Solving POMDPs Using Quadratically Constrained Linear Programs

Christopher Amato and **Daniel S. Bernstein** and **Shlomo Zilberstein**
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
{camato,bern,shlomo}@cs.umass.edu

Abstract

Developing scalable algorithms for solving partially observable Markov decision processes (POMDPs) is an important challenge. One approach that effectively addresses the intractable memory requirements of POMDP algorithms is based on representing POMDP policies as finite-state controllers. In this paper, we illustrate some fundamental disadvantages of existing techniques that use controllers. We then propose a new approach that formulates the problem as a quadratically constrained linear program (QCLP), which defines an optimal controller of a desired size. This representation allows a wide range of powerful nonlinear programming algorithms to be used to solve POMDPs. Although QCLP optimization techniques guarantee only local optimality, the results we obtain using an existing optimization method show significant solution improvement over the state-of-the-art techniques. The results open up promising research directions for solving large POMDPs using nonlinear programming methods.

1 Introduction

Since the early 1990s, 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 [Simmons and Koenig, 1995], medical diagnosis [Hauskrecht and Fraser, 1998] and machine maintenance [Eckles, 1968]. 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 application 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 the manufactured products. Numerous other POMDP applications are surveyed in [Cassandra, 1998b].

Developing effective algorithms for MDPs and POMDPs has become a thriving AI research area [Cassandra, 1998a; Feng and Hansen, 2002; Feng and Zilberstein, 2005; Hansen, 1998; Littman *et al.*, 1995; Meuleau *et al.*, 1999; Poupart and Boutilier, 2003]. 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 [Feng and Zilberstein, 2005], the time and space complexities of POMDP algorithms remain serious challenges. POMDP approximation algorithms can operate with a limited amount of memory, but as a consequence they provide very weak theoretical guarantees, if any. In contrast, we describe a new approach that bounds space usage while permitting a principled search based on the optimal solution.

Current techniques used to find optimal fixed-size controllers rely solely on local information [Meuleau *et al.*, 1999; Poupart and Boutilier, 2003]. We present a formulation of an optimal fixed-size controller which represents an optimal POMDP policy of a given controller size. To illustrate some of its benefits, we employ a standard nonlinearly constrained optimization technique. 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 [Bertsekas, 2004]. 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, our new formulation may facilitate a more efficient search of the solution space and produces high-quality results. Moreover, the optimization algorithms employ more advanced techniques than those previously used [Meuleau *et al.*, 1999; Poupart and Boutilier, 2003] 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 can be represented as a stochastic controller. We briefly discuss some of the previous work on solving POMDPs using these controllers and then show how to represent an optimal controller using a quadratically constrained linear program (QCLP). 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. Our results suggest that by using our QCLP formulation, small, high-valued controllers can be efficiently found for a large assortment of POMDPs.

2 Background

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 transition probability of 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, resulting in state s'

We consider the case in which the decision making 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 consider 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. These controllers address one of the main reasons for intractability in POMDP exact algorithms by not remembering whole observation histories. We also allow for stochastic transitions and action selection, as this can help to make up for limited memory [Singh *et al.*, 1994]. 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, mapping nodes to distributions over actions, and $\eta : Q \times A \times O \rightarrow \Delta Q$ represents the node transition model, mapping nodes, actions and observations to distributions over the resulting nodes. The value of a node q at state s, given action selection and node transition probabilities $P(a|q)$ and $P(q'|q,a,o)$, is given by:

For a given node q and variables x_a and $x_{q',a,o}$
Maximize ϵ, for
Improvement constraints:
$\forall s\ V(q,s) + \epsilon \leq \sum_a \big[x_a R(s,a) +$
$\quad \gamma \sum_{s'} P(s'|s,a) \sum_o O(o|s',a) \sum_{q'} x_{q',a,o} V(q',s')\big]$
Probability constraints:
$\sum_a\ x_a = 1$ and $\forall a\ \sum_{q'} x_{q',a,o} = x_a$
$\forall a\ x_a \geq 0$ and $\forall q', a, o\ x_{q',a,o} \geq 0$

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

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

This equation is referred to as the Bellman equation.

2.1 Previous work

While many POMDP approximation algorithms have been developed, we will only discuss controller-based approaches, as that is the focus of our work. These techniques seek to determine the best action selection and node transition parameters for a fixed-size controller.

