Optimizing Fixed-Size Stochastic Controllers for POMDPs

Christopher Amato and **Daniel S. Bernstein** and **Shlomo Zilberstein**
Department of Computer Science
University of Massachusetts
Amherst, MA 01003 USA

Abstract

In this paper, we discuss a new approach that represents POMDP policies as finite-state controllers and formulates the optimal policy of a desired size as a nonlinear program (NLP). This new representation allows a wide range of powerful nonlinear programming algorithms to be used to solve POMDPs. Although solving the NLP optimally is often intractable, the results we obtain using an off-the-shelf optimization method are competitive with state-of-the-art POMDP algorithms. Our approach is simple to implement and it opens up promising research directions for solving POMDPs using nonlinear programming methods.

Introduction

Developing effective algorithms for partially observable Markov decision processes (POMDPs) is an active research area that has seen significant progress (Cassandra 1998; Hansen 1998; Ji *et al.* 2007; Littman, Cassandra, & Kaelbling 1995; Pineau, Gordon, & Thrun 2003; Poupart 2005; Poupart & Boutilier 2003; Smith & Simmons 2004; 2005; Spaan & Vlassis 2005). Despite this progress, it is generally accepted that exact solution techniques are limited to toy problems due to high memory requirements. Consequently, approximation algorithms are often necessary. Approximation methods perform well in many problems, but have some known disadvantages. For instance, point-based methods (Ji *et al.* 2007; Pineau, Gordon, & Thrun 2003; Spaan & Vlassis 2005; Smith & Simmons 2004; 2005) perform better with a small reachable belief space. Other algorithms rely on linear programming (Poupart & Boutilier 2003) or require fine tuning of heuristic parameters to produce high-valued solutions (Poupart 2005).

In this paper, we explore a solution technique for POMDPs that optimizes fixed-size controllers. Our approach formulates the optimal memory-bounded solution as a nonlinear program (NLP), and can provide a solution by utilizes existing nonlinear optimization techniques. Nonlinear programming is an active field of research that has produced a wide range of techniques that can efficiently solve a variety of large problems (Bertsekas 2004). Parameters are optimized for a fixed-size controller which produces the policy. The formulation provides a new framework for which future algorithms can be developed. We discuss how to solve these problems optimally, but as this would often be intractable in practice, we also evaluate an effective approximation technique using standard NLP solvers. Our approach facilitates scalability as it offers a tradeoff between solution quality and the available resources. That is, for a given amount of memory, we can search for a controller that is optimal for that size. Large controllers may be needed to provide a near optimal solution in some problems, but our experiments suggest that smaller controllers produce high quality solutions in a wide array of problems.

The rest of the paper is organized as follows. We first give an overview of the POMDP model and explain how infinite-horizon solutions can be represented as stochastic controllers. We discuss some previous work and then show how to define optimal controllers using a nonlinear program. We then provide experimental results using a generic off-the-shelf NLP solver in various POMDP domains. Our approach is competitive with other state-of-the-art solution methods in terms of solution quality and running time, but it generally produces higher quality when given less memory. Our results show that by using the NLP formulation, concise, high-valued controllers can be efficiently found for a large assortment of POMDPs. Furthermore, performance is likely to improve as more specialized and scalable NLP solvers are developed that can take advantage of the structure of the controller optimization problem.

Background

We begin with an overview of the POMDP model and then discuss the relevant POMDP approximate algorithms.

The POMDP model

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 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 the resulting state is 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 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. This will help the agent to identify the set of possible states at any step. The objective of the agent is to maximize the expected discounted sum of rewards received. Because we consider the infinite-horizon problem, we use a discount factor, $0 \leq \gamma < 1$, to maintain finite sums.

