Proceedings of the 2009 Winter Simulation Conference
M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, eds.

AMBULANCE REDEPLOYMENT: AN APPROXIMATE DYNAMIC PROGRAMMING APPROACH

Matthew S. Maxwell
Shane G. Henderson
Huseyin Topaloglu

Department of Operations Research and Information Engineering
Cornell University
Ithaca, NY 14853, USA

ABSTRACT

Emergency medical service (EMS) providers are charged with the task of managing ambulances so that the time required to respond to emergency calls is minimized. One approach that may assist in reducing response times is ambulance redeployment, i.e., repositioning idle ambulances in real time. We formulate a simulation model of EMS operations to evaluate the performance of a given allocation policy and use this model in an approximate dynamic programming (ADP) context to compute high-quality redeployment policies. We find that the resulting ADP policies perform much better than sub-optimal static policies and marginally better than near-optimal static policies. Representative computational results for Edmonton, Alberta are included.

1 INTRODUCTION

Emergency medical service (EMS) providers are faced with the complex decision of how to position ambulances throughout a region to respond to emergency calls quickly. Furthermore, they are often under pressure from contractual obligations or managerial goals to maintain a certain level of performance. This problem is exacerbated by the current trends of increasing call rates, worsening traffic conditions, and increasing operating costs. Ambulance redeployment strategies are one approach that may alleviate some of this pressure. Ambulance redeployment, also known as relocation, system-status management, move up, or dynamic repositioning, refers to strategies under which idle ambulances are repositioned in real-time to compensate for other ambulances that are busy and hence unavailable to respond to incoming calls. Allocation policies that use ambulance redeployment will be referred to as dynamic policies or dynamic-allocation policies, whereas those policies that do not will be referred to as static policies or static-allocation policies.

In this paper, which builds upon Restrepo (2008) and Maxwell et al. (2009), we present a simulation model as a means to assess the performance of a given EMS allocation policy and formulate an algorithm to optimize an allocation policy within given resource constraints. Our optimization algorithm is an approximate dynamic programming (ADP) approach. Restrepo (2008) provides computational methods to efficiently compute both promising static-allocation policies and dynamic-allocation policies. Maxwell et al. (2009) focuses on the details and theory of the ADP approach, whereas this paper centers on the details of the simulation model used.

There are three classes of models for ambulance redeployment in the literature. The first class involves solving integer programs in real-time each time a redeployment decision is to be made; see Kolesar and Walker (1974), Gendreau, Laporte, and Semet (2001), Brotcorne, Laporte, and Semet (2003), and Nair and Miller-Hooks (2006). The objective functions of these integer programs involve a combination of coverage for future calls and relocation costs for moving ambulances. This computationally intensive approach may necessitate the implementation of a parallel computing environment to provide decision support that is fast enough for real-time application. The second class involves computing optimal ambulance positions for every number of available ambulances via a similar integer programming formulation in an offline preparatory phase. Dispatchers then attempt to redeploy ambulances in real time to match the configurations of these solutions; see Gendreau, Laporte, and Semet (2006), Ingolfsson (2006) and Goldberg (2007). Unlike the first two classes, the third class attempts to model the randomness in the system explicitly. One approach is to formulate the ambulance redeployment problem as a Markov decision process and then solve for an optimal policy using exact dynamic programming; see Berman (1981a), Berman (1981b), Berman (1981c), and Zhang, Mason, and Philpott (2008). As is often the case with dynamic programming methods, these models provide important insights, but only work for simplified models involving only a few ambulances. Another approach is to heuristically make redeployment decisions based upon an approximation of the value of being in a particular system configuration. Andersson (2005) and Andersson and Vaerband (2007) take this approach through the use of a "preparedness function" that attempts to measure the capacity of a particular configuration to answer future calls. This preparedness function is similar to the value function computed by dynamic programming algorithms but is heuristic in nature.

Another stream of literature related to this work is the literature on ADP. For more background and results related to ADP and ambulance redeployment see Maxwell et al. (2009), Henderson (2009) and the references contained therein.

Our approach to ambulance redeployment offers a number of advantages compared to the current methods. First, unlike the integer programming formulations, our method captures the random behavior of the system since it is based upon a dynamic programming structure. Although our approach is also computationally intense, the bulk of the computation is completed prior to real-time operations so that real-time computations are very fast. Unlike the second class of models, our approach does not rely on dispatchers trying to match a given configuration. Instead, the ADP approach offers decision support in the form of redeployment recommendations, but continues to work seamlessly if dispatchers choose to ignore these requests. Unlike the exact dynamic-programming formulation, our approach is able to handle high-dimensional and even uncountable state spaces and, unlike the use of a heuristic preparedness function, our ADP formulation approximates the value function directly.

The main contribution of this paper is to describe the simulation model used to evaluate EMS performance and show how that model can be used within an iterative scheme to improve allocation policies. As such, this paper can be seen as a companion paper to Maxwell et al. (2009) which focuses on the details involved in the iterative algorithm itself. Additionally, in this paper we empirically show that in a realistic scenario our ADP policy for ambulance redeployments is able to greatly improve upon the performance of a sub-optimal ambulance allocation policy but is only able to marginally improve upon a near-optimal allocation policy. Even so, the ADP policy may be less difficult to compute than finding a near-optimal allocation policy.

