An Approximate Dynamic Programming Approach to Decentralized Control of Stochastic Systems

Randy Cogill[1], Michael Rotkowitz[2], Benjamin Van Roy[3], and Sanjay Lall[4]

[1] Department of Electrical Engineering, Stanford University.
`rcogill@stanford.edu`
[2] Department of Aeronautics and Astronautics, Stanford University.
`rotkowitz@stanford.edu`
[3] Departments of Management Science and Engineering and Electrical Engineering, Stanford University.
`bvr@stanford.edu`
[4] Department of Aeronautics and Astronautics, Stanford University.
`lall@stanford.edu`

Summary. We consider the problem of computing decentralized control policies for stochastic systems with finite state and action spaces. Synthesis of optimal decentralized policies for such problems is known to be NP-hard [1]. Here we focus on methods for efficiently computing meaningful suboptimal decentralized control policies. The algorithms we present here are based on approximation of optimal Q-functions. We show that the performance loss associated with choosing decentralized policies with respect to an approximate Q-function is related to the approximation error.

1 Introduction

The study of decentralized control is motivated by the fact that, in many practical control applications, control decisions must be made using incomplete information about the system state. In particular, it is common that multiple interacting decision makers must make control decisions at each time period, and each decision must be made using a different limited observation of the overall system state. The complicating factor in such problems is that the impact of individual decisions is system-wide.

We consider the problem of determining decentralized control policies for stochastic systems. Decentralized control problems have been well studied over the past several decades [2], and it is generally recognized that decentralized problems are often considerably more complex than their centralized counterparts. For example, it has been shown in [1] that computing optimal policies for a simple single stage control problem is NP-hard, whereas the corresponding centralized problem is trivial. This single-stage problem arises as a special case of the problems considered here. With this in mind, our focus is on efficient computation of meaningful suboptimal decentralized control policies.

In stochastic control problems, the relevant information required for decision making can be captured by a *Q-function.* When a Q-function has special structure, we can easily compute a decentralized policy which is greedy with respect to this function. The approach taken here can be interpreted as approximating the optimal Q-function for a problem by one with this special structure. We show that the performance loss associated with choosing policies with respect to an approximate Q-function is related to the approximation error.

A related approach for computing suboptimal policies for specially structured multi-agent control problems can be found in [3]. The problems considered there have transition probabilities modeled as a dynamic Bayesian network and structured single-stage costs. The authors show that by using specially structured approximations to the cost-to-go function, the required computations can be simplified.

The approach discussed here resembles a method commonly used to synthesize decentralized control policies for linear time-invariant systems [4]. Semidefinite programming conditions can be formulated for these problems which produce stabilizing decentralized control laws when they are satisfied. These conditions involve a simultaneous search for a stabilizing control law and a Lyapunov function proving stability. The Lyapunov functions used are restricted to have special structure, where the structure is chosen to facilitate the search for a decentralized control law.

2 Dynamic Programming Background

Here we consider discrete-time stochastic control problems. The systems considered here have a finite state space $\mathcal{X}$, and a finite set $\mathcal{U}$ of actions available at each time step. Taking action $u \in \mathcal{U}$ when in state $x \in \mathcal{X}$ incurs a cost $g(x, u)$. After taking action u in state x, the system state in the next time period is $y \in \mathcal{X}$ with probability $p(y \mid x, u)$.

The goal is to find a rule for choosing actions which minimizes some measure of the overall cost incurred. A rule for choosing actions is commonly referred to as a *policy.* A *static state-feedback* policy $\mu : \mathcal{X} \to \mathcal{U}$ is a rule which chooses an action based on the current system state.

Here we consider the problem of choosing a policy to minimize the *expected total discounted cost*:

$$J^{\mu}(x) = \mathbb{E}\left[\sum_{t=0}^{\infty} \alpha^{t} g(x_t, \mu(x_t)) \,\middle|\, x_0 = x \right],$$

where $0 \leq \alpha < 1$. For any policy μ, we can compute the *cost-to-go* J^μ by solving the linear equations

$$J^\mu(x) = g(x, \mu(x)) + \alpha \sum_{y \in \mathcal{X}} p(y|x, \mu(x)) J^\mu(y). \tag{1}$$

We can write these equations in matrix-vector notation as $J^\mu = g_\mu + \alpha P_\mu^T J^\mu$. Here, g_μ is the cost vector evaluated for the policy μ, P_μ is the state transition matrix associated with the policy μ, and P_μ^T is its transpose. We can further simplify this equation using the shorthand $J^\mu = T_\mu J^\mu$, where the operator T_μ maps a vector J to the vector $g_\mu + \alpha P_\mu^T J$. This notation will be convenient in later sections.

