k^\mu)))$ and the discounted total expected cost incurred by starting from initial state s is given by

$$J^\mu(s) = \mathbb{E}\Big[\sum_{k=1}^{\infty} \alpha^{\tau(s_k^\mu)}\, g(s_k^\mu, \mu(s_k^\mu), s_{k+1}^\mu) \,|\, s_1^\mu = s\Big],$$

where $\alpha \in [0, 1)$ is a fixed discount factor. The expectation in the expression above involves the random variables $\{s_k^\mu : k = 1, 2, \ldots\}$ and $\tau(s_k^\mu)$ is the time at which the system visits the kth state. It is well-known that the policy μ^* that minimizes the discounted total expected cost can be found by computing the value function through the optimality equation

$$J(s) = \min_{u \in \mathcal{U}(s)} \left\{ \mathbb{E}\Big[g(s, u, f(s, u, X(s, u))) + \alpha^{\tau(f(s,u,X(s,u))) - \tau(s)}\, J(f(s, u, X(s, u)))\Big] \right\} \tag{2}$$

and letting $\mu^*(s)$ be the minimizer of the right-hand side above; see Bertsekas and Shreve (1978).

The difficulty with the optimality equation (2) is that the state variable takes uncountably many values. Even if we are willing to discretize the state variable to obtain a countable state space, the state variable is still a high-dimensional vector and solving the optimality equation (2) through classical dynamic-programming approaches is computationally very difficult. In the next two sections, we propose a method to construct tractable approximations to the value function.

The discounting in (2) may seem odd, as it implies that we are minimizing the expected *discounted* number of calls that are not reached within the threshold time. This is purely a computational device, in the sense that the discount factor is very helpful in stabilizing the ADP approach that we describe in the next two sections. The key issue is that the effect of relocation decisions can be small relative to the simulation noise, and discounting appears to mitigate this to some extent. This observation is in agreement with empirical observations from the case studies in Chapter 8 of Bertsekas and Tsitsiklis (1996). The increase in stability is also supported by the theoretical results in Chapter 6.2 of Bertsekas and Tsitsiklis (1996). When presenting our computational results in Sections 5 and 6, we report the *undiscounted* number of calls that are not reached within the threshold response time. These results indicate that although we construct the value function approximations with a view towards minimizing the expected *discounted* cost, the same value function approximations provide very good performance in minimizing the expected *undiscounted* cost.

3. Approximate Dynamic Programming

The ADP approach that we use to construct approximations to the value function is closely related to the traditional policy iteration algorithm in the Markov decision processes literature. We begin with a brief description of the policy iteration algorithm. Throughout the rest of the paper, the greedy policy induced by an arbitrary function $\hat{J}(\cdot)$ refers to the policy that takes a decision in the set

$$\operatorname*{argmin}_{u \in \mathcal{U}(s)} \left\{ \mathbb{E}\Big[g(s,u,f(s,u,X(s,u))) + \alpha^{\tau(f(s,u,X(s,u)))-\tau(s)}\, \hat{J}(f(s,u,X(s,u))) \Big] \right\} \tag{3}$$

whenever the state of the system is s.

Policy Iteration

Step 1. Initialize the iteration counter n to 1 and initialize $J^1(\cdot)$ arbitrarily.

Step 2. (Policy improvement) Let μ^n be the greedy policy induced by $J^n(\cdot)$.

Step 3. (Policy evaluation) Let $J^{n+1}(\cdot) = J^{\mu^n}(\cdot)$, where $J^{\mu^n}(s)$ denotes the expected discounted cost incurred when starting from state s and using policy μ^n.

Step 4. Increase n by 1 and go to Step 2.

In our setting the cost function $g(\cdot,\cdot,\cdot)$ is nonnegative and the set of feasible decisions $\mathcal{U}(s)$ is finite, so Proposition 9.17 in Bertsekas and Shreve (1978) applies and hence $J^n(\cdot)$ converges pointwise to the solution to the optimality equation (2).

The difficulty with the policy iteration algorithm is that when dealing with uncountable state spaces, it is not even feasible to store $J^{\mu^n}(\cdot)$, let alone compute the expected discounted cost incurred by using policy μ^n. We overcome this difficulty by using value function approximations of the form

$$J(s,r) = \sum_{p=1}^{P} r_p\, \phi_p(s).$$

In the expression above, $r = \{r_p : p = 1,\ldots,P\}$ are tuneable parameters and $\{\phi_p(\cdot) : p = 1,\ldots,P\}$ are fixed basis functions. The challenge is to construct the basis functions and tune the parameters so that $J(\cdot, r)$ is a good approximation to the solution to the optimality equation (2). To achieve this, each basis function $\phi_p(\cdot)$ should capture some essential information about the solution to the optimality equation. In Section 4, we describe one set of basis functions that work well. Once a good set of basis functions is available, we can use the following approximate version of the policy iteration algorithm to tune the parameters $\{r_p : p = 1, \ldots, P\}$.

Approximate Policy Iteration

Step 1. Initialize the iteration counter n to 1 and initialize $r^1 = \{r_p^1 : p = 1, \ldots, P\}$ arbitrarily.

Step 2. (Policy improvement) Let μ^n be the greedy policy induced by $J(\cdot, r^n)$.

Step 3. (Policy evaluation through simulation) Simulate the trajectory of policy μ^n over the planning horizon $[0, T]$ for Q sample paths. Let $\{s_k^n(q) : k = 1, \ldots, K(q)\}$ be the state trajectory of policy μ^n in sample path q and $G_k^n(q)$ be the discounted cost incurred by starting from state $s_k^n(q)$ and following policy μ^n in sample path q.

Step 4. (Projection) Let

$$r^{n+1} = \operatorname*{argmin}_{r \in \mathbb{R}^P} \left\{ \sum_{q=1}^{Q} \sum_{k=1}^{K(q)} \big[G_k^n(q) - J(s_k^n(q), r) \big]^2 \right\}.$$

Step 5. Increase n by 1 and go to Step 2.

In Step 3 of approximate policy iteration we use simulation to evaluate the expected discounted cost incurred by policy μ^n. Therefore, $\{G_k^n(q) : k = 1, \ldots, K(q),\ q = 1, \ldots, Q\}$ are the sampled cost trajectories of the system under policy μ^n. In Step 4, we tune the parameters so that the value function approximation $J(\cdot, r)$ provides a good fit to the sampled cost trajectories.

