Solving POMDPs Using Quadratically Constrained Linear Programs

Christopher Amato
camato@cs.umass.edu

Daniel S. Bernstein
bern@cs.umass.edu

Shlomo Zilberstein
shlomo@cs.umass.edu

Department of Computer Science
University of Massachusetts
Amherst, MA 01003

ABSTRACT

Developing scalable algorithms for solving partially observable Markov decision processes (POMDPs) is an important challenge. One promising approach is based on representing POMDP policies as finite-state controllers. This method has been used successfully to address the intractable memory requirements of POMDP algorithms. We illustrate some fundamental theoretical limitations of existing techniques that use controllers. We then propose a new approach that formulates the problem as a quadratically constrained linear program (QCLP), the solution of which provides an optimal controller of a desired size. We evaluate several optimization methods for solving QCLPs and compare their performance with existing POMDP optimization methods. While the optimization algorithms used in this paper only guarantee locally optimal solutions, the results show consistent improvement of solution quality over the state-of-the-art techniques. The results show that powerful nonlinear programming algorithms can be used effectively to improve the performance and scalability of POMDP algorithms.

1. INTRODUCTION

Since the early 1990's, Markov decision processes (MDPs) and their partially observable counterparts (POMDPs) have been widely used by the AI community for planning under uncertainty. POMDPs offer a rich language to describe situations involving uncertainty about the domain, stochastic actions, noisy observations, and a variety of possible objective functions. POMDP applications include robot control [16], medical diagnosis [12] and machine maintenance [7]. Robots typically have sensors that provide uncertain and incomplete information about the state of the environment, which must be factored into the planning process. In a medical setting, the internal state of the patient is often not known with certainty. The machine maintenance problem, one of the earliest applications areas of POMDPs, seeks to find a cost-effective strategy for inspection and replacement of parts in a domain where partial information about the internal state is obtained by inspecting products. Numerous other applications of POMDPs are surveyed in [3].

Developing effective algorithms for MDPs and POMDPs has become a thriving AI research area [4, 8, 11, 13, 14, 15]. Thanks to these new algorithms and improvements in computing power, it is now possible to solve very large and realistic MDPs. In contrast, current POMDP exact techniques are limited by high memory requirements to toy problems. POMDP exact and approximate solution techniques cannot usually find near-optimal solutions with a limited amount of memory. Even though an optimal solution may be concise, current exact algorithms that use dynamic programming often require an intractable amount of space. While work has been done to make this process more efficient [8], the time and space complexity of POMDP algorithms remains a serious challenge. POMDP approximation algorithms can operate with a limited amount of memory, but as a consequence they provide very weak theoretical guarantees. In contrast, we describe a new approach that addresses the space requirement of POMDP algorithms while maintaining well-defined optimality guarantees.

Current techniques to find optimal fixed-size controllers rely solely on local information [14, 15] thus can give solutions that are unboundedly below the global optimum. We present a formulation that defines the optimal fixed-size controller, and we employ standard nonlinearly constrained optimization techniques in order to find POMDP policies using this formulation. Nonlinearly constrained optimization is an active field of research that has produced a wide range of techniques that can quickly solve a variety of large problems [2]. The quadratic constraints and linear objective function of our formulation belong to a special class of optimization problems for which many robust and efficient algorithms have been developed. While these techniques only guarantee locally optimal solutions, the new formulation facilitates a more efficient search of the solution space and produces high-quality results. Moreover, the algorithms employ more advanced techniques than those previously used [14, 15] to avoid convergence at certain suboptimal points.

The rest of the paper is organized as follows. We first give an overview of the POMDP model and explain how a solution policy can be represented as a stochastic controller. We then show how to produce the optimal controller using a quadratically constrained linear program (QCLP). The optimal solution of the QCLP provides an optimal controller of a desired size. We conclude by demonstrating for a set of large POMDPs that our formulation permits higher valued fixed-size controllers to be found than those generated by previous approaches while maintaining similar running times. This suggests that by using our QCLP, small, high-valued controllers can be efficiently found for a large assortment of POMDPs.

