Proceedings of the 2010 Winter Simulation Conference
B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, eds.

IDENTIFYING EFFECTIVE POLICIES IN APPROXIMATE DYNAMIC PROGRAMMING: BEYOND REGRESSION

Matthew S. Maxwell
Shane G. Henderson
Huseyin Topaloglu

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

ABSTRACT

Dynamic programming formulations may be used to solve for optimal policies in Markov decision processes. Due to computational complexity dynamic programs must often be solved approximately. We consider the case of a tunable approximation architecture used in lieu of computing true value functions. The standard methodology advocates tuning the approximation architecture via sample path information and regression to get a good fit to the true value function. We provide an example which shows that this approach may unnecessarily lead to poorly performing policies and suggest direct search methods to find better performing value function approximations. We illustrate this concept with an application from ambulance redeployment.

1 INTRODUCTION

One common approach for approximate dynamic programming (ADP) is to approximate the true value function as a linear combination of fixed "basis" or "feature" functions which attempt to describe the main features of the model. The weights on these functions are tuned so that the approximation is close to the true value function. This approximation architecture is then used in lieu of the true value function when making decisions via the dynamic programming optimality equation; Bertsekas and Tsitsiklis (1996) and Powell (2007) are excellent resources on this topic.

This tuning approach is generally justified by theoretical results that bound the performance of a greedy policy with respect to a given approximation by the discrepancy between the approximate value function and the true value function (see Bertsekas and Tsitsiklis 1996 for details). Unfortunately, such bounds usually hinge on the state with the largest discrepancy between the two value functions. For dynamic programs with very large state spaces this distance is likely to be considerable, and hence these bounds for ADP policies are quite weak.

Another approach would be to perform simulation optimization over the tunable parameters directly. Often this direct search method is more computationally expensive than the regression-based tuning, but it has the distinct advantage of tuning the parameters to maximize performance directly rather than tuning the parameters to better approximate the value function, which may or may not increase overall performance. One example of the benefits to this approach is found in Szita and Lörincz (2006) where a noisy cross-entropy method is used to obtain policies for the game of Tetris which perform over 30 times better than policies found using standard ADP approaches.

Additionally, direct search methods may lead to policies superior than those obtainable through regression-based tuning. In Section 3 we give an illustrative example of this situation. In this example the optimal policy lies within the space spanned by the approximation architecture; however, regression-based tuning approaches will always yield a non-optimal policy. In Section 4 we consider

a case study of tuning ADP policies for ambulance redeployment. The comparison of different tuning methods in this case study show similar results as that observed in the illustrative example. Section 2 is a brief review of dynamic programming (DP) and ADP topics that will be used within this document.

2 BACKGROUND

Consider a stochastic shortest path problem where the objective is to minimize the expected cumulative transition cost from any starting state s to the absorbing state s_a. Superficially this appears to be a minimization over a possibly infinite sequence of decisions; however, using DP recursion we can reformulate this minimization problem into a simpler form. Let $J(s)$ denote the value of being in state s, i.e. the expected cumulative transition cost from s to s_a. The DP optimality equation states that we need only to optimize over the combined transition cost and value function of possible future states to make optimal decisions. Specifically, an optimal policy can be found by choosing a decision in each state that minimizes the sum of the transition cost and the expected value function of the subsequent state. The DP literature contains standard algorithms to compute $J(s)$ for all s such as value iteration and policy iteration, see, e.g., Bertsekas (2005).

With this recursive formulation, any function f on the state space yields a policy—simply choose the decision that minimizes the combined transition cost and expected value of $f(s')$, where s' denotes the (possibly random) subsequent state. Thus the function f is used in lieu of J for making decisions. This policy is called the greedy policy with respect to f. In an ADP setting it is often common to model f as a linear combination of basis functions over the state space, where the coefficients are described herein as tunable weights.