There is still one computational difficulty in the approximate policy iteration algorithm. When simulating the trajectory of policy μ^n in Step 3, we need to solve an optimization problem of the form (3) to find the action taken by the greedy policy induced by $J(\cdot, r^n)$. This optimization problem involves an expectation that is difficult to compute. We resort to Monte Carlo simulation to overcome this difficulty. In particular, if the state of the system is s and we want to find the action taken by the greedy policy induced by $J(\cdot, r^n)$ in this state, then we enumerate over all decisions in the feasible set $\mathcal{U}(s)$. Starting from state s and taking decision u, we simulate the trajectory of the system until the next event and this provides a sample of $f(s, u, X(s, u))$. By using multiple samples, we estimate the expectation

$$\mathbb{E}\Big[g(s, u, f(s, u, X(s, u))) + \alpha^{\tau(f(s,u,X(s,u))) - \tau(s)} \, J(f(s, u, X(s, u)), r^n) \Big]$$

through a sample average. Once we estimate the expectation above for all $u \in \mathcal{U}(s)$, we choose the decision that yields the smallest value and use it as the decision taken by the greedy policy induced by $J(\cdot, r^n)$ when the state of the system is s. This approach is naturally subject to sampling error, but it provides good performance in practice.

Proposition 6.2 in Bertsekas and Tsitsiklis (1996) provides a performance guarantee for the approximate policy iteration algorithm. (The result is easily extended from finite to infinite state spaces.) This is an encouraging result as it provides theoretical support for the approximate policy iteration algorithm, but its conditions are difficult to verify in practice. In particular, the result assumes that we precisely know the error induced by using regression to estimate the discounted total expected cost of a policy, and it assumes that expectations are computed exactly rather than via sampling as in our case. For this reason, we do not go into the details of this result and refer the reader to Bertsekas and Tsitsiklis (1996) for further details.

4. Basis Functions

In this section, we describe the basis functions $\{\phi_p(\cdot) : p = 1, \ldots, P\}$ that we use in our value function approximations. We use $P = 6$ basis functions, some of which are based on the queueing insights developed in Restrepo, Henderson, and Topaloglu (2007).

4.1. Baseline

The first basis function is of the form $\phi_1(s) = 1 - \alpha^{T-\tau(s)}$. The role of this basis function is to give a very rough estimate of the discounted number of missed calls between the time associated with state s and the end of the planning horizon. In particular, if we assume that we miss a call every θ time units, then the discounted number of missed calls over the interval $[\tau(s), T]$ is

$$1 + \alpha^{\theta} + \alpha^{2\theta} + \ldots + \alpha^{\lfloor \frac{T-\tau(s)}{\theta} \rfloor \theta} = \frac{1 - \alpha^{\left[\lfloor \frac{T-\tau(s)}{\theta} \rfloor + 1\right]\theta}}{1-\alpha} \approx \frac{1 - \alpha^{T-\tau(s)}}{1-\alpha}.$$

4.2. Unreachable Calls

The second basis function computes the number of waiting calls for which an ambulance assignment has been made, but the ambulance will not reach the call within the threshold response time. This is easily computed in our setting where travel times are deterministic. In the case of stochastic travel times, we could use the probability that the ambulance will reach the call within the threshold response time instead. We do not give a precise expression for this basis function, since the precise expression is cumbersome yet straightforward to define.

4.3. Uncovered Call Rate

The third basis function captures the rate of call arrivals that cannot be reached on time by any available ambulance. To define this precisely we need some additional notation. Recall that calls arrive across the region $R \subset \mathbb{R}^2$ according to a Poisson point process with arrival intensity $C = \{C(t, x, y) : t \geq 0,\ (x, y) \in R\}$. Partition the region R into L subregions $\{\rho_l : l = 1, \ldots, L\}$, and associate a representative point or "center of mass" (x_l, y_l) with each subregion $l = 1, \ldots, L$.

The coverage of subregion l is the number of available ambulances that can reach the center of mass (x_l, y_l) within the threshold time standard. Let $d(t,(x_1,y_1),(x_2,y_2))$ be the travel time between points (x_1,y_1) and (x_2,y_2) when travel starts at time t. Then, letting $\mathbf{1}(\cdot)$ be the indicator function, the coverage of subregion l can be written as a function of the state of the system as

$$N_l(s) = \sum_{a \in \mathcal{A}(s)} \mathbf{1}(d(\tau(s),(x_a(s),y_a(s)),(x_l,y_l)) \leq \Delta).$$

Here, $\mathcal{A}(s)$ is the set of available ambulances and $(x_a(s), y_a(s))$ is the location of ambulance a at time $\tau(s)$ when the system is in state s. The rate of call arrivals at time t in subregion l is

$$C_l(t) = \int_{\rho_l} C(t,x,y)\, dx\, dy.$$

Therefore, when the state of the system is s, we can compute the rate of call arrivals that are not covered by any available ambulance by

$$\phi_3(s) = \sum_{l=1}^{L} C_l(\tau(s))\, \mathbf{1}(N_l(s) = 0).$$

4.4. Missed Call Rate

The previous 2 basis functions capture calls already received that we know we cannot reach on time, and the rate of arriving calls that cannot be reached on time because they are too far from any available ambulance. We could also fail to reach a call on time due to queueing effects from ambulances being busy with other calls. The fourth basis function represents an attempt to capture this effect. This basis function is of the form

$$\phi_4(s) = \sum_{l=1}^{L} C_l(\tau(s))\, P_l(s),$$

where $P_l(s)$ is the probability that all ambulances that could reach a call in subregion l on time are busy with other calls. We estimate $\{P_l(s) : l = 1, \ldots, L\}$ by treating the call service processes in different subregions as Erlang-loss systems. In Erlang-loss systems, calls arriving when all servers are busy are lost. In reality such calls are queued and served as ambulances become free, but the time threshold is almost always missed for such calls, so counting them as lost seems reasonable. The issue that such calls impose some load on the true system but are discarded in an Erlang-loss system creates a slight mismatch, but our computational results show that this function is still highly effective as a basis function.