For the expected total discounted cost criterion, there is a single policy which minimizes the cost-to-go for all initial states. Also, the minimum cost-to-go J^* is unique and satisfies *Bellman's equation*

$$J^*(x) = \min_{u \in \mathcal{U}} \left(g(x, u) + \alpha \sum_{y \in \mathcal{X}} p(y|x, u) J^*(y) \right). \tag{2}$$

We can simplify notation by writing this equation as $J^* = TJ^*$, where the operator T maps a vector J to the vector TJ defined as

$$(TJ)(x) = \min_{u \in \mathcal{U}} \left(g(x, u) + \alpha \sum_{y \in \mathcal{X}} p(y|x, u) J(y) \right).$$

It is often useful to work directly with the expression in the right hand side of Bellman's equation. This is often referred to as a *Q-function*, and the optimal Q-function Q^* is given by

$$Q^*(x, u) = g(x, u) + \alpha \sum_{y \in \mathcal{X}} p(y|x, u) J^*(y).$$

The optimal policy μ^* is obtained from Q^* by taking $\mu^*(x) = \text{argmin}_u Q^*(x, u)$. Like cost-to-go functions, we can compute the Q-function associated with a particular policy μ by solving the linear equations

$$Q^\mu(x, u) = g(x, u) + \alpha \sum_{y \in \mathcal{X}} p(y|x, u) Q^\mu(y, \mu(y)).$$

We can introduce an operator F_μ and write the equations above simply as $Q^\mu = F_\mu Q^\mu$. Also, Bellman's equation can be expressed directly in terms of Q functions as

$$Q^*(x, u) = g(x, u) + \alpha \sum_{y \in \mathcal{X}} p(y|x, u) \left(\min_w Q^*(y, w) \right), \tag{3}$$

with Q^* being the unique solution. As before, we can express this equation in the notation $Q^* = FQ^*$, where the operator F maps a vector Q into a vector FQ satisfying

$$(FQ)(x,u) = g(x,u) + \alpha \sum_{y \in \mathcal{X}} p(y|x,u) \left(\min_{w} Q(y,w) \right)$$

Q-functions will play an important role in the upcoming material.

We will conclude this section by mentioning that the operators T_μ, T, F_μ, and F have some important properties [5]. The following properties, presented only in terms of F for convenience, hold for T_μ, T, F_μ, and F:

- *Contractiveness:* For any Q_1 and Q_2, $\|FQ_1 - FQ_2\|_\infty \leq \alpha \|Q_1 - Q_2\|_\infty$. As a result of the contraction mapping theorem, F has a unique fixed point Q^* which is obtained as the limit of the sequence $\{F^n Q\}$ for any Q.
- *Monotonicity:* For any Q_1 and Q_2, if $Q_1 \leq Q_2$, then $FQ_1 \leq FQ_2$.

These properties will be used in the proofs of some upcoming theorems.

3 Computing Decentralized Policies

The problem of computing decentralized policies is considerably more complex than the corresponding problem of computing centralized policies. For example, we can consider the simple special case where $\alpha = 0$. In this case, the optimal centralized policy is simply $\mu^*(x) = \operatorname{argmin}_u \, g(x,u)$. On the other hand, we can consider a simple decentralized variant of this problem where the state is described by two state variables x_1 and x_2 and each action is described by two decision variables u_1 and u_2. The desired decentralized policy μ chooses u_1 based only on x_1 and chooses u_2 based only on x_2. Unlike the centralized problem, there may not be one decentralized policy which achieves lower cost than all other decentralized policies at all states. Therefore, the problem of minimizing expected cost may not be well defined until we specify an initial probability distribution on the states. Once an initial distribution is specified, the resulting optimization problem is NP-hard [1]. This example highlights the importance of efficient algorithms for computing meaningful *suboptimal* policies for decentralized control problems.