This paper is organized as follows, Section 2 describes the components of a discrete-event simulation which is used to evaluate the performance of a specific allocation policy, Section 3 introduces the concept of a dynamic allocation policy and provides an algorithm to obtain such a policy based upon the model described in Section 2, and Section 4 provides some representative computational results.

2 SIMULATION MODEL

We evaluate the performance of EMS systems using a discrete-event system encompassing the entire arrival/service cycle of an emergency call: (1) an emergency call arrives, (2) an ambulance travels to the scene, (3) paramedics treat the patient at the scene, (4) the ambulance transports the patient to a hospital, (5) the paramedics transfer the patient to hospital staff, (6) the ambulance returns to a base. The main source of randomness in the simulation comes from the number and location of arriving calls as well as the service times both at the scene of the emergency and at the hospital. The simulation has a finite planning horizon which we take to be two weeks.

Often the performance of EMS providers as well as contract requirements are measured by the percentage of calls responded to within a certain amount of time. Our model adopts this approach as a means to evaluate the performance of a given allocation policy. Consequently, any calls not responded to within the time threshold (which we take to be eight minutes, in accordance with many EMS organizations) are considered "lost" in the sense that the performance criterion was not met. Calls are served in a first-come, first-served fashion, so that calls that have already been "lost" are not postponed or abandoned in favor of other calls for which the service criterion may be met.

2.1 Physical Components

Modeling EMS operations within a city mandates inclusion of essential physical components such as ambulance bases, hospitals, and the city-wide transportation network. Furthermore, information about the location, schedule, and status of operating ambulances is also crucial.

Every ambulance is assumed to have a "home" base, or a base to which the ambulance returns after finishing a call. These designated bases define a (static) allocation policy. However, to more closely resemble actual EMS operations, each ambulance has an associated schedule that defines when the ambulance is on or off duty and when an ambulance transfers to a new home base. Throughout the simulation the location and status of each ambulance is tracked; the status of an ambulance is either "off duty", "idle at base", "going to scene of call", "serving at scene of call", "going to hospital", "transferring patient to hospital", or "returning to base." We assume that any ambulance has the capacity to serve any emergency call.

Additionally, we assume that ambulances travel at two different (constant) speeds corresponding to whether or not they are responding to a call.

The road network in our model mimics an actual city transportation network on the avenue level. Travel times are deterministic and do not depend upon the time of day. Travel time to a point off the road network is calculated using the Manhattan distance from the nearest node of the road network to the specified point and an appropriate travel speed. Additionally, to decrease the runtime of the simulation, the shortest path between every two nodes on the road network is computed and stored before running any simulations.

2.2 Call Arrival and Service Dynamics

To model call arrivals we split the city into an appropriate-sized grid and assume that each cell l within the grid has a time-dependent arrival rate $\Lambda_l(t)$. Calls arrive uniformly within cell l according to a Poisson process with rate $\Lambda_l(t)$. Consequently, the arrival process for the whole city is an inhomogeneous Poisson process with rate $\Lambda(t) = \sum_{l \in \mathscr{L}} \Lambda_l(t)$, where $\mathscr{L}$ represents the set of all cells.

Calls are served in a first-come, first-served fashion with calls being placed on a waiting list if there are no available ambulances. If multiple ambulances are available when a call arrives the closest available ambulance is dispatched to the call. An ambulance, upon arriving at a call scene, treats the patient for an exponentially distributed amount of time with a mean of 12 minutes. After treating the patient at the call scene, the ambulance transports the patient to a hospital with probability 0.75. The destination hospital is determined randomly according to a cell-dependent probability distribution as estimated from historical data. The time an ambulance spends at the hospital has a Weibull distribution with mean 30 minutes and standard deviation 13 minutes. Upon finishing a call, the ambulance returns to its "home" base if following a static allocation policy or transfers to an assigned base if following a dynamic allocation policy (see Section 3.1). Additionally, the turn-out time is assumed to be 45 seconds, i.e. if an ambulance crew is at a base when notified of a call, then it takes 45 seconds to get on the road. An ambulance crew already on the road does not incur turn-out time.

2.3 Random Number Generation

The random number generator we use in the simulation is the RngStream package as described in L'Ecuyer et al. (2002). This random number generator facilitates the use of many long random number streams and substreams to enhance synchronization between different elements of the simulation. In our implementation the call arrival times, call arrival locations, and hospital requirement probabilities use distinct random number streams. Further synchronization details are discussed in Section 3.2.2.

3 POLICY OPTIMIZATION

The simulation model described in Section 2 is not limited to just evaluating the performance of a single allocation policy, but can also be used as the central component of an iterative optimization algorithm. Formulating the ambulance redeployment problem as a dynamic program provides us with computational algorithms to improve an allocation policy. Unfortunately, the complexity of the problem renders exact dynamic programming intractable. As a result, we use approximate dynamic programming as a computationally efficient method to obtain policies which, while not guaranteed to be optimal, may improve upon the current allocation policy.