The Erlang-loss probability $\mathcal{L}(\lambda, \mu, c)$ for a system with arrival rate λ, service rate μ and c servers is

$$\mathcal{L}(\lambda,\mu,c) = \frac{(\lambda/\mu)^c/c!}{\sum_{i=0}^{c} (\lambda/\mu)^i/i!}.$$

To characterize the Erlang-loss system for subregion l, given that the state of the system is s, we need to specify the number of servers, and the arrival and service rates. Let $\mathcal{N}_l(s)$ be the set of available ambulances that can serve a call in subregion l within the threshold response time. Then

$$\mathcal{N}_l(s) = \{a \in \mathcal{A}(s) : d(\tau(s),(x_a(s),y_a(s)),(x_l,y_l)) \leq \Delta\}.$$

We use $c = |\mathcal{N}_l(s)|$ as the number of servers in the Erlang loss system for subregion l. Let $\mu_l(t)$ be the service rate in the loss system, i.e., the rate at which an ambulance can serve a call at time t in subregion l. It is difficult to come up with a precise value for $\mu_l(t)$. It primarily depends on the time spent at the scene of a call and any transfer time at a hospital, since travel times are usually small relative to these quantities. In our practical implementation, we use historical data to estimate the time spent at the call scenes and the hospitals and add a small padding factor to capture travel times. Finally, let $\Lambda_l(s)$ be the rate of call arrivals that should be served by ambulances in the set $\mathcal{N}_l(s)$. Coming up with a value for $\Lambda_l(s)$ is even more difficult than devising a value for $\mu_l(t)$. One option is to let $\Lambda_l(s) = C_l(\tau(s))$, which is the rate of call arrivals at time $\tau(s)$ in subregion l. However, ambulances in the set $\mathcal{N}_l(s)$ serve not only calls in subregion l, but also calls in other subregions. To attempt to capture this let

$$\Lambda_l(s) = \sum_{a \in \mathcal{N}_l(s)} \sum_{i=1}^{L} C_i(\tau(s))\, \mathbf{1}(d(\tau(s), (x_a(s), y_a(s)), (x_i, y_i)) \le \Delta), \tag{4}$$

so that $\Lambda_l(s)$ reflects the total call arrival rate in subregions that are close to any of the ambulances in the set $\mathcal{N}_l(s)$. We then use the approximation

$$P_l(s) \approx \mathcal{L}(\Lambda_l(s), \mu_l(\tau(s)), |\mathcal{N}_l(s)|). \tag{5}$$

There are at least 2 shortcomings in the estimate for $\Lambda_l(s)$ in (4) and the approximation to $P_l(s)$ in (5). First, there is double counting in the estimate of $\Lambda_l(s)$. In particular, if 2 ambulances $a_1, a_2 \in \mathcal{N}_l(s)$ can both reach subregion ℓ within the time threshold, then the summation for $\Lambda_l(s)$ counts $C_\ell(\tau(s))$ twice. Second, $C_\ell(\tau(s))$ could be counted in the demand rates for multiple subregions. To be more precise, if there are three subregions l_1, l_2, ℓ and two ambulances $a_1 \in \mathcal{N}_{l_1}(s)$, $a_2 \in \mathcal{N}_{l_2}(s)$ such that both a_1 and a_2 can reach subregion ℓ within the time threshold, then the summations for $\Lambda_{l_1}(s)$ and $\Lambda_{l_2}(s)$ both count $C_\ell(\tau(s))$. Therefore, we typically have $\sum_{l=1}^{L} \Lambda_l(s) > \sum_{l=1}^{L} C_l(\tau(s))$. To overcome this problem, we dilute the call arrival rates by a factor κ. In particular, we use the call arrival rate $\kappa\, C_\ell(\tau(s))$ in (4) instead of $C_\ell(\tau(s))$ so that we may roughly have $\sum_{l=1}^{L} \Lambda_l(s) = \sum_{l=1}^{L} \kappa\, C_l(\tau(s))$. We find a good value for κ through preliminary experimentation. Interestingly, as we will see in our computational results, it is not necessarily the case that the best choice of κ lies in $(0, 1)$. We emphasize that κ is the only tuneable parameter in our ADP approach that requires experimentation.

4.5. Future Uncovered Call Rate

In certain settings, we may not be willing to redeploy ambulances that are already in transit to a particular location. In this case, from the perspective of covering future calls, the ambulance destinations are as important as their current locations. This is the motivation underlying the fifth and sixth basis functions. Our fifth basis function parallels the third one, replacing the true locations of ambulances by their destinations.

The definition of this basis function is therefore identical to that of ϕ_3, but the configuration of the ambulances that we use to compute $N_l(s)$ is not the current one, but rather, an estimated future configuration that is obtained by letting all ambulances in transit reach their destinations and all stationary ambulances remain at their current locations. To be precise, given that the system is in state $s = (\tau, e, \mathbf{A}, \mathbf{C})$ with $\mathbf{A} = (\mathbf{a}_1, \ldots, \mathbf{a}_N)$ and $\mathbf{a}_i = (\sigma_{a_i}, o_{a_i}, d_{a_i}, t_{a_i})$, we define a new state $\vec{s}(s) = (\tau + 1/\sum_{l=1}^{L} C_l(\tau), e, \vec{\mathbf{A}}, \mathbf{C})$ with $\vec{\mathbf{A}} = (\vec{\mathbf{a}}_1, \ldots, \vec{\mathbf{a}}_N)$ and $\vec{a_i} = (\vec{\sigma}_{a_i}, d_{a_i}, d_{a_i}, \tau + 1/\sum_{l=1}^{L} C_l(\tau))$, where $\vec{\sigma}_{a_i}$ is the status of ambulance a_i when it reaches its destination d_{a_i}. In this case, we can write the fifth basis function as

$$\phi_5(s) = \sum_{l=1}^{L} C_l(\tau(\vec{s}(s)))\, \mathbf{1}(N_l(\vec{s}(s)) = 0).$$

The time $\tau(\vec{s}(s)) = \tau + 1/\sum_{l=1}^{L} C_l(\tau)$ is used as an approximation to the expected time of the next call arrival. The next call may arrive before or after the ambulances actually reach their destinations, but we heuristically use the time $\tau(\vec{s}(s))$ simply to look into the future. The idea is that the estimated future configuration of the ambulances $\vec{\mathbf{A}}$ is more likely to hold at the future time $\tau(\vec{s}(s))$ than at the current time $\tau(s)$.