Poupart and Boutilier [2003] 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. This approach alternates between policy improvement and evaluation. It iterates through the nodes in the controller and uses a linear program, shown in Table 1, to examine the value of probabilistically taking an action and then transitioning into the old controller. If an improvement can be found for all states, the action selection and node transition probabilities are updated accordingly. The controller is then evaluated and the cycle continues until no further improvement can be found. BPI guarantees to at least maintain the value of a provided controller, but it is not likely to find a concise optimal controller.

A heuristic has also been proposed for incorporating start state knowledge and increasing the performance of BPI in practice [Poupart and Boutilier, 2004]. In this extension, termed biased BPI, improvement is concentrated in certain node and state pairs by weighing each pair by the (unnormalized) occupancy distribution, which can be found by solving the set of linear equations:

$$o(q',s') = bp_0(q',s') + \gamma \sum_{q,s,a,o} o(q,s) P(s'|s,a) P(a|q) O(o|s',a) P(q'|q,a,o)$$

for all states and nodes. The variables used are those previously defined and the probability, bp_0, of beginning in a node state pair. A factor, δ, can also be included, which allows value to decrease by that amount in each node and state pair. This makes changes to the parameters more likely, as a small amount of value can now be lost in note state pairs. As a result, value may be higher for the start state and node, but it

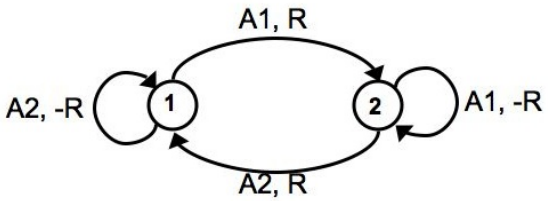

Figure 1: Simple POMDP for which BPI and GA fail to find an optimal controller

could instead decrease value for that pair. While these heuristics may increase performance, they are difficult to adjust and not applicable in all domains.

Meuleau et al. [1999] 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 by considering the states of the MDP to be all combinations of the states of the POMDP and the nodes of the controller and actions of the MDP to be based on actions of the POMDP and deterministic transitions in the controller after an observation is seen. The value of the resulting MDP can be determined 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 be followed in an attempt to improve the controller, but due to the complex and incomplete gradient calculation, this method can be time consuming and error prone.

2.2 Disadvantages of BPI and GA

Both BPI and GA fail to find an optimal controller for very simple POMDPs. 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 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. Notice that this example also shows that with limited memory (one node), a stochastic controller can provide arbitrarily greater total discounted reward than any deterministic controller of that same size.

If the initial controller is deterministic and chooses either action, say A1, BPI will not converge to an optimal controller. The value of the initial controller in state 1 is $R-\gamma R/(1-\gamma)$ and $-R/(1-\gamma)$ in state 2. For $\gamma > 0.5$, 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. Biased BPI sets equal weights for each state, thus preventing improvement. Allowing value to be lost by using a predetermined δ does not guarantee controller improvement, and for many chosen values quality may instead decrease.

Likewise, the gradient calculation in GA will have difficulty finding an optimal controller. Because Meuleau et al. formulate the problem as unconstrained, some heuristic must adjust the gradient to ensure proper probabilities are maintained. For the example problem, some heuristics will improve the controller, while others remain stuck. In general, no heuristic can guarantee finding the globally optimal solution. Essentially, this controller represents a local maximum for both of these methods, causing suboptimal behavior. One premise of our work is that a more general formulation of the problem, which defines an optimal controller, facilitates the design of solution techniques that can overcome the above limitation and produce better stochastic controllers. While no existing technique guarantees global optimality, experimental results show that our new formulation is advantageous.

In general, the linear program used by BPI may allow for controller improvement, but can easily get stuck in local maxima. While the authors suggest heuristics for becoming unstuck, our nonlinear approach offers some advantages. Using a single step or even a multiple step backup to improve a controller will generally not allow an optimal controller to be found. While one set of parameters may appear better in the short term, only the infinite lookahead defined in the QCLP representation can predict the true change in value.

GA also gets stuck often in local maxima. 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.

3 Optimal fixed-size controllers

Unlike BPI and GA, our formulation defines an optimal controller for a given size. This is done by creating a set of variables that represents the values of each node and state pair. Intuitively, this allows changes in the controller probabilities to be reflected in the values of the nodes of the controller. 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 optimization techniques used to solve QCLPs may require more resources, but commonly produce much better results than simpler methods.

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 QCLP defining an 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.