In the systems we consider, the action taken at each time period can be described by a collection of decision variables as $u = (u_1, \dots, u_k)$. The action space for this system is $\mathcal{U} = U_1 \times \cdots \times U_k$. Likewise, the state of the system is described by a collection of state variables and the corresponding state space is $\mathcal{X} = X_1 \times \cdots \times X_p$. In general, the policies produced by dynamic programming determine each action u_i based on the entire state $x = (x_1, \dots, x_p)$. However, we are interested in computing policies where each decision is only based on a subset of the state variables. A *decentralized* policy is a policy in which each action u_i depends only on some specified subset of the state variables $x_1, \dots, x_p$.

The required dependencies of actions on state variables is typically referred to as the *information structure* of the desired policy. We will use the following notation to describe the information structure of a particular policy. The set of variables which may be used to decide action i is denoted as

$$\mathcal{I}_i = \{j \mid u_i \text{ depends on } x_j\}.$$

In particular, $\mathcal{I}_i$ gives the indices of the state variables that action i depends on. The Cartesian product of the spaces of the variables used to decide action i is denoted as

$$\mathcal{X}_i = \bigtimes_{j \in \mathcal{I}_i} X_j.$$

Using this notation, a decentralized policy is a set of functions $\mu_i : \mathcal{X}_i \to U_i$ for $i = 1, \ldots, k$.

Recall that the optimal centralized policy is $\mu^*(x) = \operatorname{argmin}_u Q^*(x,u)$, where Q^* is obtained as the unique solution to Bellman's equation. Consider a policy $\mu(x) = \operatorname{argmin}_u \widehat{Q}(x,u)$, where $\widehat{Q}$ is a function of the form $\widehat{Q} = \sum_{i=1}^k Q_i$ and each $Q_i : \mathcal{X}_i \times U_i \to \mathbb{R}$. Since u_i is the only decision variable appearing in the argument of Q_i, choosing u to be $\operatorname{argmin}_u \widehat{Q}(x,u)$ is equivalent to choosing each u_i to be $\operatorname{argmin}_{u_i} Q_i(x,u_i)$. Also, since the only state variables appearing in the argument of Q_i are those in $\mathcal{X}_i$, the choice of u_i only depends on the values of the variables in $\mathcal{X}_i$. Therefore, the policy μ is decentralized since each decision u_i is determined only in terms of the state variables in $\mathcal{X}_i$. This suggests an algorithm, in which an appropriately structured $\widehat{Q}$ is chosen to approximate Q^*, and a decentralized policy is obtained from $\widehat{Q}$. Such an algorithm will be presented in the next section. Theorem 1 justifies this approach by showing that the suboptimality of the performance achieved by μ is related to the approximation error between $\widehat{Q}$ and Q^*.

Before presenting Theorem 1, we will introduce some notation. Let Q_μ denote the function Q evaluated for the policy μ. That is, $Q_\mu(x) = Q(x,\mu(x))$. We must be careful to make the distinction between $Q_\mu : \mathcal{X} \to \mathbb{R}$, the function Q evaluated for the policy μ, and $Q^\mu : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$, the Q-function associated with following the policy μ from time $t = 1$ onward. Also, let $\|\cdot\|_{1,\nu}$ denote the weighted 1-norm, defined as

$$\|J\|_{1,\nu} = \sum_x \nu(x)|J(x)|,$$

where the weights ν are positive. For the upcoming theorem, we will restrict ourselves, without loss of generality, to weights which satisfy $\sum_x \nu(x) = 1$.

Theorem 1. *Suppose Q satisfies $Q \leq FQ$. If μ is the policy defined as $\mu(x) = \operatorname{argmin}_u Q(x,u)$, then the following bound holds*

$$\|J^\mu - J^*\|_{1,\nu} \leq \frac{1}{1-\alpha}\|Q^*_\mu - Q_\mu\|_{1,\omega} \ ,$$

where ν is an arbitrary probability distribution on $\mathcal{X}$ and ω is the probability distribution defined as $\omega = (1-\alpha)(I-\alpha P_\mu)^{-1}\nu$.