4.6. Future Missed Call Rate

The sixth basis function, ϕ_6, parallels ϕ_4 in that it captures the rate of calls that will likely be lost due to queueing congestion. As with the fifth basis function, it uses an estimated future configuration of the ambulances. It is defined as

$$\phi_6(s) = \sum_{l=1}^{L} C_l(\tau(\vec{s}(s)))\, P_l(\vec{s}(s)).$$

5. Computational Results on Edmonton

In this section, we present computational results for the EMS system in the city of Edmonton, Alberta in Canada. We begin with a description of the data set along with our assumptions.

5.1. Experimental Setup

Our data are based on the data set used in the computational experiments in Ingolfsson, Erkut, and Budge (2003). The city of Edmonton has a population of 730,000 and covers an area of around 40×30 km^2. The EMS system includes 16 ambulances, 11 bases and 5 hospitals. We assume for simplicity that all ambulances are of the same type and operate all day. An ambulance, upon arriving at a call scene, treats the patient for an exponentially distributed amount of time with mean 12 minutes. After treating the patient at the call scene, the ambulance transports the patient to a hospital with probability 0.75. The probability distribution for the hospital chosen is inferred from historical data. The time an ambulance spends at the hospital has a Weibull distribution with mean 30 minutes and standard deviation 13 minutes. The turn-out time for the ambulance crews is 45 seconds. In other words, if the ambulance crew is located at a base when it is notified of a call, then it takes 45 seconds to get ready. An ambulance crew already on the road does not incur the turn-out time. The road network that we use in our computational experiments models the actual road network on the avenue level. There are

252 nodes and 934 arcs in this road network. The travel times are deterministic and do not depend on the time of the day.

There was not enough historical data to develop a detailed model of the call arrivals so we proceeded with a revealing, but not necessarily realistic, model. The model maintains a constant overall arrival rate, but the distribution of the location of calls changes in time. We divided the city of Edmonton into 20×17 subregions and assumed that the rate of call arrivals in subregion l at time t is given by

$$C_l(t) = C\big[\gamma_l + \delta_l \sin(2\pi t/24)\big],$$

where t is measured in hours. In the expression above, C, γ_l and δ_l are fixed parameters that satisfy $C \geq 0$, $\gamma_l \geq |\delta_l|$, $\sum_{l=1}^{340} \gamma_l = 1$ and $\sum_{l=1}^{340} \delta_l = 0$. We have $\sum_{l=1}^{340} \gamma_l + \sum_{l=1}^{340} \delta_l \sin(2\pi t/24) = 1$ so that we can interpret C as the total call arrival rate into the system and $\gamma_l + \delta_l \sin(2\pi t/24)$ as the probability that a call arriving at time t falls in subregion l. If $\delta_l > 0$, then the peak call arrival rate in subregion l occurs at hours $\{6, 30, 54, \ldots\}$, whereas if $\delta_l < 0$, then the peak call arrival rate in subregion l occurs at hours $\{18, 42, 66, \ldots\}$. The average call arrival rate over a day in subregion l is $C\gamma_l$. We estimated C and γ_l using historical data and C ranged from 3 to 7 calls per hour. We chose appropriate values of δ_l so that we have higher call arrival rates in the business subregions early in the day and higher call arrival rates in the residential subregions later in the day.

We implemented ADP on top of the BartSim simulation engine developed by Henderson and Mason (2004). In all of our computational experiments, we use a discount factor of $\alpha = 0.85$ per day. The simulation horizon is 14 days. We use 30 sample paths in Step 3 of the approximate policy iteration algorithm. After some experimentation, we found that $\kappa = 0.1$ in the fourth and sixth basis functions gave the best results.

We use the static redeployment policy as a benchmark. In particular, the static redeployment policy preassigns a base to each ambulance and redeploys each ambulance back to its preassigned base whenever it becomes free after serving a call. We found a good static redeployment policy by simulating the performance of the system under a large number of possible base assignments and choosing the base assignment that gave the best performance.

5.2. Baseline Performance

In our first set of computational experiments, we test the performance of ADP under the assumption that the only time we can redeploy an ambulance is when it becomes free after serving a call. We do not redeploy ambulances at other times. Following the discussion of Section 2.2, this is equivalent to setting $\mathcal{R}(s) = \varnothing$, except when $e(s)$ corresponds to an ambulance becoming free after serving a call, in which case $\mathcal{R}(s)$ is the singleton set corresponding to the ambulance just becoming free. The motivation for using this redeployment strategy is that it minimizes the disturbance to the crews and never redeploys an ambulance that is located at an ambulance base.

Figure 1 shows the performance of ADP. The horizontal axis gives the iteration number in the approximate policy iteration algorithm and the vertical axis gives the percentage of calls not reached

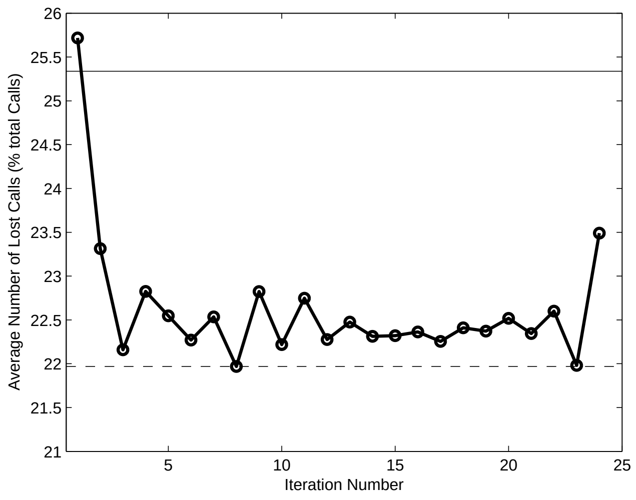

Figure 1: Performance of ADP and the benchmark policy.

within the threshold response time. The plot gives the percentage of calls missed by the greedy policy induced by the value function approximation at a particular iteration. The dashed horizontal line shows the best performance attained by ADP, while the thin horizontal line shows the performance of the best static policy. ADP attains a good policy within two iterations and then bounces around this policy. The best ADP policy misses 21.9% (± 0.2%) of the calls, whereas the best static redeployment policy misses 25.3% (±0.2%) of the calls. (These figures are undiscounted numbers of missed calls.)