The experimental results suggest that many POMDPs have small optimal controllers or can be approximated concisely. Because of this, it may be unnecessary to use a large amount of memory in order to find a good approximation. As our approach optimizes fixed-size controllers, it allows the algorithm to scale up to large problems without using an intractable amount of space. In the rest of this section, we give a formal description of the QCLP and prove that its optimal solution defines an optimal controller of a fixed size.

3.1 QCLP formulation

Unlike BPI, which alternates between policy improvement and evaluation, our quadratically constrained linear 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 $V(q, s)$. To ensure that the value variables 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. Linear constraints are used to maintain the proper probabilities.

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

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

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 often more difficult than a linear program, but possibly 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 defines an 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 resulting observation which 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 represents 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, thus finding globally optimal solutions cannot be guaranteed.

Nevertheless, a wide range of nonlinear programming algorithms have been developed that are able to efficiently find solutions for nonconvex problems with many variables and constraints. 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

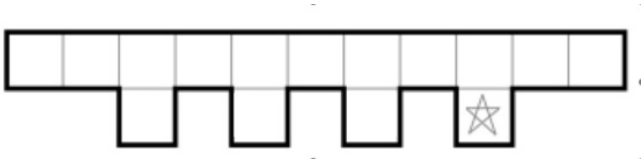

Figure 2: Hallway domain with goal state designated by a star

and the problem space can be made 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 the representation is more likely to be convex than problems with a higher degree objective and constraints.

For this paper, we used a freely available nonlinearly constrained optimization solver called *snopt* [Gill *et al.*, 2005] on the NEOS server (www-neos.mcs.anl.gov). 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.

5 Experiments

In this section, we compare the results obtained using our new formulation and the snopt solver with those of BPI and biased BPI. GA was also implemented, but produced significantly worse results and required substantially more time than the other techniques. In the interest of saving space, we omit the details of GA and focus on the more competitive techniques, BPI and biased BPI, which we implemented according to their original descriptions in [Poupart and Boutilier, 2003] and [Poupart and Boutilier, 2004]. Choices of the loss parameter, δ, in biased BPI often decreased performance in the second domain studied. Because of this, the best value for δ was found experimentally after much trial and error. This search centered around the value suggested by Poupart and Boutilier, $(max_{s,a}R(s,a) - min_{s,a}R(s,a))/400(1-\gamma)$. Note that our goal in these experiments is to demonstrate the benefits of our formulation when used in conjunction with an "off the shelf" solver such as snopt. The formulation is very general and many other solvers may be applied. In fact, we are currently developing a customized solver that would take further advantage of the inherent structure of the QCLP and increase scalability.

Two domains are used to compare the performance of the QCLP and the two versions of BPI. Each method was initialized with ten random deterministic controllers and we report mean values and times after convergence. To slightly increase the quality of the QCLP produced controllers, upper and lower bounds were added. These represent the value of taking the highest and lowest valued action respectively for an infinite number of steps. While the experiments were

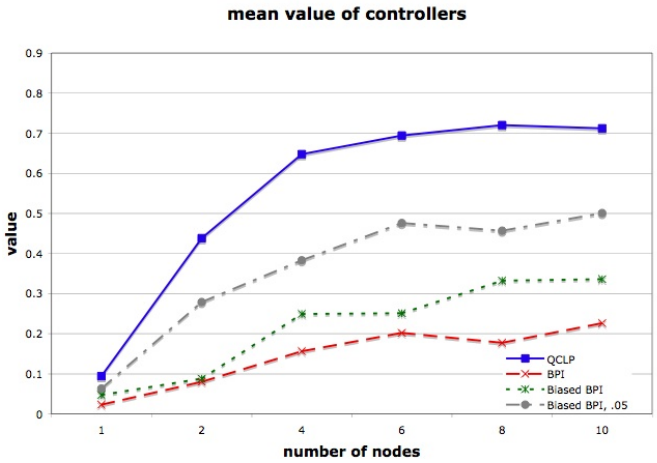

Figure 3: Hallway domain: the mean values using BPI and the QCLP for increasing controller size

conducted on different computers, we expect that this affects solution times by only a small constant factor in our experiments. Mean time for biased BPI with or without a delta is not reported as the time for each was only slightly higher than that for BPI.