3.1 Dynamic Allocation Policies

Until this point we have mostly discussed static policies, i.e., policies in which ambulances return to their home base after completing a call if no calls are on the waiting list. As an extension, one might take an ambulance's home base to be time-dependent; see Rajagopalan, Saydam, and Xiao (2008). Another alternative is to consider transferring a newly freed ambulance to any base in the city depending on current system conditions. To be more precise, one could consider moving an ambulance to the base that would minimize the expected number of missed calls over the remaining time horizon. Procedures that attempt to do this are known as "ambulance redeployment" procedures. An allocation policy that uses ambulance redeployments will be referred to as a dynamic policy or dynamic allocation policy.

It is also possible to consider additional ambulance redeployments that transfer idle ambulances from one base to another at times other than call completion times. Although this approach may add additional improvements it is more disruptive for crews and requires them to spend more time in the vehicles than they might prefer. As a result, we consider an ambulance as available for redeployment only immediately after it finishes transferring a patient to a hospital (or finishing at the call scene if hospital transport is unnecessary). Ambulances that are idle at the bases or moving to different locations are not considered for redeployment.

3.2 Approximate Dynamic Programming Approach

The ADP approach begins by formulating the ambulance redeployment problem as a Markov decision process. Given this formulation (and under certain conditions) dynamic programming algorithms may be used to calculate the optimal value function and hence find an optimal policy. Unfortunately, for our application, this formulation requires a high-dimensional and uncountable state space, and hence exact dynamic programming algorithms are computationally intractable and cannot be used directly. To deal with this difficulty we approximate the value function via a linear approximation architecture characterized by a few parameters. We then iteratively tune these parameters by simulating trajectories of the system and updating the parameters to fit the observed values. In this sense our approach is similar to the policy iteration algorithm for dynamic programming. In particular, simulating trajectories under a set of parameters is analogous to policy evaluation and updating the parameters in response to the observed values is analogous to policy improvement.

3.2.1 Ambulance Redeployment as a Markov Decision Process

Formulating a model as a Markov decision process (MDP) provides a framework for which dynamic programming algorithms can be used to compute optimal policies. This formulation requires the definition of a state space, a control space, system dynamics, transition costs, and an objective function. Precise details on how to formulate the ambulance redeployment problem as a MDP can be found in Maxwell et al. (2009). We give a brief summary of each of these elements below.

The state of the system s includes the status, location, and destination of each ambulance as well as the time the ambulance began traveling from the location to the destination (required to make deterministic travel times Markovian). If an ambulance is not moving, the destination and starting time can be ignored. The state also includes the location, arrival time, and status of emergency calls (whether an ambulance has been assigned to the call or not) for which an ambulance has not yet arrived on scene. Additionally, the state also includes the current time of the simulation and the most recent event from the discrete-event simulation.

The control space of the system depends upon the current state of the system. If we are considering an ambulance redeployment in the current state, i.e., an ambulance has finished servicing a call, is still on duty, and there are no calls on the waiting list, then the possible controls are to redeploy the newly freed ambulance to any base within the city. If we are not considering an ambulance redeployment in the current state, then the only control available is the “default” control which signifies that the simulation should continue uninterrupted. We denote a decision as x and the set of all redeployment decisions available in state s as $\mathscr{X}(s)$. States in which we are considering an ambulance redeployment will be called decision states.

The system dynamics dictate the probability distribution of the subsequent state for any given state in the system. For our system, these probability distributions are not easily described in mathematical notation, but they are easily expressed in terms of our discrete-event simulation. In this sense, the system dynamics are defined implicitly through the random processes and event-driven nature of the discrete-event system. We capture the dynamics of the system symbolically by $s_{k+1} = f(s_k, x_k, \omega(s_k, x_k))$, where s_k is the state of the system at the time of the kth event and x_k is the decision made by the dispatcher (if any) when the state of the system is s_k, $\omega(s_k, x_k)$ is a random element of an appropriate space encapsulating all the sources of randomness in the simulation, and $f(\cdot,\cdot,\cdot)$ is the transfer function.

The ADP framework is amenable to many different transition cost structures and additive objective functions. For our application we have chosen to use an indicator function indicating whether the current event resulted in an ambulance being assigned to an emergency call that it cannot reach within the time threshold. Since we assume deterministic travel times and no preemption, the sum of these indicator functions will equal the total number of calls not responded to within the time threshold. As such we use the total expected “lost” calls over our time horizon as the objective function and attempt to find a dynamic policy to minimize this quantity. We denote the transition cost from state s_k to state s_{k+1} through decision x_k as $c(s_k, x_k, s_{k+1})$.

One criticism of this cost structure is that calls responded to immediately after the time threshold and far beyond it are weighted equally even though the medical outcome of such “lost” calls are unlikely to be equivalent; see Erkut, Ingolfsson, and Erdoğan (2008). Nevertheless, empirical results indicate that under this cost structure the entire response-time distribution of calls is shifted to the left, i.e., if T is a generic random response time, then $P(T > t)$ is reduced for all t and not just near the time threshold $t = 8$ minutes; see Section 4.3.