The ADP approach does not converge on a single policy. This lack of convergence is not a concern from a practical viewpoint since we can simply choose the best policy that is obtained during the course of the approximate policy iteration algorithm and implement this policy in practice.

To emphasize the importance of choosing the basis functions carefully, Figure 2 shows the performance of ADP when we use an arrival rate $\lambda = C_l(\tau(s))$ instead of the quantity in (4) in the fourth and sixth basis functions. The best policy obtained through ADP misses 23.6% (± 0.2%) of the calls. This is in contrast with the best policy missing 21.9% (± 0.2%) of the calls in Figure 1. Furthermore, the fluctuation in the performance of the policies in Figure 2 is much more drastic when compared with Figure 1. The results indicate that choosing the basis functions carefully makes a significant difference in the performance of ADP.

5.3. Comparison with Random Search

For a fixed set of basis functions $\{\phi_p(\cdot) : p = 1, \ldots, P\}$, a set of values for the tuneable parameters $r = \{r_p : p = 1, \ldots, P\}$ characterize a value function approximation $J(\cdot, r)$ and this value function approximation induces a greedy policy. Therefore, a brute-force approach for finding a good set of values for the tuneable parameters is to carry out a random search over an appropriate subset of $\mathbb{R}^P$ and use simulation to test the performance of the greedy policies induced by the different sets of values

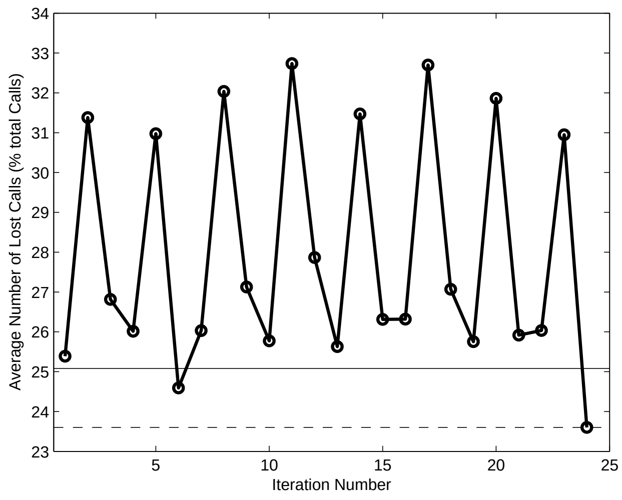

Figure 2: Performance of ADP with poorly chosen basis functions.

for the tuneable parameters.

To implement this idea, we first use ADP to obtain a good set of values for the tuneable parameters. Letting $\{\hat{r}_p : p = 1, \ldots, P\}$ be this set of values, we sample $r = \{r_p : p = 1, \ldots, P\}$ uniformly over the box $\left[\hat{r}_1 - \frac{1}{2}\,\hat{r}_1, \hat{r}_1 + \frac{1}{2}\,\hat{r}_1\right] \times \ldots \times \left[\hat{r}_P - \frac{1}{2}\,\hat{r}_P, \hat{r}_P + \frac{1}{2}\,\hat{r}_P\right]$ and use simulation to test the performance of the greedy policy induced by the value function approximation $J(\cdot, r)$. We sampled 1,100 sets of values for the tuneable parameters and this provides 1,100 value function approximations. Figure 3 gives a histogram for the percentage of the calls missed by the greedy policies induced by these 1,100 value function approximations. The vertical lines correspond to the percentage of calls missed by the best policy obtained by ADP and the best static redeployment policy. The figure indicates that very few (3%) of the sampled sets of values for the tuneable parameters provide better performance than the best policy obtained by ADP. On the other hand, approximately 28% of the samples provide better performance than the best static redeployment policy.

The random search procedure we use is admittedly rudimentary and one can use more sophisticated techniques to focus on the more promising areas of the search space. Nevertheless, our results indicate that when one looks at the broad landscape of the possible values for the tuneable parameters, ADP is quite effective in identifying good parameters. Moreover, the computation time required by the random search procedure is on the order of several days, whereas the computation time required by ADP is a few hours.

5.4. Making Additional Redeployments

The computational experiments in Section 5.2 and 5.3 allow redeployments only when an ambulance becomes free after serving a call. We now explore the possibility of improving performance by allowing additional ambulance redeployments. Define an additional event type "consider redeployment" and

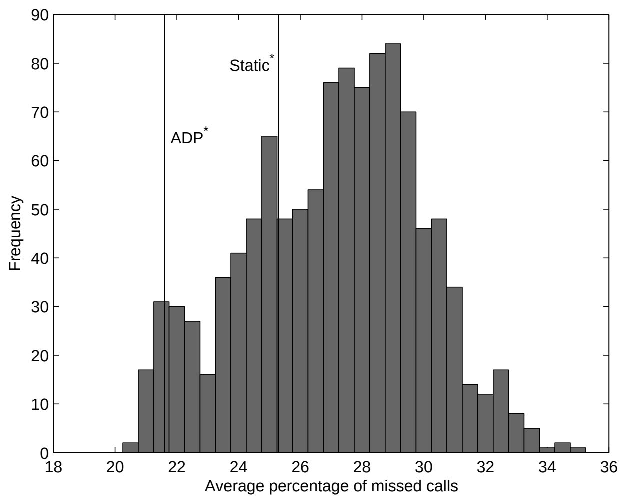

Figure 3: Performance of the 1,100 greedy policies obtained through random search.

schedule an event of this type with a certain frequency that is detailed below. Whenever an event of this type is triggered, we consider redeploying the ambulances that are either at a base or returning to a base, so that $\mathcal{R}(s)$ can contain multiple ambulances at such times. The set $\mathcal{R}(s)$ continues to be a singleton when $e(s)$ corresponds to an ambulance becoming free after serving a call, and at all other events, $\mathcal{R}(s) = \varnothing$.