Proof. First we will show that ω is a probability distribution. Let e denote the vector with 1 for each entry. Since since $e^T\nu = 1$ and $e^T P_\mu^t = e^T$ for all t,

$$\begin{aligned} e^T\omega &= (1-\alpha)e^T(I-\alpha P_\mu)^{-1}\nu \\ &= (1-\alpha)\sum_{t=0}^{\infty}\alpha^t e^T P_\mu^t \nu \\ &= (1-\alpha)\sum_{t=0}^{\infty}\alpha^t e^T \nu \\ &= 1 \end{aligned}$$

Also, since $P_\mu \geq 0$ and $\nu \geq 0$ componentwise, $\omega \geq 0$.

Recall that we require $Q \leq FQ$. By monotonicity of F, this implies $Q \leq F^n Q$ for all n. Since $F^n Q \to Q^*$ for any Q, this implies $Q \leq Q^*$. Also, the policy μ is greedy with respect to Q, so $Q_\mu \leq J^* \leq J^\mu$. Therefore,

$$\begin{aligned} \|J^\mu - J^*\|_{1,\nu} &\leq \|J^\mu - Q_\mu\|_{1,\nu} \\ &= \nu^T((I-\alpha P_\mu^T)^{-1}g_\mu - Q_\mu) \\ &= \nu^T(I-\alpha P_\mu^T)^{-1}(g_\mu - (I-\alpha P_\mu^T)Q_\mu) \\ &= \frac{1}{1-\alpha}\omega^T((g_\mu + \alpha P_\mu^T Q_\mu) - Q_\mu) \end{aligned}$$

Since μ is greedy with respect to Q, we have $(FQ)_\mu = g_\mu + \alpha P_\mu^T Q_\mu$. Therefore, the inequality above can be expressed as

$$\|J^\mu - J^*\|_{1,\nu} \leq \frac{1}{1-\alpha}\|(FQ)_\mu - Q_\mu\|_{1,\omega}$$

Since $Q_\mu \leq (FQ)_\mu \leq Q_\mu^*$, we have the result

$$\|J^\mu - J^*\|_{1,\nu} \leq \frac{1}{1-\alpha}\|Q_\mu^* - Q_\mu\|_{1,\omega}$$

■

A theorem similar to Theorem 1, based only on cost-to-go functions, can be found in [6].

4 The Linear Programming Approach

It is well known that the solution to Bellman's equation J^* can be obtained by solving a linear program [7]. We can determine $Q^*(x,u) = g(x,u) +$

$\alpha \sum_{y\in\mathcal{X}} p(y|x,u)J^*(y)$ once J^* is known. In the previous section, we outlined a method for computing a decentralized policy based on an approximation of Q^*. In this section, we present an algorithm for computing such an approximation. This algorithm is based on an extension of the linear programming approach to solving Bellman's equation. The standard linear programming formulation is extended to explicitly include Q. The need for including Q will become clear shortly.

Theorem 2. *The solutions to the equations (2) and (3) can be obtained by solving the linear program*

$$\begin{aligned} &\textit{maximize: } \textstyle\sum_{x\in\mathcal{X}}\sum_{u\in\mathcal{U}} Q(x,u) \\ &\textit{subject to: } Q(x,u) \le g(x,u) + \alpha \textstyle\sum_{y\in\mathcal{X}} p(y|x,u)J(y) && \textit{for all } x\in\mathcal{X}, u\in\mathcal{U} \\ &\qquad\qquad\quad J(x) \le Q(x,u) && \textit{for all } x\in\mathcal{X}, u\in\mathcal{U} \end{aligned}$$

Proof. Q^* satisfies

$$Q^*(x,u) = g(x,u) + \alpha \sum_{y\in\mathcal{X}} p(y|x,u)\left(\min_w Q^*(y,w)\right).$$

Also, $J^*(x) = \min_u Q^*(x,u) \le Q^*(x,u)$ for all x, u. Therefore, Q^* and J^* are feasible for this LP.
Also, since $J(x) \le Q(x,u)$ for any feasible J, any feasible Q satisfies

$$\begin{aligned} Q(x,u) &\le g(x,u) + \alpha \sum_{y\in\mathcal{X}} p(y|x,u)J(y) \\ &\le g(x,u) + \alpha \sum_{y\in\mathcal{X}} p(y|x,u)\left(\min_w Q(y,w)\right), \end{aligned}$$

or more succinctly $Q \le FQ$. By monotonicity of F, this implies $Q \le F^nQ$ for all n. Since $F^nQ \to Q^*$ for any Q, any feasible Q satisfies $Q \le Q^*$. Therefore Q^* is the optimal feasible solution. ■

Note that the usual LP approach to solving Bellman's equation only involves J, and therefore contains fewer variables and constraints. The explicit inclusion of Q in the linear program is required because we will require Q to have special structure in order to obtain a decentralized policy. In general, requiring J to have some special structure is not sufficient to guarantee that Q will have the required structure.