In DP algorithms it is generally assumed that $J(s)$ is calculated for each s. In the ADP setting this is often infeasible because the state space may be very large or even uncountable. Hence the standard DP methods for calculating $J(s)$ can not be applied directly to the approximating function $f(s)$ in ADP. Instead, ADP usually uses similar methods which attempt to tune the coefficients of $f(s)$ to approximate $J(s)$ well. One common approach is to use sample path data (i.e. sample path-based estimates of $J(s)$) as regressors in a least-squares fitting over the weighting coefficients of $f(s)$. Section 4.5 contains a detailed description of one such method.

3 AN ILLUSTRATIVE EXAMPLE

Consider the stochastic shortest path Markov decision process (MDP) shown in Figure 1 where state 0 is an absorbing state and the objective is to minimize the sum of the transition costs from state 3 to state 0. In this MDP there is only one decision available, from state 3 go to state 1 or state 2, and hence there are only two deterministic policies, π_1 and π_2, which denote choosing to go to state 1 or state 2 from state 3 respectively. Since we are trying to minimize the sum of the transition costs, π_1 is optimal.

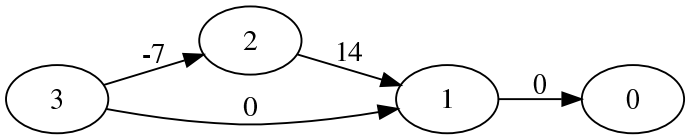

Figure 1: Example MDP

Now consider the basis function $\phi(s) = s$ and the approximation architecture $J_r(s) = r\phi(s) = rs$ for some tunable parameter r. If (and only if) $r > 7$, then $-7 + J_r(2) = -7 + 2r > r = J_r(1)$ and the greedy policy with respect to J_r is equivalent to π_1 and hence optimal. Thus the optimal policy is within the span of policies which are greedy with respect to J_r. In some sense, this indicates that our approximation architecture J_r is rich enough to induce an optimal policy provided that the value of r is chosen sufficiently well. Unfortunately, standard approaches for tuning r are unable to do this.

Let $J^{\pi_i}(s)$ for $i = 1,2$ denote the value of being in state s and following policy π_i. To fit r via regression we would first fix a policy π_i and fit the coefficient via

$$r^*_{i,p} = \operatorname*{argmin}_{r\in\mathbb{R}} \sum_{s=0}^{3} |J^{\pi_i}(s) - rs|^p, \tag{1}$$

where $r^*_{i,p}$ denotes the tuning coefficient when π_i is used for sampling states and regression is performed using a $1 \le p \le \infty$ norm.

Proposition 1. *For $i = 1,2$ and $1 \le p \le \infty$ the greedy policy with respect to $J_{r^*_{i,p}}$ is π_2, i.e. is not optimal.*

Proof. Assume $1 \le p < \infty$. The case where $p = \infty$ can be proved in similar fashion to what follows.

$$\begin{aligned}\sum_{s=0}^{3} |J^{\pi_1}(s) - rs|^p &= |0-0|^p + |0-r|^p + |14-2r|^p + |0-3r|^p \\ &= (1+3^p)\,|r|^p + 2^p\,|7-r|^p \end{aligned} \tag{2}$$

For any $r > 7$ both terms of (2) can be reduced by decreasing r until $r = 7$. Hence we have that $r^*_{1,p} \le 7$. Now assume $r > 0$ (otherwise we already know the greedy policy with respect to $J_{r^*_{i,p}} = \pi_2$). Then

$$\begin{aligned}\lim_{r\uparrow 7}\left(\frac{d}{dr}\left((1+3^p)\,|r|^p + 2^p\,|7-r|^p\right)\right) &= \lim_{r\uparrow 7}\left(p(1+3^p)r^{p-1} - p2^p(7-r)^{p-1}\right) \\ &= p(1+3^p)7^{p-1} \\ &> 0\end{aligned}$$

Since the slope of (2) as $r \uparrow 7$ is strictly positive for all $1 \le p < \infty$ we know that (2) can be reduced by setting $r = 7-\varepsilon$ for some $\varepsilon > 0$. Hence $r^*_{1,p} < 7$ and the greedy policy with respect to $J_{r^*_{1,p}} = \pi_2$.