3.2.2 Optimality Equation

In a dynamic programming context, a policy μ is a map from the state space to the control space that prescribes what action to take for any given state. For our application this corresponds to a mapping that dictates which base to redeploy a newly-freed ambulance to for any possible decision state. Associated with any policy μ and any starting state s is the value function

$$J^{\mu}(s) = \mathbb{E}\left[\sum_{k=1}^{\infty} c\left(s_k^{\mu}, \mu\left(s_k^{\mu}\right), s_{k+1}^{\mu}\right) \middle| s_1^{\mu} = s\right],$$

where $c(s_k^{\mu}, \mu\left(s_k^{m}u\right), s_{k+1}^{\mu})$ is defined to be zero if the time of the event s_k^{μ} is greater than or equal to the specified time horizon. In our context the value function $J^{\mu}(s)$ expresses the expected number of lost calls that will be incurred from state s until we reach the end of the simulation horizon when following policy μ. An optimal policy μ^* can be found by computing the value function through the recursive optimality equation

$$J(s) = \min_{x \in \mathscr{X}(s)} \left\{ \mathbb{E}\Big[c(s,x,f(s,x,\omega(s,x))) + J(f(s,x,\omega(s,x)))\Big] \right\} \tag{1}$$

and by letting $\mu^*(s)$ be a feasible action that minimizes the right-hand side of (1); see Bertsekas and Shreve (1978).

Dynamic programming algorithms provide a method to compute $J(s)$ using (1); however, exact dynamic programming solutions are computationally intractable except for problems with rather small state spaces. In our application not only do we have a high-dimensional state space, the state space is actually uncountable, hence storing the function $J(\cdot)$ is impossible even if it could be computed. Furthermore, exact dynamic programming methods require the expectation of the transition costs $c(s,x,f(s,x,\omega(s,x)))$ and the probability distribution of future states $f(s,x,\omega(s,x))$, neither of which is known.

To overcome the difficulty of expressing the value function $J(\cdot)$, a common ADP approach is to use a linear approximating architecture of the form $J(\cdot,r) = \sum_{p=1}^{P} r_p\,\phi_p(\cdot)$ where $\{\phi_p(\cdot) : p = 1,\ldots,P\}$ are fixed basis functions representing the relevant features of the state space and $\{r_p : p = 1,\ldots,P\}$ are tunable parameters used to fit the approximation architecture to the value function; see Bertsekas and Tsitsiklis (1996) and Powell (2007). A number of methods exist to tune the parameters $\{r_p : p = 1,\ldots,P\}$ so that $J(\cdot) \approx J(\cdot,r)$ such as temporal-difference learning and Q-learning; see Sutton (1988), Watkins and Dayan (1992), Tsitsiklis (1994), Bertsekas and Tsitsiklis (1996), Tsitsiklis and Van Roy (1997), and Si et al. (2004).

The five basis functions $\{\phi_p(\cdot) : p = 1,\ldots,5\}$ used in our application are the time remaining until the end of the horizon, an approximation of the current ambulance coverage and Erlang loss over the city, and approximations of the ambulance coverage and Erlang loss over the city at a conceptual state in the near future which assumes that every ambulance that is returning to a base reaches that base before any new calls arrive. A more detailed description of these basis functions can be found in Maxwell et al. (2009).

To overcome the difficulty of computing the expectation and minimization contained in (1) we use Monte Carlo simulation. In particular, if the current state of the system is s we enumerate over all of the feasible decisions contained in $\mathscr{X}(s)$. For each decision x we simulate the system forward one event. This one-step simulation provides us with a sample of $f(s,x,\omega(s,x))$ and hence allows us to compute a sample of $c(s,x,f(s,x,\omega(s,x)))$ and $J(f(s,x,\omega(s,x)))$ (or $J(f(s,x,\omega(s,x)),r)$ when using an approximation architecture). Using the average of these sample values provides us with an estimate of the expectation contained in (1) for each decision. We then approximate the minimization by choosing the smallest sample average.

One limitation to the above sampling method is that the time of state s and the time of the subsequent state $f(s,x,\omega(s,x))$ may be very close together. Under these conditions one-step simulation gives very little information on the effects of choosing action x. An alternative method is to simulate the effect of a decision from the current state s until the next decision state s' instead of the one step transition $f(s,x,\omega(s,x))$. We use this alternative method in our Monte Carlo estimation.

We call these extended simulations “micro simulations” and implement them by generating independent simulations of the system from state s that stop once a decision state is reached. For a given redeployment decision, each micro simulation is implemented using a different random number substream. These random number substreams are synchronized over each feasible redeployment decision. This formulation facilitates the use of common random numbers among the different possible decisions despite the fact that micro simulations are likely to have different lengths (in terms of random numbers generated). The variance reduction provided by this formulation increases the probability that an optimal decision will be found when estimating the minimizer of the right-hand side of (1) using Monte Carlo estimates of the true expectations.