Using this linear program, we have the following algorithm for computing a decentralized policy.

Algorithm 1.

1. *For any information structure, let* $\widehat{Q} = \sum_{i=1}^{k} Q_i$ *and* $\widehat{J} = \sum_{i=1}^{k} J_i$*, where* $Q_i : \mathcal{X}_i \times U_i \to \mathbb{R}$ *and* $J_i : \mathcal{X}_i \to \mathbb{R}$.
2. *For arbitrary positive values* $\omega(x)$*, solve the linear program*

$$\begin{aligned} &\textit{maximize: } \textstyle\sum_{x\in\mathcal{X}}\sum_{u\in\mathcal{U}} \omega(x)\widehat{Q}(x,u) \\ &\textit{subject to: } \widehat{Q}(x,u) \le g(x,u) + \alpha \textstyle\sum_{y\in\mathcal{X}} p(y|x,u)\widehat{J}(y) \quad \textit{for all } x \in \mathcal{X}, u \in \mathcal{U} \\ &\qquad\qquad\quad \widehat{J}(x) \le \widehat{Q}(x,u) \qquad\qquad\qquad\qquad\qquad \textit{for all } x \in \mathcal{X}, u \in \mathcal{U}, \end{aligned}$$

3. *Let* $\mu(x) = \mathit{argmin}_u \widehat{Q}(x,u)$*. This policy is decentralized with the desired information structure.*

It is not immediately clear how the optimization problem solved in Algorithm 1 relates to the cost-to-go achieved by μ. We will show that this optimization problem is equivalent to minimizing a weighted norm of the error between Q^* and $\widehat{Q}$ subject to some constraints. From Theorem 1, this error is directly related to the performance achieved by μ.

Theorem 3. *The linear program in Algorithm 1 is equivalent to*

$$\begin{aligned} &\textit{minimize: } \|Q^* - \widehat{Q}\|_{1,\Omega} \\ &\textit{subject to: } \widehat{Q} \le F\widehat{Q}, \end{aligned}$$

where $\Omega(x,u) = \omega(x)$ *for all* $x \in \mathcal{X}$ *and* $u \in \mathcal{U}$.

Proof. From the proof of Theorem 2, $\widehat{Q}$ is feasible if and only if $\widehat{Q} \le F\widehat{Q}$, which implies $\widehat{Q} \le Q^*$. Therefore, for any feasible $\widehat{Q}$

$$\begin{aligned} \|\widehat{Q} - Q^*\|_{1,\Omega} &= \sum_{x\in\mathcal{X}}\sum_{u\in\mathcal{U}} \omega(x)(Q^*(x,u) - \widehat{Q}(x,u)) \\ &= -\sum_{x\in\mathcal{X}}\sum_{u\in\mathcal{U}} \omega(x)\widehat{Q}(x,u) + \sum_{x\in\mathcal{X}}\sum_{u\in\mathcal{U}} \omega(x)Q^*(x,u). \end{aligned}$$

Since $\sum_{x\in\mathcal{X}}\sum_{u\in\mathcal{U}} \omega(x)Q^*(x,u)$ is constant, minimizing this quantity is equivalent to maximizing $\sum_{x\in\mathcal{X}}\sum_{u\in\mathcal{U}} \omega(x)\widehat{Q}(x,u)$, the objective in the original LP. ■

From the previous theorem, we see that Algorithm 2 seeks to minimize the approximation error $\|\widehat{Q} - Q^*\|_{1,\Omega}$ subject to the constraint $\widehat{Q} \le F\widehat{Q}$. For any policy $\hat{\mu}$ and any probability distribution ω, the bound $\|\widehat{Q}_{\hat{\mu}} - Q^*_{\hat{\mu}}\|_{1,\omega} \le \|\widehat{Q} - Q^*\|_{1,\Omega}$ holds. This, together with Theorem 1, assures us that Algorithm

1 attempts to produce a decentralized policy $\widehat{\mu}$ with performance close to that of an optimal centralized policy.