We use two methods to vary the redeployment frequency. In the first method, we equally space consider-redeployment events to obtain frequencies between 0 and 15 per hour. In the second method, the frequency of consider-redeployment events is fixed at 24 per hour, but we make a redeployment only when the estimated benefit from making the redeployment exceeds the estimated benefit from not making the redeployment by a significant margin. More precisely, letting $\epsilon \in [0, 1)$ be a tolerance margin and using $\bar{0}(s)$ to denote the $|\mathcal{R}(s)| \times |\mathcal{B}|$ dimensional matrix of zeros corresponding to the decision matrix of not making a redeployment, if we have

$$\begin{aligned}\underset{u \in \mathcal{U}(s)}{\operatorname{argmin}} \Big\{ \mathbb{E}\Big[g(s, u, f(s, u, X(s, u))) + \alpha^{\tau(f(s,u,X(s,u))) - \tau(s)}\, J(f(s, u, X(s, u)), r) \Big] \Big\} \\ \leq (1 - \epsilon)\, \mathbb{E}\Big[g(s, \bar{0}, f(s, u, X(s, \bar{0}))) + \alpha^{\tau(f(s,\bar{0},X(s,\bar{0}))) - \tau(s)}\, J(f(s, \bar{0}, X(s, \bar{0})), r) \Big],\end{aligned}$$

then we make the redeployment decision indicated by the optimal solution to the problem on the left-hand side. Otherwise, we do not make a redeployment. Larger values of ϵ decrease the frequency of redeployments. We vary ϵ between 0.1 and 0.005.

Figure 4 shows the performance improvement obtained by the additional redeployments. The horizontal axis gives the frequency of the redeployments measured as the number of redeployments per ambulance per day. The vertical axis gives the percentage of missed calls. The dashed (solid) data line corresponds to the first (second) method of varying the redeployment frequency. From Figure 1 we

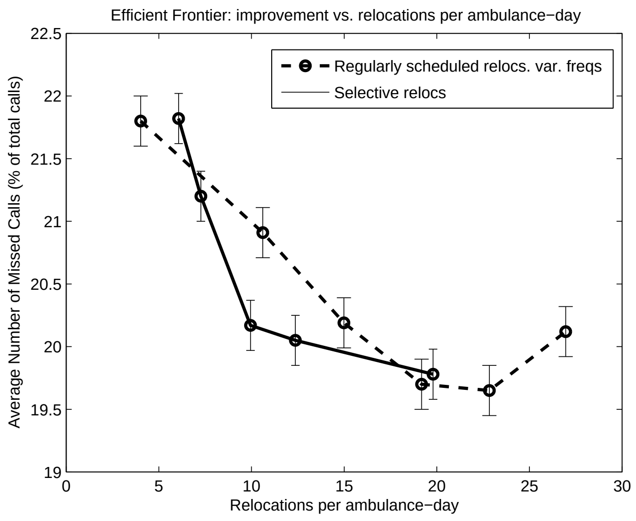

Figure 4: Performance of ADP as a function of the frequency of the additional redeployments.

miss 21.9% of the calls without making any additional redeployments. By making about 10 additional redeployments per ambulance per day, we can decrease the percentage of missed calls to 20.2%. Beyond this range, we reach a plateau and additional redeployments do not provide much improvement. Another important observation is that the second method tends to provide significantly better performance improvements with the same frequency of additional redeployments. For example, the second method reduces the percentage of missed calls to 20.2% with 10 additional redeployments per ambulance per day, whereas the first method needs 15 additional redeployments per day to reach the same level. Therefore, it appears that making redeployments only when the value function approximation signals a significant benefit is helpful in avoiding pointless redeployments.

6. Computational Results on a Second Metropolitan Area

In this section, we present the performance of ADP on the EMS system operating in a second metropolitan area. This EMS system is also studied in Richards (2007). We cannot disclose the name of the metropolitan area due to confidentiality agreements.

6.1. Experimental Setup

The population of the city in question is more than 5 times that of Edmonton and its size is around 180×100 km^2. The EMS system includes up to 97 ambulances operating during peak times, 88 bases and 22 hospitals. The turn-out times, call-scene times and hospital-transfer times are comparable to those in Edmonton. We were able to use a detailed model for determining to which hospital a patient needs to be transported. In particular, the destination hospital depends on the location of the call. Calls originating at a given location are transported to any of a small set of hospitals – usually no more than 2 or 3 out of the 22 hospitals in the system. The corresponding probabilities are inferred from historical data. We assume that all ambulances are of the same type and a call that is not served within 8 minutes is interpreted as missed. The road network that we use in our computational experiments models the actual network on the avenue level and there are 4,955 nodes and 11,876 arcs in this road network.

The call arrival model that we used is quite realistic. The data that we used were collected from one year of operations of the EMS system and consisted of aggregated counts of calls for each hour of the week during a whole year, for each of 100×100 geographic zones. Due to the irregular and non-convex shape of the metropolitan area, roughly 80% of these zones had zero total call counts and did not intervene in the dynamics. From the remaining 20% a significant percentage had very low hourly counts of at most 5 calls. Thus, it was necessary to apply smoothing procedure for the lowest intensity zones so as to reduce the sampling noise. In order to do this, we first classified the zones in a few groups according to their average intensity along the week. Then, for the lowest intensity groups, we computed a total intensity for each hour and then distributed this total intensity uniformly among the zones in this group. In this way we obtained an intensity model that combined a uniform low intensity background with actual (accurate) counts on the highest intensity zones. The average call arrival rate was 570 calls per day and fluctuated on any day of the week from a low of around 300 calls per day to a high of around 750 calls per day.

We used the actual base assignments and ambulance shifts as a benchmark. In the EMS system, a maximum of 97 ambulances operate at noon. Out of these, 66 ambulances work all day in two 12 hour shifts and the remaining 31 have single shifts typically ranging from 10 to 12 hours per day. The resulting shift schedule provides 1,708 ambulance hours per day. Ambulances are redeployed to their preassigned bases whenever they become free after serving a call. We refer the reader to Richards (2007) for details on the ambulance shifts.

6.2. Baseline Performance

Figure 5 shows the performance of ADP. The interpretations of the axes in this figure are the same as those in Figures 1 and 2. The solid horizontal line plots the percentage of calls missed by the benchmark policy and the dashed horizontal line plots the best performance obtained using ADP. The solid data line plots the performance of ADP when we use $\kappa = 2$ in the fourth basis function, whereas the dashed line plots the performance when we use $\kappa = 1$. The value $\kappa = 2$ was found by experimentation to give the best policy and least fluctuations across iterations of all the values tried in the interval $(0, 2]$.

For both values of κ, the best ADP policies are attained in one iteration and these policies perform very similarly. The improvements in the number of reached calls are roughly 3.0% and 4.0% in the case of $\kappa = 1$ and $\kappa = 2$. We get significantly more stable performance with $\kappa = 2$ than with $\kappa = 1$. This indicates that it is important to carefully tune the parameter κ through some initial experimentation.