3.2.3 Approximate Policy Iteration

The ADP approach is a two-stage approach to policy optimization. In the first stage the algorithm runs iteratively to tune (or train) the parameters $\{r_p : p = 1,\ldots,P\}$ so that they fit the value function well. This computationally intense training process can take a considerable amount of time, but can be completed offline before any redeployment decisions need to be made. In the second stage the parameters obtained in the first stage are used to determine redeployment decisions in real-time operations. As such, it is important that real-time deployment decisions be computed quickly.

Using a linear approximation architecture and Monte Carlo sample averages of the expectations in (1) results in the following approximate policy iteration algorithm to tune (or train) the parameters $\{r_p : p = 1,\ldots,P\}$, as presented in Maxwell et al. (2009).

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 $J(\cdot, r^n)$, i.e.,

$$\mu^n(s) \in \underset{x\in\mathscr{X}(s)}{\operatorname{argmin}}\left\{\frac{1}{N}\sum_{i=1}^{N}\Big(c(s,x,s_i') + J(s_i', r^n)\Big)\right\}, \tag{2}$$

where N is the number of micro simulations sampled and s_i' denotes the decision state following state s in the ith micro simulation.

Step 3. (Policy evaluation through simulation) Simulate the trajectory of policy μ^n over the planning horizon for Q replications. Let $\{s_k^n(q) : k = 1,\ldots,K(q)\}$ be the state trajectory of policy μ^n in replication q and $C_k^n(q)$ be the cost incurred by starting from state $s_k^n(q)$ and following policy μ^n in replication q.

Step 4. (Least squares projection) Compute the tunable parameters at the next iteration as

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

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

Once training is completed and a suitable set of parameters r^* are found, the resulting policy is implemented by evaluating (2) with r^* to make real-time decisions. The computational complexity of making a decision is equivalent to simulating until the next decision event and computing the value function approximation at the resulting state once for each micro simulation for each base in the city. In our application the decision time is well under one second for a reasonable number of micro simulations (e.g. 100), which is more than sufficient for real-time operations.

The least squares projection in Step 4 of the algorithm is not the only method for tuning the parameters r to fit value function approximation $J(s_k^n(q), r)$ to the sample-trajectory costs $C_k^n(q)$. As mentioned in Section 3.2.2, alternative methods include temporal-difference learning and Q-learning.

3.3 Discussion

There are a number of considerations involved when considering a redeployment policy based on ADP as compared to a static policy.

First, an ADP approach is more complex to implement than a static policy, and may be perceived as not worth the trouble. But the potential increases in performance from ADP are viewed as more than sufficient to warrant the effort.

Second, an ADP approach requires system-wide information and central communication to make a decision. For EMS operations this is generally not a problem since the infrastructure for these elements is already in place through computer-aided dispatch systems.

Third, there is no guarantee that an ADP policy will perform better than a given static policy due to the use of *approximations* of the value functions. (Such a guarantee does hold when using exact dynamic programming.) Furthermore, nothing is known about the optimality gaps of static and ADP policies. Nevertheless, one can compare such policies through simulation. Although the computational effort required to evaluate one static allocation policy is less than that required to evaluate an ADP policy, the computational effort required to find a highly effective static policy is much greater than that required to find a highly effective ADP policy because of the combinatorial search involved.

Fourth, the performance of an ADP policy is intrinsically linked to how well the approximation architecture matches the actual value function. Specifically, for a linear approximation architecture the choice of fixed basis functions $\{\phi_p(\cdot) : p = 1,\dots,P\}$ will determine the effectiveness of the ADP policy. This performance can range from very poor to optimal. Although there is no general purpose method to choose basis functions, our computational experience leads us to believe that the basis functions contained in Section 3.2.2 are sufficient to discover good ADP policies for the ambulance redeployment problem. More effective basis functions may exist, and may, in fact, be essential in more complex simulation models that take into account other realistic aspects of these systems such as level of care.

4 COMPUTATIONAL RESULTS

This section contains details and results for a realistic example. Two different static policies are used to benchmark performance. The first is termed the "sub-optimal" policy and represents a static policy that performs fairly well in this scenario. This policy was designed heuristically by attempting to minimize the ambulance utilizations. The second is termed the "near-optimal" policy and represents a static policy that performs very well in this scenario. This policy was obtained by performing a search over a large set of possible static policies that were likely to contain the optimal static policy. The performance of these static policies is compared with the performance of the policies obtained through the ADP approach using each of these static policies as the starting point in the ADP iteration. Additionally, we provide results for the performance of an ADP policy using different numbers of micro simulations. Finally, we give run times for different aspects of the computation.

4.1 Experimental Setup

The computational results in this paper are based upon data for the city of Edmonton, Alberta in Canada as studied in Ingolfsson, Erkut, and Budge (2003). The city has a population over 800,000 and has an area of approximately 40×30 km^2. The EMS system has 16 ambulances, 11 bases, and 5 hospitals. The road network in the simulation models the actual road network and contains 252 nodes and 934 arcs.

We did not have sufficient data to generate an accurate call arrival model. As a result we designed a representative call model with sinusoidal fluctuations in the call arrival rates of each cell and a constant overall arrival rate. These fluctuations are designed so that regions within the city proper are more active during the day and regions without the city are more active during the night. The intensity of the fluctuations were determined from historical data as was the overall arrival rate of 4 calls per hour. Since we keep the call arrival rate constant we also assume that all 16 ambulances operate on a 24 hour schedule. More details of the call arrival process for this scenario can be found in Maxwell et al. (2009).