Similar analysis holds for $r^*_{2,p}$ as well. □

Hence, regardless of our starting policy and choice of p norm, regression cannot achieve parameters which induce an optimal policy. It is easy to create analogous examples where π_2 has arbitrarily bad performance compared to π_1.

It may be argued that the MDP in Figure 1 is a contrived example and a similar situation is unlikely to happen in practice. We argue that this MDP is a quintessential example of the DP concept of considering both the immediate cost and the cost-to-go when making decisions. The failure of a tuning method to perform well in this simplistic situation raises concerns for the method applied to more complex situations. It may also be argued that the proposed approximation architecture is not suitable for the MDP. It is true that the optimal value function is not within the span of the basis functions (although in this particular case a function that induces an optimal policy is), but that is the same reality faced in nearly all ADP applications. The fact that there is a greedy policy with respect to J_r that yields the optimal policy provides evidence that the approximation architecture is sufficient for this MDP.

4 CASE STUDY: AMBULANCE REDEPLOYMENT

In this section we consider a class of ADP policies for ambulance redeployment in Edmonton, Alberta, Canada as indexed by a vector of tunable parameters r. First, we describe the ambulance redeployment problem and give the MDP formulation of ambulance redeployment. Next, we define the class of ADP policies and the approximating basis functions used within the policies. Using this class of policies and the discrete-event simulation model we tune the parameters r via standard regression-based tuning

algorithms and compare the resulting policies to those found by direct search over r. Ultimately, this case study shows that even when tuning ADP polices in realistic situations we see results similar to those of Section 3.

4.1 Ambulance Redeployment

Two primary roles of ambulance dispatchers are to assign ambulances to respond to incoming emergency calls and to coordinate the positions of idle ambulances in preparation for future calls. Typically the role of assigning ambulances to emergency calls is handled by sending the closest available ambulance to respond to incoming calls. Although this strategy may not be "optimal" it is widely used in practice because it generally performs well and it eliminates potential liability concerns faced by emergency medical service (EMS) organizations utilizing other dispatch methods.

The role of positioning idle ambulances to prepare for future calls is accomplished by redeploying ambulances from one location to another location in real-time to fill holes in the ambulances' coverage area. These ambulance redeployment policies can be quite complex, but despite the complexity significant performance gains can be achieved as shown in Gendreau, Laporte, and Semet (2001), Nair and Miller-Hooks (2009), and Maxwell et al. (2010).

This case study is modeled after the EMS operations of Edmonton, Alberta, Canada. Edmonton is a city of about 800,000 people in a 40×30 km^2 area. The city is served by 16 ambulances and 5 hospitals. We consider a set of 11 possible bases at which idle ambulances may wait incoming calls. We developed a discrete-event simulation model for EMS operations in Edmonton and used this model to compare redeployment policies. A travel model for Edmonton using major streets and a representative arrival process is included in the simulation. See Maxwell, Henderson, and Topaloglu (2009) for additional information on the discrete-event simulation.

4.2 MDP Formulation

Formulating ambulance redeployment as a MDP requires defining a state space, a control space, the system dynamics, the transition costs, and the objective function for the problem. A brief description of these components and the associated notation will be given below, and a more precise description of this formulation can be found in Maxwell et al. (2010).

Let $\mathscr{S}$ denote the state space and $s_k \in \mathscr{S}$ denote the kth realized state in the discrete-event simulation. A state s_k contains all information necessary such that future states are independent of past states conditional upon s_k. For ambulance redeployment this includes the location, destination, and status of all the ambulances as well as the location of emergency calls. Additionally, since we assume constant travel times and non-exponential service times, the time at which these events begin is also included as part of our state. Finally, the current simulation time and the current event from the discrete-event simulation are included in the state as well.

Let $\mathscr{X}(s_k)$ denote the feasible actions in state s_k. If s_k is a decision state, meaning a state in which an ambulance just became free and there are no calls on the waiting list, then $\mathscr{X}(s_k)$ is the set of all ambulance bases indicating the destination base for the newly freed ambulance. If s_k is not a decision state then $\mathscr{X}(s_k)$ contains a single "do nothing" action which has no effect on the state or the system dynamics.