2. BACKGROUND

In this section, we describe the POMDP model and discuss earlier work on fixed-size solutions to POMPDs. We also describe the representation of a POMDP policy as a stochastic finite-state controller.

2.1 POMDP framework

A POMDP can be defined with the following tuple: $M = \langle S, A, P, R, \Omega, O \rangle$, with

- S, a finite set of states with designated initial state distribution b_0
- A, a finite set of actions
- P, the state transition model: $P(s'|s,a)$ is the probability of transitioning to state s' if action a is taken in state s
- R, the reward model: $R(s,a)$ is the expected immediate reward for taking action a in state s
- Ω, a finite set of observations
- O, the observation model: $O(o|s',a)$ is the probability of observing o if action a is taken and this results in state s'

In this paper, we consider the case in which the process unfolds over an infinite sequence of stages. At each stage the agent selects an action, which yields an immediate reward, and receives an observation. The agent must choose an action based on the history of observations seen. Note that because the state is not directly observed, it may be beneficial for the agent to remember the observation history. The objective of the agent is to maximize the expected discounted sum of rewards received. Because we are interested in the infinite sequence problem, we use a discount, $0 \leq \gamma < 1$, to maintain finite sums.

Finite-state controllers can be used as an elegant way of representing POMDP policies using a finite amount of memory. The state of the controller is based on the observation sequence, and in turn the agent's actions are based on the state of its controller. We allow for stochastic transitions and action selection, as this can help to make up for limited memory [17]. The finite-state controller can formally be defined by the tuple $\langle Q, \psi, \eta \rangle$, where Q is the finite set of controller nodes, $\psi : Q \rightarrow \Delta A$ is the action selection model for each node, and $\eta : Q \times A \times O \rightarrow \Delta Q$ represents the node transition model for each node given an action was taken and an observation seen. The value of a node q at state s, given action selection and node transition probabilities, is provided by:

$$V(q,s) = \sum_a P(a|q)\big[R(s,a) + \gamma \sum_{s'} P(s'|s,a) \sum_o O(o|s',a) \sum_{q'} P(q'|q,a,o) V(q',s')\big]$$

This equation is referred to as the Bellman equation.

2.2 Previous work

Policy Iteration (PI) [11] is a technique to find optimal POMDP controllers that alternates between policy improvement and evaluation. Although PI was originally developed for deterministic controllers, it can be extended to stochastic ones as well. In the improvement phase, dynamic programming is used to enlarge the controller. This is referred to as a backup. Nodes that have lesser or equal value for all states are removed. The incoming edges of these nodes are redirected to the dominating nodes, guaranteeing at least equal value. The new controller is then evaluated using the above Bellman equation to determine the updated value for each node and state. This process continues until the controller is no longer changed by the improvement phase. While this method guarantees that a controller arbitrarily close to optimal will be found, the controller may be very large and many unnecessary nodes may be generated along the way. This is exacerbated by the fact that the algorithm cannot take advantage of an initial state distribution and must attempt to improve the controller for any initial state.

Poupart and Boutilier [15] have developed a method called bounded policy iteration (BPI) that uses a one step dynamic programming lookahead to attempt to improve a POMDP controller without increasing its size. Like PI, it alternates between policy improvement and evaluation, but does not add nodes during the improvement phase. BPI iterates through the nodes in the controller and uses a linear program, shown in Table 1, to find a distribution over backed up values that dominates the node for all states. The linear program examines the value of probabilistically taking an action and then transitioning into the old controller, and if an improvement can be found, these values are used to update the action selection and node transition probabilities of the controller. Like PI, BPI cannot readily take advantage of an initial state distribution, thus it may require many more improvements and nodes than is necessary for a given start state. BPI guarantees to at least maintain the value of a provided controller, but it is not likely to find a concise optimal controller without adding nodes.