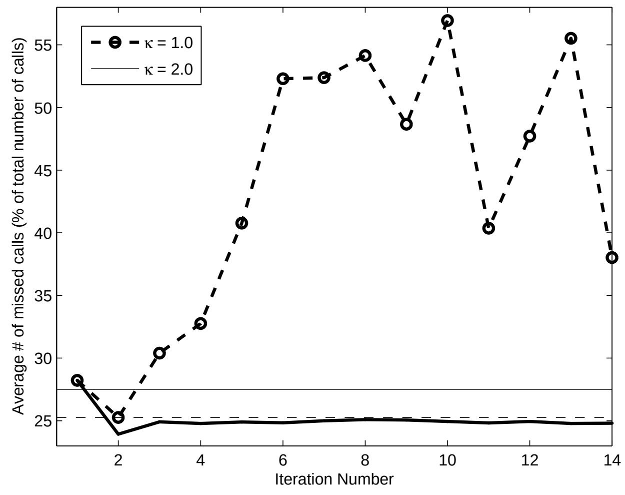

Figure 5: Performance of ADP and the benchmark policy.

633 calls per day		
	Turn out = 45 secs.	Turn out = 0 secs.
% of calls missed by benchmark	27.5	23.0
% of calls missed by ADP	23.9	19.3
Improvement	3.6	3.2
Rel. improvement	3.6 / 27.5 = 0.13	3.2 / 23.0 = 0.139

846 calls per day		
	Turn out = 45 secs.	Turn out = 0 secs.
% of calls missed by benchmark	35.5	30.7
% of calls missed by ADP	30.5	26.3
Improvement	4.9	4.5
Rel. improvement	4.9 / 35.5 = 0.139	4.5 / 30.7 = 0.145

Table 1: Effect of turn-out time on the performance gap between ADP and the benchmark policy.

6.3. Effect of Turn-Out Time

Recall that if the ambulance crew is stationed at a base when it is notified of a call, then it takes 45 seconds to get ready, i.e., the turn-out time is 45 seconds. On the other hand, an ambulance crew that is already on the road does not incur turn-out time. A potential argument against ambulance redeployment is that any gains are simply due to ambulance crews being on the road more often, and therefore incurring less turn-out time delays.

To check the validity of this argument, we used ADP under the assumption that turn-out time is zero. In other words, an ambulance crew does not take any time to get ready, even if it is located at a base when it is notified of a call. Table 1 shows the results for two different call arrival regimes (633 calls per day and 846 calls per day). The third and fourth rows show the absolute and relative improvement of ADP over the benchmark strategy. The results indicate that ADP provides significant improvement over the benchmark strategy in all cases. At first sight, this improvement seems slightly smaller for the cases without turn-out time, but the relative improvement is roughly the same in all cases.

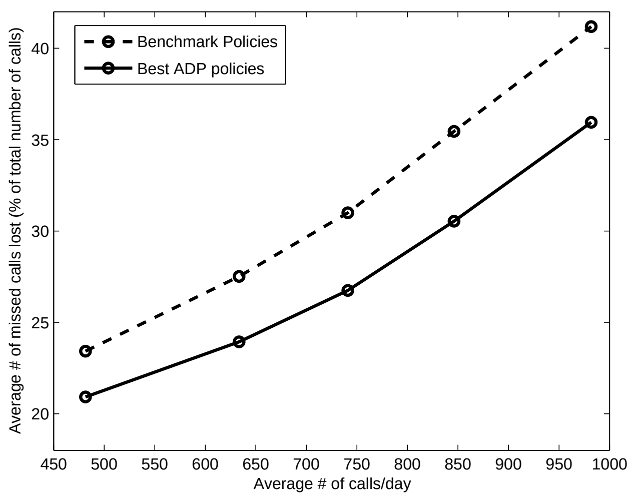

Figure 6: Performance of ADP and benchmark policy for different call arrival rates.

6.4. Varying Call Arrival Rates and Fleet Sizes

We now explore the effect of different call arrival rates and fleet sizes on the performance of ADP. We first carry out a number of runs under the experimental setup described in Section 6.1, but we vary the call arrival rates by uniformly scaling them by a constant factor. Recall that the average call arrival rate in the experimental setup in Section 6.1 is around 570 calls per day. Figure 6 shows the percentage of the calls that are missed by the best policy obtained by ADP and the benchmark strategy as a function of the average call arrival rate. The solid data line corresponds to ADP, whereas the dashed data line corresponds to the benchmark strategy. ADP provides substantial improvements over all call arrival regimes. The improvements provided by ADP are more significant when the call arrival rate is relatively high. It appears that when the call arrival rate is high, there is more room for improvement by redeploying ambulances carefully and ADP effectively takes advantage of this greater room for improvement.

In a second set of computational experiments, we hold the call arrival rate constant and vary the number of ambulances in the fleet. Figure 7 shows the performance of ADP and the benchmark strategy as a function of the fleet size. Since we do not have access to the base assignments used by the benchmark strategy for different fleet sizes, we modify the base assignments described in Section 6.1 heuristically. In particular, to produce a base assignment with fewer ambulances, we choose the ambulances assigned to bases serving low demand and take them off shift. Similarly, to produce a base assignment with extra ambulances, we add ambulances to the bases with the highest demand.

The results indicate that ADP performs consistently better than the benchmark policy. An important observation from Figure 7 is that if our goal is to keep the percentage of the missed calls below a given threshold, say 28%, then ADP allows us to reach this goal with roughly 5 or 6 fewer ambulances than the benchmark policy. This translates into significant cost savings in an EMS system. It is also

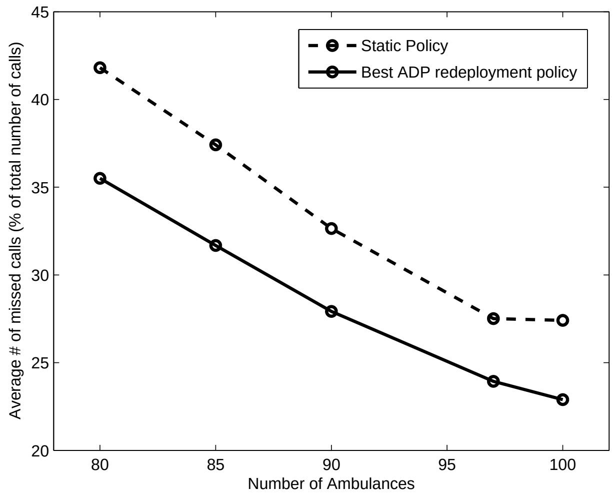

Figure 7: Performance of ADP and benchmark policy for different fleet sizes.

interesting that the performance of the benchmark policy does not improve significantly beyond 97 or 98 ambulances in the fleet, whereas ADP continues to provide progressively lower missed-call rates. This partly reflects the quality of our heuristic benchmark strategy, but it also indicates that a redeployment strategy can help mitigate poor static allocations and make effective use of extra capacity.