4.2 Performance Comparison

In Figure 1 we show the performance of the policies identified in two training runs each consisting of 25 iterations of the approximate policy iteration algorithm. The horizontal axis shows the iteration number, and the vertical axis shows the average percentage of lost calls. The two training runs differ in the policy each was initiated with: one is initiated with the sub-optimal policy and the other is initiated with the near-optimal policy. Both runs identify policies that outperform both static policies. Also, the ADP policy initiated with the sub-optimal static policy is able to obtain nearly the same performance as the ADP policy initiated with the near-optimal static policy after just a few iterations. There is a performance increase of nearly 5% over the sub-optimal static policy and about .5% improvement over the near-optimal static policy. This indicates that the ADP approach is especially useful in situations where an optimal static policy is not known, because there is then an increased potential for performance gains.

Further investigation into the call model shows that approximately 18.6% of emergency calls are located over 8 minutes from any base. Such calls will almost certainly be categorized as lost calls regardless of which base an ambulance is dispatched from. This does not provide a strict lower bound on the performance of any allocation policy, however, since an ambulance that is dispatched while traveling may still be able to reach these calls within the time threshold. Nevertheless, it serves as a good proxy for a lower bound and hence a strong indicator of the maximum improvements we might be able to expect.

4.3 Response Time Comparison

In Figure 2 we show the empirical cumulative histogram of response times for each policy considered. We note that the proportion of calls responded to for any given time threshold is higher for the two ADP policies than for either of the static policies. This indicates that the ADP polices are not only effective at reducing the average percentage of lost calls, but they reduce the average response times overall.

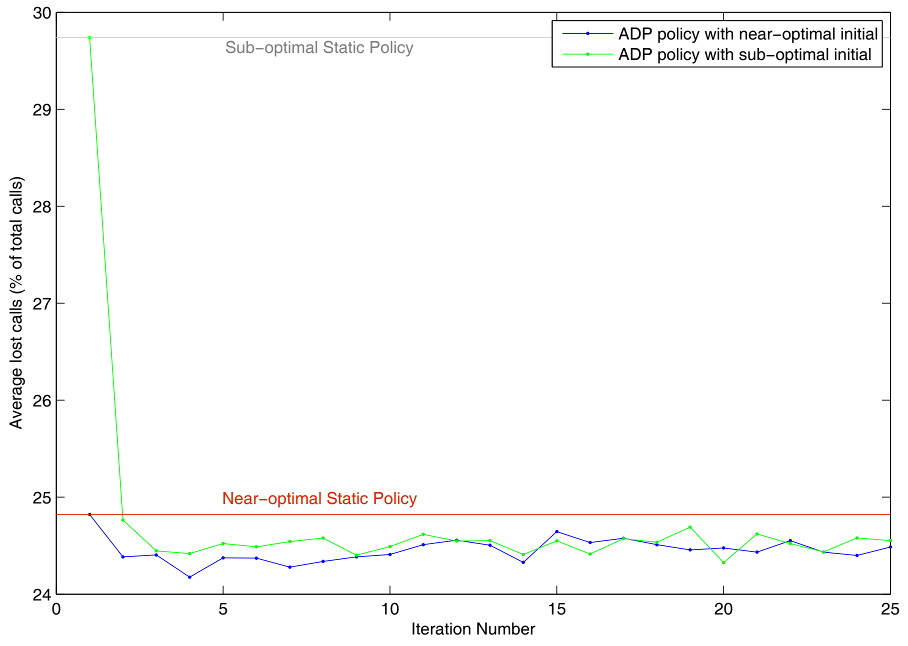

Figure 1: Comparative performance of four policies.

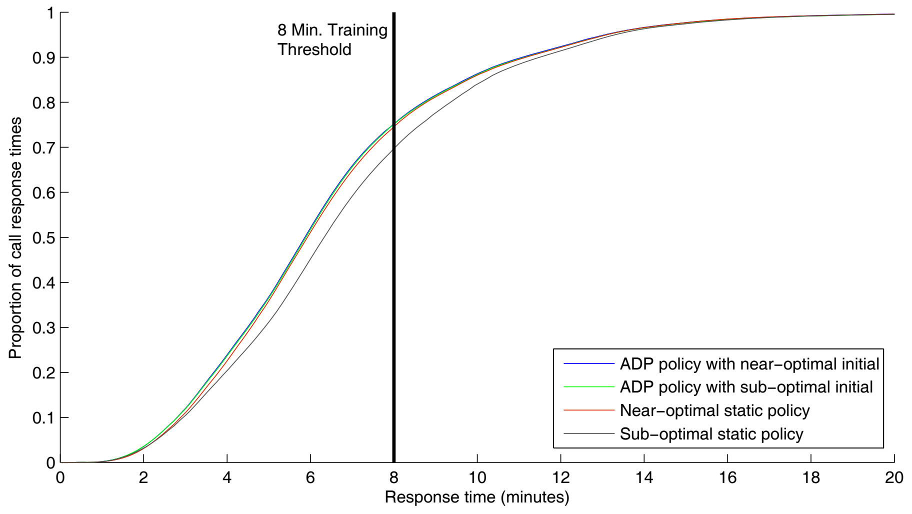

Figure 2: Comparative response time of four policies.