Meuleau et al. [14] have proposed another approach to improve a fixed-size controller. The authors use gradient ascent (GA) to change the action selection and node transition probabilities and increase value. A cross-product MDP is created from the controller and the POMDP, and matrix operations allow the gradient to be calculated. Unfortunately, this calculation does not preserve the parameters as probability distributions. This further complicates the search space and is less likely to result in a globally optimal solution.The gradient can then be followed in an attempt to improve the controller. Due to the complex and incomplete gradient calculation, this method can be time consuming and error prone.

2.3 Theoretical disadvantages of BPI and GA

Even simple POMDPs may require more advanced techniques than BPI or gradient ascent in order to find an optimal controller of a fixed size. This can be seen with the two state POMDP with two actions and one observation in Figure 1. The transitions are deterministic, with the state alternating when action $A1$ is taken in state 1 or action $A2$ is taken in state 2. When the state changes, a positive reward

For a given node q
Variables x_a and $x_{q',a,o}$
Maximize ϵ, for
Improvement constraints:

$$\forall s\ V(q,s) + \epsilon \leq \sum_a \left[x_a R(s,a) + \gamma \sum_{s'} P(s'|s,a) \sum_o O(o|s',a) \sum_{q'} x_{q',a,o} V(q',s') \right]$$

Probability constraints:

$$\sum_a\ x_a = 1 \text{ and } \forall a\ \sum_{q'} x_{q',a,o} = x_a$$

$$\forall a\ x_a \geq 0 \text{ and } \forall q', a, o\ x_{q',a,o} \geq 0$$

Table 1: The linear program for BPI. Variable x_a represents $P(a|q)$ and variable $x_{q',a,o}$ represents $P(q', a|q, o)$ for the given node, q.

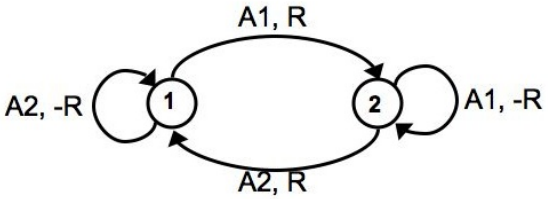

Figure 1: Simple POMDP for which BPI and GA fail to find an optimal controller

is given. Otherwise, a negative reward is given. Since there are no informative observations, given only a single node and an initial state distribution of being in either state with equal likelihood, the best policy is to choose either action with equal probability. This can be modeled by a one node stochastic controller with value equal to 0.

If the initial controller is deterministic and chooses either action, say A1, BPI will not converge to the optimal controller. The value for this controller in state 1 is $R - \gamma R/(1-\gamma)$ and $-R/(1-\gamma)$ in state 2. For $\gamma > .5$, which is common, value is negative in each state. Based on a one step lookahead, assigning any probability to the other action, A2, will raise the value for state 2, but lower it for state 1. This is because the node is assumed to have the same value after a new action is taken, rather than calculating the true value of updating action and transition probabilities. Since BPI requires that there is a distribution over nodes that increases value for all states, it will not make any improvements.

Likewise, the gradient calculation in GA will have difficulty finding the optimal controller. Because Meuleau et al. formulate the problem as unconstrained, some heuristic must be used to adjust the gradient to ensure proper probabilities are maintained. For the example problem, some heuristics will improve the controller, while others will remain stuck. In general, no method can guarantee finding the globally optimal solution. Essentially, this controller represents a local maximum for both of these methods, resulting in suboptimal behavior.

3. OPTIMAL FIXED-SIZE CONTROLLERS

The linear program used by BPI may allow for controller improvement, but can easily get stuck in local maxima. While Poupart and Boutilier suggest heuristics for becoming unstuck, we believe that a nonlinear approach is more appropriate. Using a single step or even a multiple step backup to improve a controller will generally not allow the optimal controller to be found. While one node may appear better in the short term, only an infinite lookahead can predict the true change in value.

Meuleau et al. must construct a cross-product MDP from the controller and the underlying POMDP in a complex procedure to calculate the gradient. Also, their representation does not take into account the probability constraints and thus does not calculate the true gradient of the problem. Techniques more advanced than gradient ascent may be used to traverse the gradient, but these shortcomings remain.