5.1 Hallway benchmark

The hallway domain, shown in Figure 2 was introduced by Littman, Cassandra and Kaelbling [1995] and is a frequently used benchmark for POMDP algorithms. It consists of a grid world with 57 states, 21 observations and 5 actions. There 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 greater than those produced by each version of BPI. Biased BPI, and in particular biased BPI with a δ of 0.05, improves upon BPI, but the quality remains limited. The value of an optimal POMDP policy is not known. However, the value of an optimal policy for the underlying MDP, which represents an upper bound on an optimal POMDP policy, was

size	QCLP$_{hal}$	BPI$_{hal}$	QCLP$_{mac}$	BPI$_{mac}$
1	$<$ 1 min	$<$ 1 min	$<$ 1 min	1.2 mins
2	$<$ 1min	$<$ 1 min	$<$ 1 min	4.3 mins
4	$<$ 1 min	$<$ 1 min	7.3 mins	13.9 mins
6	1.5 mins	1.5 mins	38.2 mins	25.2 mins
8	6.9 mins	2.5mins	72.2 mins	40.2 mins
10	9.1 mins	3.7mins	122.5 mins	59.5 mins

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

found to be 1.519. The QCLP provides a very good solution to the POMDP that is significantly closer to this bound than that found by BPI.

The time taken to produce these controllers remains mostly similar, but the difference increases as controller size grows. While in this case the significantly higher quality results produced using QCLP required more computation time, it is not generally the case that the QCLP is less efficient. Also, it can be noted that a four node controller improved by the QCLP produces a higher quality solution than any BPI method for any controller size while using very little time.

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 10 node controller with 40% higher value in 0.21% of the time with 0.67% of the 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 [Cassandra, 1998a]. 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 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. Components may degrade each day unless repaired or replaced. 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 this problem, again the value of an optimal POMDP policy is not known, but the value of an optimal policy for the underlying MDP is known to be 66.236. We see in Figure 4 that very small controllers produced by the QCLP provide solutions with values which are exceptionally close (92%) to this upper bound. The QCLP generates higher quality controllers for all sizes and we are encouraged to see particularly significant performance improvement for small controller sizes. Again, a four node controller produced with the QCLP representation has a higher mean value than any BPI approach.

Running times, as seen in Table 3, are more favorable for our fomulation in this problem. The QCLP required less time than BPI for small controller sizes and remains reasonable for larger controller sizes. While more time is required for larger controller sizes, these results suggest that the QCLP representation not only permits controllers to be found that outperform those provided by BPI, but finding solutions can also be more efficient for small controller sizes. A real strength of the QCLP approach seems to be finding very compact high valued controllers.

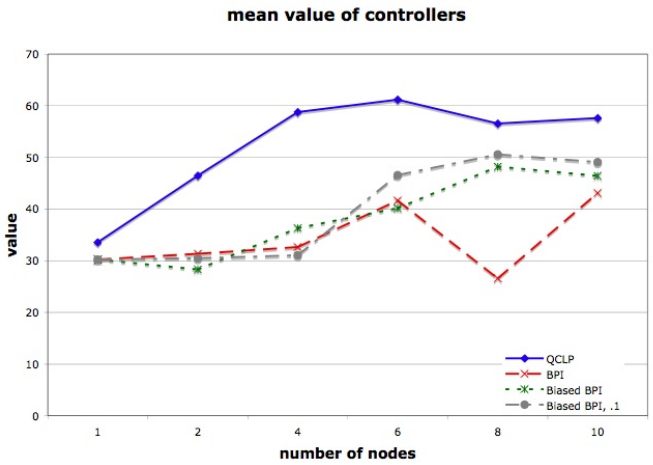

Figure 4: Machine domain: the mean values using BPI and the QCLP for increasing controller size

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. This allows a wide range of powerful nonlinear optimization algorithms to be applied. We showed that by using a standard nonlinear optimization algorithm, we can produce higher valued controllers than those found by the existing state-of-the-art approach, BPI. Our approach provided consistently higher values over a range of controller sizes even when the best possible heuristic settings were used for BPI in each domain. We also showed that better solutions can be found with significantly smaller controllers by using the QCLP. These results suggest that concise optimal or near-optimal controllers could be found for large POMDPs, making our approach very useful in a range of practical application domains. We are particularly encouraged by the ability of our approach to generate a very small controller with 40% higher value in 0.21% of the time used by BPI on a large (1500 node) controller.