The complex system dynamics of ambulance redeployment are best described implicitly via the discrete-event simulation. Let U_{k+1} denote a random uniform vector of appropriate dimension which will be transformed to include all sources of randomness present between state s_k and s_{k+1}. Then $s_{k+1} = f(s_k, x, U_{k+1})$ where $f(s_k, x, U_{k+1})$ denotes the system dynamics starting in state s_k choosing action $x \in \mathscr{X}(s_k)$ with the random realization defined by U_{k+1}.

The transition cost $c(s_k, x, U_{k+1})$ is one if s_{k+1} corresponds to an emergency call arrival event for which the closest ambulance is over 8 minutes away (i.e. a "lost call") and zero otherwise. This cost function is justified due to the fact that EMS provider contracts are often designed around the proportion of lost calls in a given time period. Despite contractual agreements, one possible

criticism of this cost structure is that calls responded to immediately after 8 minutes and calls responded to much beyond 8 minutes contribute the same penalty although the medical outcomes of the latter is likely to be more severe; see Erkut, Ingolfsson, and Erdoğan (2007). Regardless, Maxwell, Henderson, and Topaloglu (2009) shows that this cost structure is effective at reducing response times uniformly as opposed to only those near or below the 8 minute threshold.

4.3 ADP Policy

Let $J(s_k)$ denote the value of being in state s_k, i.e. the expected cumulative costs from state s_k until the end of the time horizon. The MDP formulation allows us to use DP to solve for $J(s_k)$ via the optimality equation

$$J(s_k) = \min_{x \in \mathscr{X}(s_k)} \mathbb{E}\left[c(s_k, x, U_{k+1}) + J\left(f(s_k, x, U_{k+1})\right)\right], \tag{3}$$

and an optimal policy can be found by choosing an action x that minimizes the right-hand side of (3) (see Bertsekas and Shreve 1978). The uncountable state space of our problem causes (3) to be intractable. Thus we will attempt to approximate the right-hand side of (3) and use this approximation to define our ADP policy.

We approximate the true value function J via the approximation $J_r(\cdot) = \sum_{b=1}^{B} r_b \phi_b(\cdot)$ where the $\phi_b(\cdot)$ are fixed basis functions and $r = (r_1, \ldots, r_B)$ are tunable parameters. Furthermore, since we cannot compute the expectation in (3) exactly we estimate the expectation via Monte Carlo samples. Thus the greedy policy with respect to the approximation J_r for any $s_k \in \mathscr{S}$ is given by choosing a minimizer of

$$\min_{x \in \mathscr{X}(s_k)} \frac{1}{N} \sum_{i=1}^{N} \left(c(s_k, x, u_{k+1}^{(i)}) + J_r\left(f(s_k, x, u_{k+1}^{(i)})\right)\right), \tag{4}$$

where $u_{k+1}^{(1)}, \ldots, u_{k+1}^{(N)}$ denote N realizations of the random variable U_{k+1} (see an improved sampling method in (Maxwell, Henderson, and Topaloglu 2009)). One crucial assumption for solving (4) is that the cardinality of $\mathscr{X}(s_k)$ is small enough that the minimization can be accomplished by evaluating for each $x \in \mathscr{X}(s_k)$. For our application the cardinality of $\mathscr{X}(s_k)$ is at most the number of ambulance bases, and hence the computation required by (4) is feasible for real-time decision support.