Unlike BPI and Meuleau et al.'s gradient ascent, our formulation allows the optimal controller to be found for a given size. This is done by considering the values of each node and state pair variables. To ensure that these values are correct given the action selection and node transition probabilities, quadratic constraints (the Bellman equations for each node and state) must be added. This results in a quadratically constrained linear program. Although it is often difficult to solve a QCLP exactly, many robust and efficient algorithms can be applied. Our QCLP has a simple gradient calculation and an intuitive representation that matches well with common optimization models. The more sophisticated nonlinearly constrained optimization techniques typically used to solve QCLPs may require more resources, but commonly produce much better results.

We believe that many POMDPs have small optimal controllers or can be approximated concisely. As the optimization complexity primarily depends on the controller size and not the size of the POMDP, this allows the algorithm to scale up to large problems. Optimization techniques permit these controllers to be found while maintaining moderate resource usage. In the next section, we give a formal description of the QCLP and a proof that it provides the optimal controller of a fixed size.

For variables: $x(q', a, q, o)$ and $y(q, s)$
Maximize

$$\sum_s b_0(s) y(q_0, s)$$

Given the Bellman constraints:

$$\forall q, s \; y(q,s) = \sum_a \left[\left(\sum_{q'} x(q', a, q, o) \right) R(s,a) + \gamma \sum_{s'} P(s'|s,a) \sum_o O(o|s', a) \sum_{q'} x(q', a, q, o) y(q', s') \right]$$

And probability constraints:

$$\forall q, o \; \sum_{q', a} x(q', a, q, o) = 1$$

$$\forall q, o, a \; \sum_{q'} x(q', a, q, o) = \sum_{q'} x(q', a, q, o_k)$$

$$\forall q', a, q, o \; x(q', a, q, o) \geq 0$$

Table 2: The quadratically constrained linear program for finding the optimal fixed-size controller. Variable $x(q', a, q, o)$ represents $P(q', a|q, o)$, variable $y(q, s)$ represents $V(q, s)$, q_0 is the initial controller node and o_k is an arbitrary fixed observation.

3.1 QCLP formulation

Unlike BPI, which alternates between policy improvement and evaluation, our quadratically constrained program improves and evaluates the controller in one phase. The value of an initial node is maximized at an initial state distribution using parameters for the action selection probabilities at each node $P(a|q)$, the node transition probabilities $P(q'|q, a, o)$, and the values of each node in each state $V(q, s)$. This approach differs from previous approaches in that it explicitly represents the node values as variables. To ensure that the values are correct given the action and node transition probabilities, nonlinear constraints must be added to the optimization. These constraints are the Bellman equations given the policy determined by the action selection and node transition probabilities.

To reduce the representation complexity, the action selection and node transition probabilities are merged into one, where:

$$P(q', a|q, o) = P(a|q) P(q', |q, a, o)$$

and

$$\sum_{q'} P(q', a|q, o) = P(a|q)$$

This results in a quadratically constrained linear program. QCLPs may contain quadratic terms in the constraints, but have a linear objective function. They are a subclass of general nonlinear programs that has structure which algorithms can exploit. This produces a problem that is more difficult than a linear program, but simpler than a general nonlinear program. The QCLP formulation also permits a large number of algorithms to be applied.

Table 2 describes the QCLP which can provide the optimal fixed-size controller. The value of a designated initial node is maximized given the initial state distribution and the necessary constraints. The first constraint represents the Bellman equation for each node and state. The second and last constraints ensure that the variables represent proper probabilities, and the third constraint guarantees that action selection does not depend on the observation that has not yet been seen.