When no structure is imposed on $\widehat{Q}$ (i.e., the centralized case), the weights ω in the objective function are unimportant since we obtain Q^* for any positive weights. However, when $\widehat{Q}$ has structure, the weights chosen will affect the quality of the policy obtained. A good choice of weights may be guided by the interpretation of these weights in Theorem 1. In particular, these weights impose a tradeoff in the quality of the approximation across the states. Ideally, we would like to closely approximate states that are visited most often under any policy. The role of these weights are discussed in greater detail in [6].

For highly decentralized information structures, the number of variables in the linear program increases *linearly* with the number of state variables. Therefore, the number of variables appearing in this linear program is often significantly less than the number of variables required for computation Q^*. However, the number of constraints appearing in the linear program is still on the order of the number of state-action pairs. The application of Algorithm 1 to large-scale problems may still be prohibitive due to the large number of constraints. Similar problems are encountered in the application of the approximate linear programming methods of [6] to large problems, however constraint sampling has been shown to be an effective method for overcoming this problem [8]. We believe that the analysis and application of constraint sampling as in [8] will carry over to the problems here, although we have not explored the details at this point.

5 Example

Here we will apply the algorithm developed in Section 4 to an example problem. The problem considered is the following. Two robots are placed in opposite corners of a region, and calls for services to be performed by the robots originate from various points within the region. This is depicted in Figure 1. Calls for service are sent to both robots. Each robot, if idle, must make a decision of whether or not to service the current location. Each robot must make this decision without knowledge of the other robot's status (idle or busy), resulting in a decentralized decision making problem.

For this problem, it is not clear which strategy the robots should follow. If both robots, when idle, respond to all calls for service, then we will often have both robots responding to calls when only one is required to service the location. On the other hand, if we partition the set of locations and assign each location to a single robot, then locations will often go unserviced when their assigned robot is busy at another location. We will see that both of these strategies are typically outperformed by strategies which assign a single robot to some locations, and both robots to others.

In our model, events occur at discrete time periods $t = 0, 1, \ldots$. The system state has three components $x = (x_1, x_2, x_3)$, where the compo-

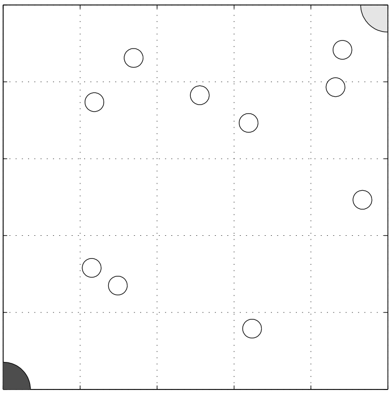
Fig. 1. Robots are stationed in the lower left and upper right corners, and respond to calls for service at the locations shown by circles.

nents are described as follows. At time t, robot i is in state $x_i(t) \in \mathcal{X}_i = \{\texttt{robot i idle},\ \texttt{robot i busy}\}$, depending on whether or not it is idle or currently serving a location. For a problem with N locations, the call for service at time t is

$$x_3(t) \in \mathcal{X}_3 = \{\texttt{location 1}, \ldots,\ \texttt{location N},\ \texttt{no service requested}\}.$$

We make the restriction that only one call for service may be active at a time. At time t, if robot i is idle and there is an active call for service, robot i chooses an action $u_i(t) \in \mathcal{U}_i = \{\texttt{respond},\ \texttt{don't respond}\}$. The overall state space for this system is $\mathcal{X} = X_1 \times X_2 \times X_3$. The overall action space is $\mathcal{U} = U_1 \times U_2$.

At each time period, calls for service are independent and identically distributed. Therefore, if there is no response to a call, it does not persist and is replaced by a new call (or no call) in the following time period. Let $p_3(y_3)$ give the probability of calls for service in each location. If a robot is busy servicing a location in any time period, it remains busy in the following period with probability 0.9. Let $p_i(y_i|x_i, u_i)$ give the probability that robot i is busy or idle in the following time period given its current state and action. The dynamics of this system are described by the transition probability function $p(y|x, u) = p_1(y_1|x_1, u_1)p_2(y_2|x_2, u_2)p_3(y_3)$.