4.4 Basis Functions

The basis functions $\phi_b(\cdot)$ used in our approximation architecture are motivated by considering each base individually as an $M/G/K_b/K_b$ queue where K_b is the number of ambulances assigned to base b (i.e. those ambulances either idle at base b or traveling to base b). Specifically, let $K_b(s_k)$ denote the number of available ambulances assigned to base b in state s_k, and define $\phi_b(s_k)$ to be the Erlang loss of the resulting queue

$$\phi_b(s_k) = \frac{\left(\frac{\lambda_b(s_k)}{\mu_b(s_k)}\right)^{K_b(s_k)} / K_b(s_k)!}{\sum_{j=0}^{K_b(s_k)} \left(\frac{\lambda_b(s_k)}{\mu_b(s_k)}\right)^{j} / j!},$$

where $\lambda_b(s_k)$ and $\mu_b(s_k)$ are, respectively, the arrival rate and service rate of emergency calls for base b in state s_k. The arrival rate parameters $\lambda_b(s_k)$ are approximated by assigning demands at locations to their closest base in the following manner. First, we discretize the call arrival process into a grid Γ and define $\Gamma_b = \{\gamma \in \Gamma : d(b,\gamma) = \min_{b' \in 1\ldots B} d(b',\gamma)\}$ where $d(b,\gamma)$ denotes the travel time from base b to the centroid of cell γ. Then we approximate $\lambda_b(s_k)$ as $\sum_{\gamma \in \Gamma_b} \lambda_\gamma(s_k)$ where $\lambda_\gamma(s_k)$ denotes

the arrival rate to cell γ in state s_k. The service rate parameters $\mu_b(s_k)$ are approximated by a constant μ_b which is the average service rate for ambulances stationed at base b and serving calls arriving in Γ_b. Thus the arrival rate for base b in state s_k is approximated by the summed arrival rates from areas closest to base b in state s_k, and the service rate for base b is approximated via a simulation containing only base b and arrivals only in Γ_b. The service rates for each base can be estimated via simulation prior to any ADP computations and included as input to the ADP algorithm.

The Erlang loss is applicable in an ambulance redeployment context because it represents the proportion of emergency call arrivals that occur near a given base when there will be no ambulances stationed at that base. Such calls must be handled by ambulances stationed at bases further away. Due to the additional travel time involved it is usually impossible to respond to these calls within the specified time threshold. Furthermore, the additional travel time causes the responding ambulance to be busy longer than it otherwise would, and hence decreases its availability. Since the additional travel time in this situation increases the cumulative costs directly and indirectly, the Erlang loss serves as a good proxy for the value function in this application.

4.5 Simulation-based Tuning via Regression

The regression-based tuning of ADP policies in this case study is similar to that in (1); however, due to the size of the state space and the inability to compute the required expectation exactly we must use sample path information to tune our parameters. Consequently, we use the following approximate policy iteration algorithm (as contained in Maxwell et al. 2010).

Step 1. Initialize the iteration counter n to 1 and initialize $r^1 = \{r_b^1 : b = 1,\ldots,B\}$ arbitrarily.
Step 2. (Policy improvement) Let μ^n be the greedy policy induced by J_{r^n}, i.e.

$$\mu^n(s_k) \in \underset{x\in\mathscr{X}(s_k)}{\operatorname{argmin}}\left\{\frac{1}{N}\sum_{i=1}^{N}\left(c(s_k,x,u_{k+1}^{(i)}) + J_{r^n}\left(f(s_k,x,u_{k+1}^{(i)})\right)\right)\right\}.$$

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

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

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

This process is repeated until some stopping condition is met. Usually the stopping condition is rather ad hoc such as stopping after a specified amount of CPU time has elapsed or a specified number of iterations have completed. After the stopping condition is met the policy having the best sample performance is usually selected as the ADP policy. An alternative method would be to use ranking and selection upon some or all of the policies explored in the approximate policy iteration algorithm to select the ADP policy.