THEOREM 1. *An optimal solution of the QCLP results in an optimal stochastic controller for the given size and initial state distribution.*

PROOF. The optimality of the controller follows from the Bellman equation constraints and maximization of a given node at the initial state distribution. The Bellman equation constraints restrict the value variables to valid amounts based on the chosen probabilities, while the maximum value is found for the initial node and state. Hence, this produces an optimal controller. □

4. METHODS FOR SOLVING THE QCLP

Constrained optimization seeks to minimize or maximize an objective function based on equality and inequality constraints. When the objective and all constraints are linear, this is called a linear program (LP). As our formulation has a linear objective, but contains some quadratic constraints, it is a quadratically constrained linear program. Unfortunately, our problem is nonconvex. Essentially, this means that there may be multiple local maxima as well as global maxima.

On the positive side, a wide range of nonlinear programming algorithms have been developed that are able to efficiently solve nonconvex problems with many variables and

constraints. Methods may also be combined to promote convergence and improve solution quality. Locally optimal solutions can be guaranteed, but at times, globally optimal solutions can also be found. For example, merit functions, which evaluate a current solution based on fitness criteria, can be used to improve convergence and the problem space can be made more convex by approximation or domain information. These methods are much more robust than gradient ascent, while retaining modest efficiency in many cases. Also, the quadratic constraints and linear objective of our problem often permits better approximations and is usually more convex than problems with higher degree objective and constraints.

For this paper, we used a freely available nonlinearly constrained optimization solver called *snopt* [9] on the NEOS server [5, 10, 6]. The algorithm finds solutions by a method of successive approximations called sequential quadratic programming (SQP). SQP uses quadratic approximations which are then solved with quadratic programming (QP) until a solution to the more general problem is found. A QP is typically easier to solve, but must have a quadratic objective function and linear constraints. In snopt, the objective and constraints are combined and approximated to produce the QP. A merit function is also used to guarantee convergence from any initial point. The POMDP and nonlinear optimization models were described using a standard optimization language AMPL, and gradients were calculated by the NEOS solver.

5. EXPERIMENTS

In this section, we compare the results of a nonlinear optimization algorithm, snopt, with those of BPI. The gradient ascent developed by Meuleau et al. was also implemented, but produced significantly worse results and required substantially more time than the other techniques. For example, the best six node controller for the machine domain examined below had a value that was over 40% lower and a time over 400% longer than the controllers found by the QCLP and BPI. GA was unable to improve larger controllers for this domain in under 8 hours. BPI was implemented in a manner that is very similar to the original techniques described in [15].

A 57 state grid world and a 256 state machine maintenance problem are used to examine the values and running time for fixed-size controllers given an initial state distribution. BPI and snopt were initialized with the same ten random deterministic controllers and best and mean solutions are reported. We also report the mean time for each algorithm to complete the optimization of a single controller. Because the NEOS server was used for running snopt, exact time comparison was not possible. Yet, based on the fact that both algorithms ran on high-end desktop computers, we can conclude that the higher-complexity of solving QCLPs does not make our approach intractable. In fact, solution times differ by a small constant factor in our experiments.

5.1 Hallway benchmark

The hallway domain, shown in Figure 2 and introduced by Littman, Cassandra and Kaelbling [13], is a frequently used benchmark for POMDP algorithms. It consists of a grid world with 57 states, 21 observations and 5 actions. There

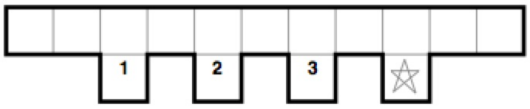

Figure 2: The grid world for the hallway domain

are 14 squares in which the robot may face north, south, east or west, and a goal square. The robot begins in a random location and orientation and must make its way to a goal state by turning, going forward or staying in place. The start state is never the same as the goal state and the observations consist of the different views of the walls in the domain. Both the observations and transitions are extremely noisy. Only the goal has a reward of 1 and the discount factor used was 0.95.

The results for this domain are shown in Table 3 and Figure 3. We see that for all controller sizes the mean value produced by the QCLP is more than twice that of BPI and the best QCLP controllers only miss this mark by a small margin in two cases. The optimal policy for the underlying MDP, which represents an upper bound on the optimal POMDP policy, was found to be 1.519. The QCLP provides a very good solution to the POMDP that is significantly closer than that found by BPI.