The control objective is expressed through costs that are incurred in each time period. Costs incurred for servicing a location are based on the distance required to reach the location and the time required to complete servicing. Upon responding to a location, a cost of 2($\texttt{distance to location}$) is incurred by each robot which responds (*i.e.*, the cost of getting there and back). For our example, the region considered is the unit square, and the distance used is the Euclidean distance from the the robot's corner to the call location. For each time period that a robot is busy, it incurs an additional cost of 0.25 units. These servicing costs for each robot are given by a function $g_i(x_i, u_i)$.

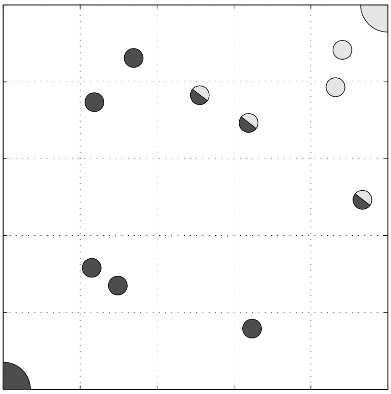

Fig. 2. The decentralized policy computed in Example 1. Lighter shaded circles are served only by robot 1, darker shaded circles are served only by robot 2, and circles with both shades are served by both robots.

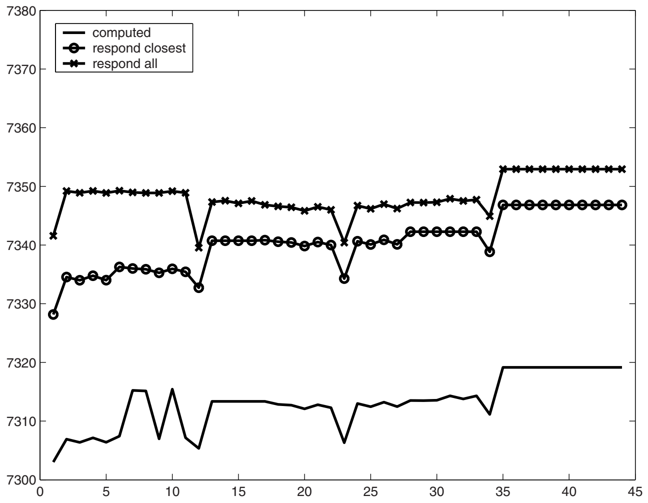

Fig. 3. The total discounted cost as a function of state for the policies used in Example 1.

Additionally, if neither robot responds to a call, a large penalty is incurred. In this case, we use a penalty of 8 units. These penalties are given by a function $g_3(x_1, x_2, u_1, u_2)$. The overall per-period costs for this problem are given by $g(x, u) = g_1(x_1, u_1) + g_2(x_2, u_2) + g_3(x_1, x_2, u_1, u_2)$. Our goal is to determine control policies $\mu_1 : X_1 \times X_3 \to U_1$ and $\mu_2 : X_2 \times X_3 \to U_2$ which make the total discounted cost as small as possible (with $\alpha = 0.999$).

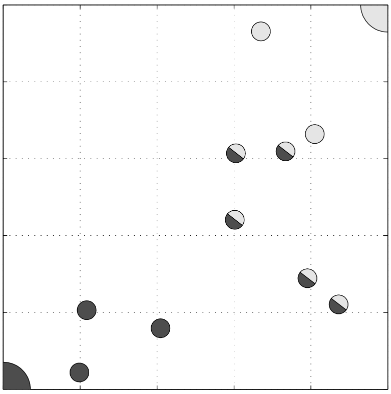

Fig. 4. The decentralized policy computed in Example 2. Lighter shaded circles are served only by robot 1, darker shaded circles are served only by robot 2, and circles with both shades are served by both robots.

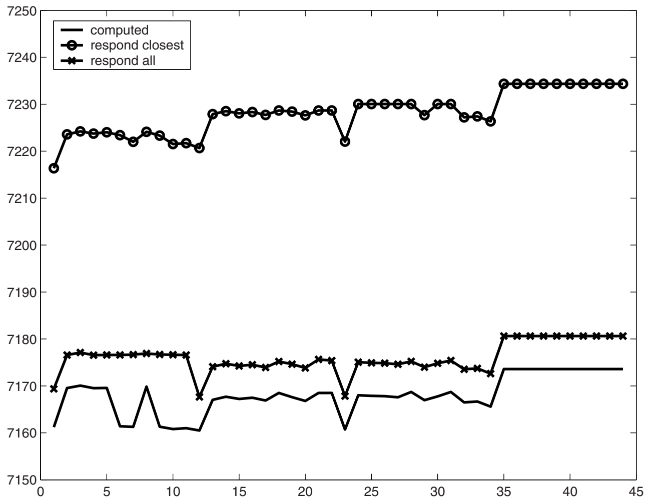

Fig. 5. The total discounted cost as a function of state for the policies used in Example 2.