4.4 Effect of Micro Simulations on the ADP Policy

Implementing an ADP policy requires estimating the expectation on the right-hand side of (1) for every feasible action through micro simulations. Hence the decisions made by an ADP policy are a function of the number of micro simulations used to estimate the expectations. In Figure 3 we show the performance of an ADP policy as a function of the number of micro simulations used in decision evaluations. We note that with as little as 5 micro simulations per feasible decision the ADP policy is able to outperform the near-optimal static policy. Additional improvements of about .7% are obtained through an increased number of micro simulations with improvements apparently stagnating at around 60 micro simulations.

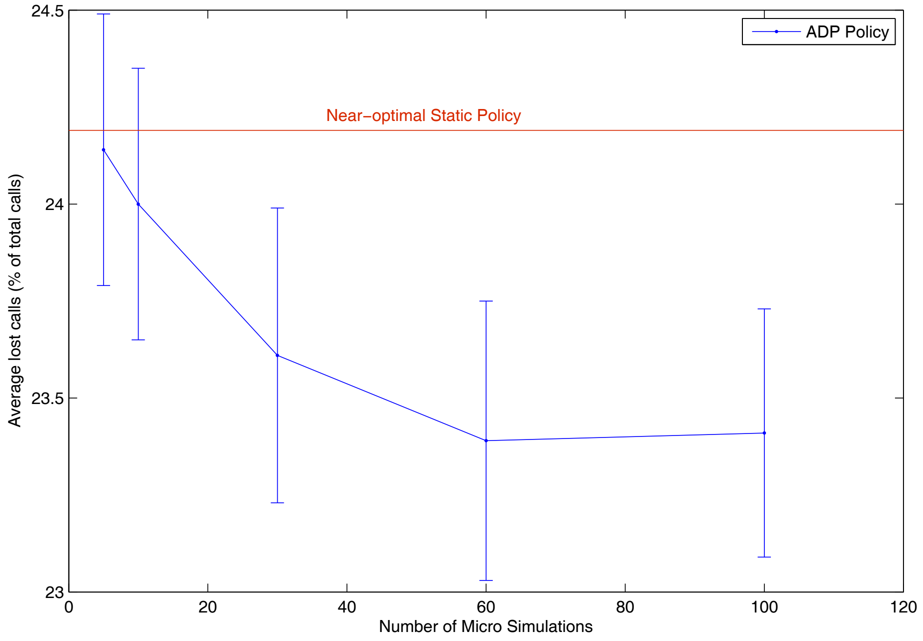

Figure 3: Performance of our ADP policy evaluation as a function of the number of micro simulations. The error bars indicate pointwise 95% confidence intervals.

4.5 Computation Time

The runtime of an ADP policy also depends on the number of micro simulations used during decision evaluations. In Table 1 we give approximate run times for an ADP policy using different numbers of micro simulations. We note that the decision times for the ADP policy are well below an acceptable level for real-time decisions even when 100 micro simulations are used, although the training time for this many micro simulations is prohibitive. The simulation time represents one simulation of two weeks, the iteration time is comprised of 30 simulation replications, and the training time is comprised of 25 iterations. For comparison the simulation time of a static policy is .11 seconds.

Table 1: Runtime of our ADP policy as a function of the number of micro simulations.

	5	10	30	60	100
ADP Decision Time (seconds)	0.013	0.025	0.077	0.152	0.253
ADP Simulation Time (seconds)	15	30	89	177	289
ADP Iteration Time (minutes)	7.7	15	45	87	145
ADP Training Time (hours)	3.2	6.3	19	39	60

5 CONCLUSION

We first developed a simulation model to evaluate the performance of allocation policies for EMS operations on a finite time horizon. We then formulated the ambulance redeployment problem as a Markov decision process and used an approximate version of the policy iteration algorithm centered around the simulation model to deal with the high-dimensional and uncountable state space. Computational experiments on a realistic problem scenario show that our ADP approach can provide high-quality redeployment policies. Specifically, the computational experiments indicate that ADP methods are likely to be most useful when a near-optimal static policy is not known, which is often the case for EMS providers.

Future research will proceed in two directions. First, we will incorporate additional elements of realism into the model such as stochastic travel times, multiple levels of care, and emergency calls which require multiple ambulances. Along these lines we will also investigate the potential benefits of stationing idle ambulances at intersections throughout the city instead of only at ambulance bases. Although this practice is common among many EMS providers, the benefit of such a policy (in terms of reduced response times) is not known. Second, we plan on investigating the benefit of variance reduction techniques both on the convergence of the approximate policy iteration algorithm (Section 3.2.3) and on the evaluation of optimal decisions via micro simulations.

ACKNOWLEDGMENTS

We thank Armann Ingolfsson and Andrew Mason for advice, support, and data. This work was partially supported by National Science Foundation Grants DMI 0400287, DMI 0422133, and CMMI 0758441.