7. Conclusions

In this paper, we formulated the ambulance redeployment problem as a dynamic program and used an approximate version of the policy iteration algorithm to deal with the high-dimensional and uncountable state space. Extensive experiments on two problem scenarios showed that ADP can provide high-quality redeployment policies. The basis functions that we constructed open up the possibility of using other approaches, such as temporal-difference learning and the linear-programming approach, to tune the parameters $\{r_p : p = 1, \ldots, P\}$. Indeed, we are currently exploring the linear-programming approach.

Other future research will incorporate additional degrees of realism into our model. We plan to include stochastic travel times, multiple call priorities, other cost functions and more realistic ambulance dynamics that involve multiple ambulances serving certain calls. Incorporating these complexities may require constructing additional basis functions.

Acknowledgments

We thank Alex Currell, Armann Ingolfsson and Andrew Mason for their advice and support, and for providing the data sets used in our computational experiments. This work was partially supported by National Science Foundation Grants DMI 0400287 and DMI 0422133.

References

Adelman, D. 2004. A price-directed approach to stochastic inventory routing. *Operations Research* **52** 499–514.

Adelman, D. 2007. Dynamic bid-prices in revenue management. *Operations Research* **55** 647–661.

Adelman, Daniel, Adam J. Mersereau. 2007. Relaxations of weakly coupled stochastic dynamic programs. *Operations Research* .

Andersson, T. 2005. Decision support tools for dynamic fleet management. Ph.D. thesis, Department of Science and Technology, Linkoepings Universitet, Norrkoeping, Sweden.

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

Berman, O. 1981a. Dynamic repositioning of indistinguishable service units on transportation networks. *Transportation Science* **15**.

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

Berman, O. 1981c. Repositioning of two distinguishable service vehicles on networks. *IEEE Transactions on Systems, Man, and Cybernetics* **SMC-11**.

Bertsekas, D.P., S.E Shreve. 1978. *Stochastic Optimal Control: The Discrete Time Case.*. Academic Press, New York.

Bertsekas, D.P., J.N. Tsitsiklis. 1996. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, Massachusetts.

Brotcorne, L., G. Laporte, F. Semet. 2003. Ambulance location and relocation models. *European Journal of Operations Research* **147** 451–463.

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

Farias, V. F., B. Van Roy. 2006. Tetris: A study of randomized constraint sampling. G. Calafiore, F. Dabbene, eds., *Probabilistic and Randomized Methods for Design Under Uncertainty*. Springer-Verlag.

Farias, V. F., B. Van Roy. 2007. An approximate dynamic programming approach to network revenue management. Tech. rep., Stanford University, Department of Electrical Engineering.

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

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

Goldberg, J. B. 2007. Personal Communication.

Henderson, S. G., A. J. Mason. 2004. Ambulance service planning: Simulation and data visualisation. M. L. Brandeau, F. Sainfort, W. P. Pierskalla, eds., *Operations Research and Health Care: A Handbook of Methods and Applications*. Handbooks in Operations Research and Management Science, Kluwer Academic, 77–102.

Ingolfsson, A. 2006. The impact of ambulance system status management. Presentation at 2006 INFORMS Conference.

Ingolfsson, A., E. Erkut, S. Budge. 2003. Simulation of single start station for Edmonton EMS. *Journal of the Operational Research Society* **54** 736–746.

Kolesar, P., W. E. Walker. 1974. An algorithm for the dynamic relocation of fire companies. *Operations Research* **22** 249–274.

Nair, R., E. Miller-Hooks. 2006. A case study of ambulance location and relocation. Presentation at 2006 INFORMS Conference.

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

Restrepo, M., S.G Henderson, H. Topaloglu. 2007. Erlang loss models for the static deployment of ambulances. Submitted.

Richards, D. P. 2007. Optimised ambulance redeployment strategies. Master's thesis, Department of Engineering Science, University of Auckland, Auckland, New Zealand.

Schweitzer, P., A. Seidmann. 1985. Generalized polynomial approximations in Markovian decision processes. *Journal of Mathematical Analysis and Applications* **110** 568–582.

Si, Jennie, Andrew G. Barto, Warren B. Powell, Donald Wunsch II, eds. 2004. *Handbook of Learning and Approximate Dynamic Programming*. Wiley-Interscience, Piscataway, NJ.

Sutton, R. S. 1988. Learning to predict by the methods of temporal differences. *Machine Learning* **3** 9–44.

Topaloglu, H., W. B. Powell. 2006. Dynamic programming approximations for stochastic, time-staged integer multicommodity flow problems. *INFORMS Journal on Computing* **18** 31–42.

Tsitsiklis, J., B. Van Roy. 1997. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control* **42** 674–690.

Tsitsiklis, J. N. 1994. Asynchronous stochastic approximation and Q-learning. *Machine Learning* **16** 185–202.

Tsitsiklis, J.N, B. Van Roy. 2001. Regression methods for pricing complex American-style options. *IEEE Transactions on Neural Networks* **12** 694–703.

Van Roy, Benjamin, Dimitri P. Bertsekas, Yuchun Lee, John N. Tsitsiklis. 1997. A neuro dynamic programming approach to retailer inventory management. *Proceedings of the IEEE Conference on Decision and Control*.

Watkins, C. J. C. H., P. Dayan. 1992. Q-learning. *Machine Learning* **8** 279–292.

Yan, X., P. Diaconis, P. Rusmevichientong, B. Van Roy. 2005. Solitaire: Man versus machine. *Advances in Neural Information Processing Systems* **17**.