The time taken to produce these controllers, although difficult to compare, remains similar, but the difference increases as controller size grows. While in this case the much better results produced using QCLP required more computation time, it is not generally the case the QCLP is less efficient. The necessity of improving the controllers for all states and reliance on an LP causes BPI to produce much less improvement than the QCLP for this problem.

Poupart and Boutilier also report results for BPI on a 1500 node controller. In their implementation an escape technique, which attempts to avoid local maxima, was also used. After 69 hours, BPI produced a controller with a value of 0.51. Our QCLP formulation generates a controller with 65% higher value in 0.22% of the time with 0.67% of the representation size.

5.2 Machine maintenance

In order to test performance on a larger problem, we consider a machine maintenance domain with 256 states, 4 actions and 16 observations [4]. There are four independent components in a machine used to manufacture a part. Each component may be in 'good,' 'fair' or 'bad' condition as well

# nodes	QCLP	BPI
1	$<$ 1 min	$<$ 1 min
2	$<$ 1min	$<$ 1 min
4	$<$ 1 min	$<$ 1 min
6	1.4 mins	1.6 mins
8	6.9 mins	2.9 mins
10	9.1 mins	4 mins

Table 3: Mean running times for the QCLP and BPI on the hallway problem

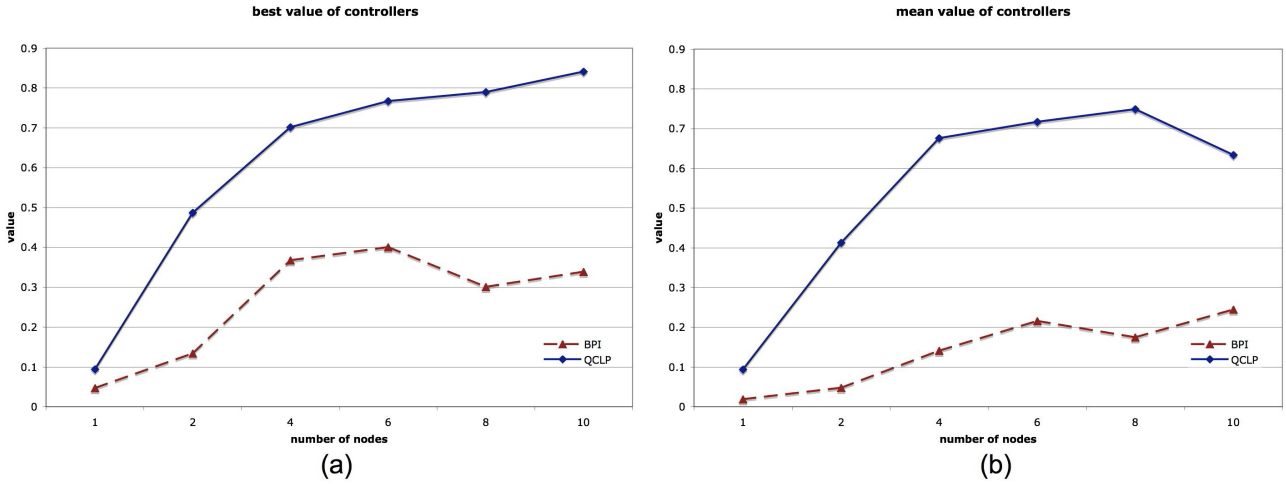

Figure 3: Hallway domain: (a) the best and (b) the mean values using BPI and the QCLP for increasing controller size

as 'broken' and in need of a replacement. Each day, four actions are possible. The machine can be used to 'manufacture' parts or we can 'inspect,' 'repair,' or 'replace' the machine. The 'manufacture' action produces 'good' parts based on the condition of the components. 'Good' components always produce 'good' parts, and 'broken' components always produce 'bad' parts. Components in 'fair' or 'bad' condition raise the probability of producing 'bad' parts. The condition of the resulting part is fully observed. Inspecting the machine causes a noisy observation of either 'good' or 'bad' for each component. Components in 'good' or 'fair' condition are more likely to be seen as 'good,' and those in 'fair' or 'broken' are more likely to be seen as 'bad.' Repairing the machine causes parts that are not 'broken' to improve one condition with high probability. The 'replace' action transitions all components to 'good' condition. Rewards for each action are: 1 for manufacturing good parts for the day, -1 for inspecting, -3 for repairing and -15 for producing bad parts. A discount factor of 0.99 was used. For more details, see [4].