We applied Algorithm 1 to several instances of this problem with $N = 10$. The results of the first example are shown in Figures 2 and 3. In this example, a decentralized policy is computed and compared against two heuristics, ‘respond closest’ and ‘respond all’. In the ‘respond closest’ policy, robots only respond to the locations closest to them. In the ‘respond all’ policy, each robot responds to every location. The results of the second example are shown in Figures 4 and 5. It is interesting to note the difference between the performance of the heuristic policies in the two examples. In particular, relative performance of the two heuristic policies depends on the particular problem instance. The computed policies outperform both heuristics in both examples.

6 Conclusions

Here we considered the problem of designing decentralized control policies for stochastic systems. An algorithm based on linear programming was presented. This algorithm obtains a decentralized policy from a function with special structure that approximates the optimal centralized Q-function. The performance loss associated with the resulting decentralized policy was shown to be related to the approximation error.

References

1. J. Tsitsiklis and M. Athans, "On the complexity of decentralized decision making and detection problems," *IEEE Trans. Automatic Control*, vol. 30, no. 5, pp. 440–446, 1985.
2. N. Sandell, P. Varayia, M. Athans, and M. Safonov, "Survey of decentralized control methods for large-scale systems," *IEEE Trans. Automatic Control*, vol. 26, pp. 108–128, 1978.
3. C. Guestrin, D. Koller, and R. Parr, "Multiagent planning with factored MDPs," *Advances in Neural Information Processing Systems*, vol. 14, pp. 1523–1530, 2001.
4. J. Bernussou, P. Peres, and J. Geromel, "Robust decentralized regulation: a linear programming approach," *Proc. IFAC Symposium on Large Scale Systems*, vol. 1, pp. 133–136, 1989.
5. D. Bertsekas and J. Tsitsiklis, *Neuro-dynamic Programming.* Athena Scientific, 1996.
6. D. de Farias and B. V. Roy, "The linear programming approach to approximate dynamic programming," *Operations Research*, vol. 51, no. 6, pp. 850–865, 2003.
7. A. Manne, "Linear programming and sequential decisions," *Management Science*, vol. 6, no. 3, pp. 259–267, 1960.
8. D. de Farias and B. V. Roy, "On constraint sampling in the linear programming approach to approximate dynamic programming," *to appear in Mathematics of Operations Research*, submitted 2001.
9. M. Aicardi, F. Davoli, and R. Minciardi, "Decentralized optimal control of Markov chains with a common past information set," *IEEE Trans. Automatic Control*, vol. 32, no. 11, pp. 1028–1031, 1987.
10. R. Cogill, M. Rotkowitz, B. V. Roy, and S. Lall, "An approximate dynamic programming approach to decentralized control of stochastic systems," *Proceedings of the 2004 Allerton Conference on Communication, Control, and Computing*, pp. 1040–1049, 2004.
11. Y. Ho, "Team decision theory and information structures," *Proceedings of the IEEE*, vol. 68, no. 6, pp. 644–654, 1980.

12. K. Hsu and S. Marcus, “Decentralized control of finite state Markov processes,” *IEEE Trans. Automatic Control*, vol. 27, no. 2, pp. 426–431, 1982.
13. C. Papadimitriou and J. Tsitsiklis, “Intractable problems in control theory,” *SIAM Journal on Control and Optimization*, vol. 24, no. 4, pp. 639–654, 1986.
14. ——, “The complexity of Markov decision processes,” *Mathematics of operations research*, vol. 12, no. 3, pp. 441–450, 1987.
15. M. Puterman, *Markov decision processes.* John Wiley and Sons, New York, 1994.
16. R. Srikant and T. Başar, “Optimal solutions in weakly coupled multiple decision maker Markov chains with nonclassical information,” *Proceedings of the IEEE Conference on Decision and Control*, pp. 168–173, 1989.