This work opens up new promising research directions that could produce further improvement in both quality and efficiency. A better understanding of the relationship between the nonlinear formulation and various optimization algorithms may improve performance. Different representations may better match current nonlinearly constrained optimization methods and may thus produce better solutions. One goal is to identify subclasses of POMDPs for which globally optimal solutions can be efficiently found. Also, we are currently working on a customized QCLP solver that can take advantage of the specific structure inherent in POMDPs. This new approach shows promise for scaling up the size of controllers that are efficiently solvable with our formulation.

The QCLP technique is competitive with the state-of-the-art and in the future, we will conduct a more comprehensive evaluation of our approach with respect to a wide range of approximation techniques and domains. BPI has already been shown to be competitive with other leading POMDP approximation techniques such as Perseus [Spaan and Vlassis, 2005] and PBVI [Pineau *et al.*, 2005]. This suggests that the QCLP approach compares favorably with these techniques as well. We are confident that our experiments will verify this as well as provide insight into the strengths and weaknesses of the QCLP approach in relation to other techniques.

Finally, we have begun to study the applicability of the QCLP approach to problems involving multiple agents modeled as decentralized POMDPs [Amato *et al.*, 2006]. The BPI technique has already been generalized successfully to the multi-agent case [Bernstein *et al.*, 2005]. Based on our recent experiments, we are confident that the QCLP approach can also be used to find concise near-optimal fixed-size controllers for a set of agents in large multi-agent domains.

7 Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments. 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.

References

[Amato *et al.*, 2006] Christopher Amato, Daniel S. Bernstein, and Shlomo Zilberstein. Optimal fixed-size controllers for decentralized POMDPs. In *Proceedings of the Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains (MSDM) at AAMAS*, Hakodate, Japan, 2006.

[Bernstein *et al.*, 2005] Daniel S. Bernstein, Eric Hansen, and Shlomo Zilberstein. Bounded policy iteration for decentralized POMDPs. In *Proc. of the Nineteenth International Joint Conference on Artificial Intelligence*, Edinburgh, Scotland, 2005.

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

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

[Cassandra, 1998b] Anthony R. Cassandra. A survey of POMDP applications. In *AAAI Fall Symposium: Planning with POMDPs*, Orlando, FL, 1998.

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

[Feng and Hansen, 2002] Zhengzhu Feng and Eric A. Hansen. Symbolic heuristic search for factored Markov decision processes. In *Proc. of the Eighteenth National Conference on Artificial Intelligence*, pages 455–460, 2002.

[Feng and Zilberstein, 2005] Zhengzhu Feng and Shlomo Zilberstein. Efficient maximization in solving POMDPs. In *Proc. of the Twentieth National Conference on Artificial Intelligence*, Pittsburgh, PA, 2005.

[Gill *et al.*, 2005] Philip E. Gill, Walter Murray, and Michael Saunders. Snopt: An SQP algorithm for large-scale constrained optimization. *SIAM Review*, 47:99–131, 2005.

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

[Hauskrecht and Fraser, 1998] Milos Hauskrecht and Hamish 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.

[Littman *et al.*, 1995] Michael L. Littman, Anthony R. Cassandra, and Leslie Pack Kaelbling. Learning policies for partially observable environments: Scaling up. Technical Report CS-95-11, Brown University, Department of Computer Science, Providence, RI, 1995.

[Meuleau *et al.*, 1999] Nicolas Meuleau, Kee-Eung Kim, Leslie Pack Kaelbling, and Anthony R. Cassandra. Solving POMDPs by searching the space of finite policies. In *Proc. of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 417–426, Stockholm, Sweden, 1999.

[Pineau *et al.*, 2005] Jolle Pineau, Geoffrey Gordon, and Sebastian Thrun. Point-based approximations for fast POMDP solving. Technical Report SOCS-TR-2005.4, McGill University. School of Computer Science, Montreal, QC, 2005.

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

[Poupart and Boutilier, 2004] Pascal Poupart and Craig Boutilier. VDCBPI: an approximate scalable algorithm for large scale POMDPs. In *Advances in Neural Information Processing Systems*, 17, pages 1081–1088, Vancouver, BC, 2004.

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

[Singh *et al.*, 1994] Satinder Singh, Tommi Jaakkola, and Michael Jordan. Learning without state-estimation in partially observable Markovian decision processes. In *Machine Learning: Proceedings of the Eleventh International Conference*, pages 284–292, 1994.

[Spaan and Vlassis, 2005] Matthijs T. J. Spaan and Nikos Vlassis. Perseus: Randomized point-based value iteration for POMDPs. *Journal of AI Research*, 24:195–220, 2005.