For this problem, the optimal policy for the underlying MDP is known to be 66.236. We see in Table 4 and Figure 4 that very small controllers produce solutions which are close to this upper bound. The techniques using the QCLP show higher best and mean values for all controller sizes than those produced by BPI. While the differences are less dramatic than the previous example, they are no less consistent. Also, the lower mean values for BPI suggest a heavier reliance on the initial controller for producing high values. And although we again notice a disparity in running time as the number of nodes increases, we also notice that the best 10 node controller produced by BPI has lesser value and longer running time than the best 4 node controller for the QCLP. This suggests that smaller controllers could replace those produced by BPI.

6. CONCLUSIONS

We introduced a new approach for solving POMDPs using a nonlinear programming formulation of the problem, which defines an optimal fixed-size stochastic controller. Near-optimal controllers can be found using standard nonlinear optimization techniques. This provides a promising new way for solving POMDPs and approximating optimal solutions with concise controllers. We show that by using nonlinear optimization algorithms, we can produce higher valued controllers than those found by BPI, the existing state-of-the-art approach. We demonstrate empirically a consistent improvement over a range of controller sizes. We also show that better solutions can be found with significantly smaller controllers. These results suggest that optimal or near-optimal controllers could be found for large POMDPs, making our approach very useful in a range of practical application domains.

In the future, we plan to examine the relationship between the nonlinear formulation and the performance of various optimization algorithms. Different representations may better match current nonlinearly constrained optimization methods and may thus produce better solutions. We also plan to study the applicability of the QCLP approach to problems involving multiple agents modeled as decentralized POMDPs. The BPI technique has already been generalized successfully to the multi-agent case [1]. In future work, we expect to be able to demonstrate similar performance gains in optimizing fixed-size controllers for a set of agents in large multi-agent domains.

# nodes	QCLP	BPI
1	< 1 min	1.3 mins
2	< 1 min	4.6 mins
4	7.9 mins	14.1 mins
6	42.4 mins	25.5 mins
8	57.5 mins	42.9 mins
10	130.8 mins	62.8 mins