REFERENCES

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

Andersson, T., and P. Vaerband. 2007. Decision support tools for ambulance dispatch and relocation. *Journal of the Operational Research Society* 58 (2): 195–201.

Berman, O. 1981a. Dynamic repositioning of indistinguishable service units on transportation networks. *Transportation Science* 15 (2): 115–136.

Berman, O. 1981b. Repositioning of distinguishable urban service units on networks. *Computers and Operations Research* 8 (2): 105–118.

Berman, O. 1981c. Repositioning of two distinguishable service vehicles on networks. *IEEE Transactions on Systems, Man, and Cybernetics* SMC-11 (3): 187–193.

Bertsekas, D., and S. Shreve. 1978. *Stochastic optimal control: The discrete time case.* New York: Academic Press.

Bertsekas, D., and J. Tsitsiklis. 1996. *Neuro-dynamic programming*. Belmont, Massachusetts: Athena Scientific.

Brotcorne, L., G. Laporte, and F. Semet. 2003. Ambulance location and relocation models. *European Journal of Operations Research* 147 (3): 451–463.

Erkut, E., A. Ingolfsson, and G. Erdoğan. 2008. Ambulance deployment for maximum survival. *Naval Research Logistics* 55:42–58.

Gendreau, M., G. Laporte, and S. Semet. 2001. A dynamic model and parallel tabu search heuristic for real time ambulance relocation. *Parallel Computing* 27 (12): 1641–1653.

Gendreau, M., G. Laporte, and S. Semet. 2006. The maximal expected coverage relocation problem for emergency vehicles. *Journal of the Operational Research Society* 57 (1): 22–28.

Goldberg, J. B. November 4, 2007. Personal Communication.

Henderson, S. G. 2009. Operations research tools for addressing current challenges in emergency medical services. *Encyclopedia of Operations Research and Management Science*. To appear.

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

Ingolfsson, A., E. Erkut, and S. Budge. 2003. Simulation of single start station for Edmonton EMS. *Journal of the Operational Research Society* 54 (7): 736–746.

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

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

Maxwell, M. S., M. Restrepo, S. G. Henderson, and H. Topaloglu. 2009. Approximate dynamic programming for ambulance redeployment. *INFORMS Journal on Computing*. To appear.

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

Powell, W. B. 2007. *Approximate dynamic programming: Solving the curses of dimensionality*. Hoboken, NJ: John Wiley & Sons.

Rajagopalan, H. K., C. Saydam, and J. Xiao. 2008. A multiperiod set covering location model for dynamic redeployment of ambulances. *Computers & Operations Research* 35:814–826.

Restrepo, M. 2008. *Computational methods for static allocation and real-time redeployment of ambulances*. Ph.D. thesis, Cornell University, Ithaca, New York.

Si, J., A. G. Barto, W. B. Powell, and D. Wunsch II. (Eds.) 2004. *Handbook of learning and approximate dynamic programming*. Piscataway, NJ: Wiley-Interscience.

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

Tsitsiklis, J., and B. Van Roy. 1997. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control* 42 (5): 674–690.

Tsitsiklis, J. N. 1994. Asynchronous stochastic approximation and Q-learning. *Machine Learning* 16 (3): 185–202.

Watkins, C. J. C. H., and P. Dayan. 1992. Q-learning. *Machine Learning* 8 (3-4): 279–292.

Zhang, O., A. J. Mason, and A. B. Philpott. 2008. Simulation and optimisation for ambulance logistics and relocation. Presentation at the INFORMS 2008 Conference.

AUTHOR BIOGRAPHIES

MATTHEW S. MAXWELL is a Ph.D. candidate in the School of Operations Research and Information Engineering at Cornell University. He has a B.Sc. in Computer Science from Brigham Young University and is a recipient of a Graduate Fellowship from the U.S. Department of Homeland Security. His research interests include discrete-event simulation, simulation optimization, and approximate dynamic programming.

SHANE G. HENDERSON is a professor in the School of Operations Research and Information Engineering at Cornell University. He is the simulation area editor at *Operations Research*, and an associate editor for the *ACM Transactions on Modeling and Computer Simulation* and *Operations Research Letters*. He co-edited the handbook *Simulation* as part of Elsevier's series of Handbooks in Operations Research and Management Science, and also co-edited the Proceedings of the 2007 Winter Simulation Conference. He likes cats but is allergic to them. His research interests include discrete-event simulation and simulation optimization, and he has worked for some time with emergency services. His web page can be found via <http://www.orie.cornell.edu>.

HUSEYIN TOPALOGLU an associate professor in the School of Operations Research and Information Engineering at Cornell University. He holds a B.Sc. in Industrial Engineering from Bogazici University in Turkey, and a Ph.D. in Operations Research and Financial Engineering from Princeton University. His research interests include stochastic programming and approximate dynamic programming with applications in transportation logistics, revenue management and supply chain management. He teaches courses on dynamic programming, simulation modeling, systems engineering and revenue management. Huseyin Topaloglu is currently serving as associate editor for *IIE Transactions*, *Mathematical Programming Computation* and *Operations Research*.