4.6 Results

We compare the least-squares approximate policy iteration algorithm in Section 4.5 with two other policy tuning approaches: Least-Squares Temporal Difference (LSTD) learning (Boyan 2002) and the Nelder-Mead simplex method (Nelder and Mead 1965). LSTD is based on the popular Temporal Difference (TD) method for ADP policy tuning found in Sutton (1988). The TD learning algorithm

uses a single Monte Carlo sample of the quantity given in (4) (i.e. $N = 1$) in conjunction with the current approximation $J_{r^n}(s_k)$ to update the tunable weights after each state transition. LSTD converges to the same parameters as TD does, but LSTD generally has better convergence properties because it uses entire sample path information to tune parameters as opposed to single-step transitions. The Nelder-Mead simplex method is a black box deterministic function minimization heuristic. Since this method is a black box method it does not use any value function properties of ADP-specific methods. Instead, it uses the discrete-event simulation to obtain performance estimates of the greedy policy with respect to J_r as a function of the tunable weights r. The Nelder-Mead method then tunes r through a derivative-free search method based solely upon policy performance at different values of the tunable weights. We used the implementation of this algorithm given in Nelder and Mead (1965). We use this very simplistic approach to what is certainly a simulation-optimization problem to see what might be possible, rather than as an example of how to tackle such problems. A natural next step would be to employ the recommendations given by Barton and Ivey (1996).

Figure 2 shows the results of these three tuning methods applied to the ambulance redeployment problem on Edmonton. Each "function evaluation" in Figure 2 corresponds to 30 replications of a two week simulation with a given ADP policy. A single function evaluation takes approximately 70 minutes of CPU time. With this large function evaluation time the computation cost of the three methods are dominated by function evaluations, and hence the comparison of these three methods based on function evaluations is justified.

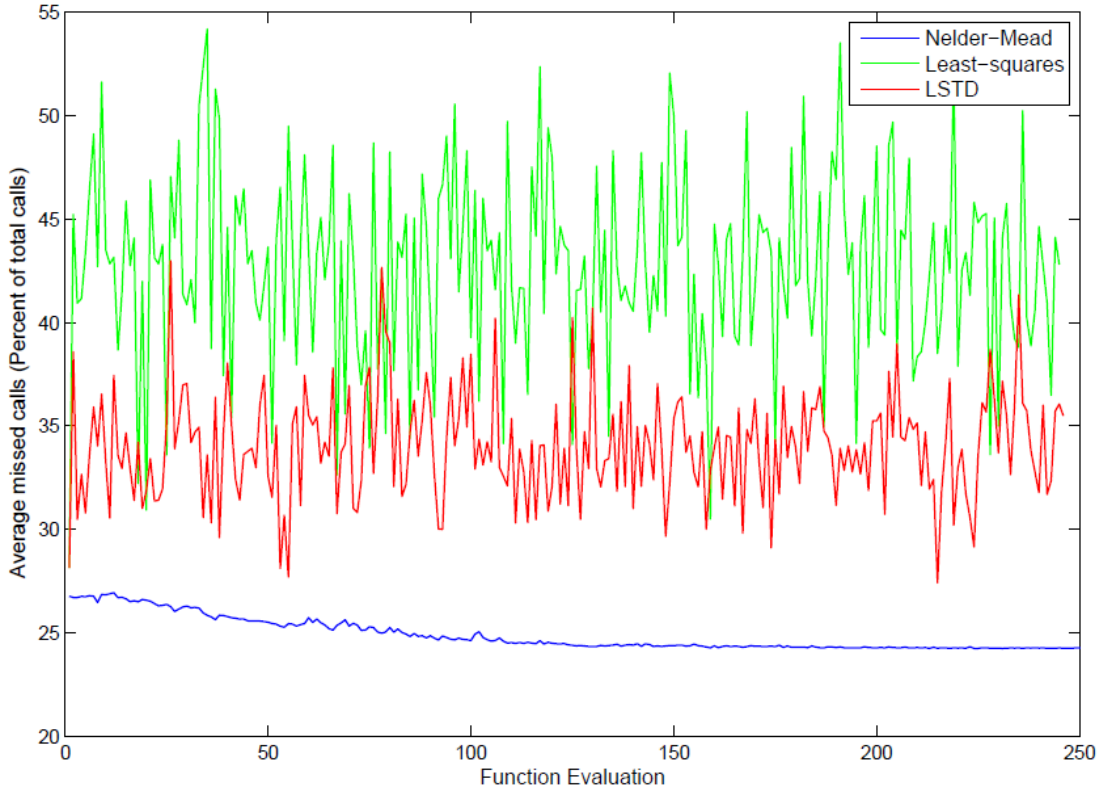

Figure 2: Coefficient Tuning Results

The simulation used synchronized random number streams between the three different tuning algorithms and between all function evaluations for a single algorithm. Since the random number streams are all synchronized a policy's performance is actually deterministic. There are two reasons for this synchronization: first, it allows the results of the three different procedures to be more accurately compared to each other, and second, it illustrates a policy's ability to find coefficients inducing good