Table 4: Mean running times for the QCLP and BPI on the machine problem

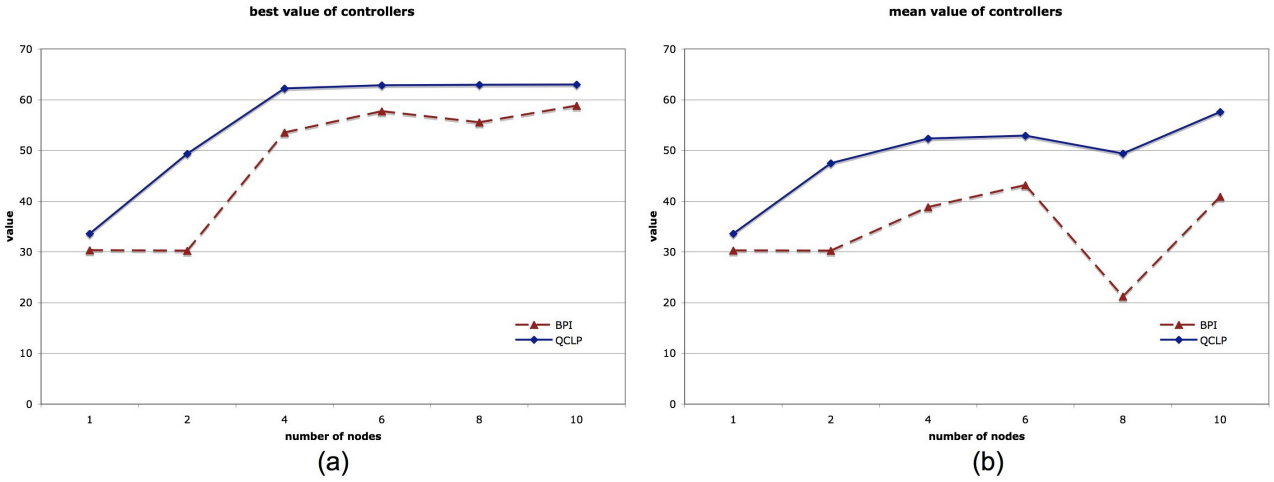

Figure 4: Machine domain: (a) the best and (b) the mean values using BPI and the QCLP for increasing controller size

7. ACKNOWLEDGMENTS

Support for this work was provided in part by the National Science Foundation under Grant No. IIS-0535061 and by the Air Force Office of Scientific Research under Agreement No. FA9550-05-1-0254.

8. REFERENCES

[1] D. S. Bernstein, E. Hansen, and S. Zilberstein. Bounded policy iteration for decentralized POMDPs. In *Proceedings of the Nineteenth International Joint Conf. on Artificial Intelligence*, Edinburgh, Scotland, 2005.

[2] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2004.

[3] A. Cassandra. A survey of pomdp applications. In *AAAI Fall Symposium*, 1998.

[4] A. R. Cassandra. *Exact and Approximate Algorithms for Partially Observable Markov Decision Processes*. PhD thesis, Brown University, Providence, RI, 1998.

[5] J. Czyzyk, M. Mesnier, and J. Moré. The NEOS server. *IEEE Journal on Computational Science and Engineering*, 5:68–75, 1998.

[6] E. Dolan. The NEOS server 4.0 administrative guide. Technical Report ANL/MCS-TM-250, Argonne National Laboratory, Mathematics and Computer Science Division, 2001.

[7] J. Eckles. Optimum maintenance with incomplete information. *Operations Research*, 16:1058–1067, 1968.

[8] Z. Feng and S. Zilberstein. Efficient maximization in solving POMDPs. In *Proceedings of the Twentieth National Conf. on Artificial Intelligence*, Pittsburgh, PA, 2005.

[9] P. E. Gill, W. Murray, and M. Saunders. Snopt: An SQP algorithm for large-scale constrained optimization. *SIAM Review*, pages 99–131, 2005.

[10] W. Gropp and J. Moré. Optimization environments and the NEOS server. *Approximation Theory and Optimization*, pages 167–182, 1997.

[11] E. A. Hansen. Solving POMDPs by searching in policy space. In *Proceedings of the Fourteenth Conf. on Uncertainty in Artificial Intelligence*, pages 211–219, Madison, WI, 1998.

[12] M. Hauskrecht and H. Fraser. Modeling treatment of ischemic heart disease with partially observable Markov decision processes. In *Proc. of American Medical Informatics Association annual symposium on Computer Applications in Health Care*, pages 538–542, 1998.

[13] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling. Learning policies for partially observable environments: Scaling up. Technical Report CS-95-11, Brown University, Department of Computer Science, Providence, RI, 1995.

[14] N. Meuleau, K.-E. Kim, L. P. Kaelbling, and A. R. Cassandra. Solving POMDPs by searching the space of finite policies. In *Proceedings of the Fifteenth Conf. on Uncertainty in Artificial Intelligence*, pages 417–426, Stockholm, Sweden, 1999.

[15] P. Poupart and C. Boutilier. Bounded finite state controllers. In *Advances in Neural Information Processing Systems*, 16, Vancouver, BC, 2003.

[16] R. Simmons and S. Koenig. Probabilistic navigation in partially observable environments. In *Proc. of the 14th International Joint Conference on Artificial Intelligence*, pages 1080–1087, 1995.

[17] S. Singh, T. Jaakkola, and M. Jordan. Learning without state-estimation in partially observable Markovian decision processes. In *Machine Learning: Proceedings of the Eleventh International Conference*, pages 284–292, 1994.