performance on a somewhat easier, deterministic version of the problem. This synchronization method has the potential to cause bias in the policy performance (compared to the true performance of a given policy); however, additional simulations with independent random number streams show that this bias is negligible.

The first striking observation in Figure 2 is that the Nelder-Mead method dominates the two regression-based methods throughout the entire training process. The least-squares and LSTD methods vary rapidly between better and worse performing policies with the LSTD method almost always finding better policies than the least-squares methods. Some of the better policies found via the regression methods might be considered adequate policies, but even the best of these policies do not perform as well as those found via the Nelder-Mead method. Furthermore, the least-squares and LSTD methods were unable to find policies with any significant improvement over their initial policies.

A second observation is that the least-squares and LSTD methods appear to be fluctuating around a constant value whereas the Nelder-Mead evaluations appear to be consistently decreasing. This means that the least-squares and LSTD methods are not making any noticeable progress toward better policies as the Nelder-Mead method is. The performance of the policies found via Nelder-Mead flatten after about 150-200 iterations. The Nelder-Mead method may have stopped at a local minimum rather than the global minimum, but even if this is the case, the policies resulting from the local minimum are superior to those found via the regression-based methods.

5 CONCLUSION

We considered two types of tuning methods for ADP policies: regression-based and direct search. The standard ADP policy tuning methods use regression to select weighting coefficients r that result in an approximating value function J_r which is close to samples of the true value function J. The direct search method tunes the parameters directly to improve performance. In Section 3 we gave an example MDP where regression-based tuning methods are theoretically unable to achieve an optimal policy even though there exists a set of tuning weights for which the greedy policy with respect to these weights is optimal. In Section 4 we empirically showed a direct search method obtaining superior performing policies than two standard regression-based approaches for a realistic ambulance redeployment problem.

Although simulation-optimization methods have had some success in ADP tuning applications previously, they are often viewed as naïve and computationally expensive. It is true that these methods are not tailored to ADP applications specifically and that the computational effort for direct search methods may be high, but we have shown, both theoretically and empirically, that these methods may be able to achieve a higher level of performance from ADP approximation architectures than traditional methods are fundamentally able to achieve. Consequently, we expect great benefits from further research on simulation-optimization methods tailored for ADP policy tuning.

ACKNOWLEDGMENTS

This research was supported in part by NSF Grant Number CMMI 0758441.

REFERENCES

Barton, R. R., and J. Ivey, John S.. 1996. Nelder-Mead Simplex Modifications for Simulation Optimization. *Management Science* 42 (7): 954–973.

Bertsekas, D. 2005. *Dynamic programming and optimal control*. Nashua, NH: Athena Scientific.

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.

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

Erkut, E., A. Ingolfsson, and G. Erdoğan. 2007. Ambulance deployment for maximum survival. *Naval Research Logistics* 55 (1): 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* 27:1641–1653.

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

Maxwell, M. S., M. Restrepo, S. G. Henderson, and H. Topaloglu. 2010. Approximate Dynamic Programming for Ambulance Redeployment. *INFORMS Journal on Computing* 22 (2): 266–281.

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

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

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

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

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

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.S. in Computer Science from Brigham Young University and is a recipient of the U.S. Department of Homeland Security's Graduate Fellowship. His research interests include discrete-event simulation, simulation optimization, and approximate dynamic programming. His web page is located at <http://people.orie.cornell.edu/msm57/>.

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 is 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. His web page is located at <http://legacy.orie.cornell.edu/~huseyin/>.