Finite-state controllers can be used as an elegant way of representing POMDP policies using a finite amount of memory (Hansen 1998). The state of the controller is based on the observation sequence seen, and in turn the agent's actions are based on the state of its controller. To help distinguish states of the finite-state controller from states of the POMDP, we will refer to controller states as nodes. These controllers address one of the main causes of intractability in POMDP exact algorithms by not storing whole observation histories. Thus states of the controller can encapsulate key information about the observation history in a fixed number of nodes. We also allow for stochastic transitions and action selection, as this can help to make up for limited memory (Singh, Jaakkola, & Jordan 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 the following Bellman equation:

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

Previous work

A controller-based approach was developed by Poupart and Boutilier (Poupart & Boutilier 2003) called bounded policy iteration (BPI). BPI uses a one step dynamic programming lookahead to attempt to improve a POMDP controller without increasing its size. Like PI, this approach also alternates between policy improvement and evaluation. It iterates through the nodes in the controller and uses a linear program to examine the value of probabilistically taking an action and then transitioning into the current 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 also does not use start state information resulting in larger than necessary controllers that are not likely to be optimal. A heuristic was also added in biased BPI (Poupart 2005) which incorporates start state knowledge and allows value to be lost at some state-node pairs.

Other approximation approaches that have recently gained a high level of popularity are point-based methods such as point-based value iteration (PBVI) (Pineau, Gordon, & Thrun 2003), PERSEUS (Spaan & Vlassis 2005), heuristic search value iteration (HSVI) (Smith & Simmons 2004) and point-based policy iteration (PBPI) (Ji *et al.* 2007). These techniques are approximations of value iteration for finite-horizon POMDPs, but they can find solutions for sufficiently large horizons that they can effectively produce infinite-horizon policies.

PBVI, PERSEUS, HSVI and PBPI use value iteration while fixing the number of belief points considered at each step. That is, at each step, the value vector of each belief point under consideration is backed up and the vector that has the highest value at that point is retained. This permits the number of vectors to remain constant and the value function is still defined over the entire belief space. If the right points are chosen, it may help concentrate the value iteration on certain regions in the belief space. PBVI selects the set of belief points by sampling from the action and observation sets of the problem while PERSEUS backs up only a randomly selected subset of these points. Thus, PERSEUS is often faster and produces a more concise solution than PBVI. PBVI has been shown to be optimal for sufficiently large and well constructed belief point sets, but this is intractable for reasonably sized problems. HSVI chooses points based on an additional upper bound that is incorporated into the algorithm. This algorithm will also converge to the optimal solution, but again this often requires an intractable amount of resources. PBPI uses Hansen's PI, but with PBVI rather than exact improvement over the whole belief space in the improvement phase. This allows faster performance and sometimes higher values than PBVI.

Optimizing fixed-size controllers

Unlike other controller based approaches, our formulation defines an optimal controller for a given size. This is done by creating a set of new variables that represent 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. This is in contrast to backups used by other methods which iteratively improve the probabilities and easily get stuck in local optima. Our approach allows both the values and probabilities in the controller to be optimized in one step, thus representing the optimal fixed-size controller. To ensure that these values are correct given the action selection and node transition probabilities, nonlinear constraints (based on the Bellman equation) must be added. The resulting NLP is generally harder to solve, but many robust and efficient algorithms can be applied.

One premise of our work is that an optimal formulation of the problem facilitates the design of solution techniques that can overcome the limitations of previous controller-based algorithms and produce better stochastic controllers. The general nature of our formulation allows a wide range of solution methods to be used. This results in a search that is

more sophisticated than previously used in controller-based methods. While no existing technique guarantees global optimality in a finite amount of time, experimental results show that our new formulation is advantageous. For instance, our results suggest that many POMDPs have small optimal controllers or can be approximated concisely with finite state-controllers. Thus, it is often unnecessary to use a large amount of memory in order to represent a good approximation. Our NLP is also able to take advantage of the start distribution of the problem, thus making better use of limited controller size. Lastly, because our method searches for stochastic controllers, it is able to find higher-valued, more concise controllers than search in the space of deterministic controllers.

Compared to point-based approaches, our formulation does not need to choose a set of points and may be able to cover the belief space better in some problems. That is, while point-based methods work well when there is a small reachable belief space or when the chosen points are very representative of the whole space, the NLP approach seeks to optimize the value of a controller for a specific initial point. For domains in which point-based methods cannot find representative belief points, our approach may still be able to construct high quality controllers. Also, since point-based methods rely on finite-horizon dynamic programming, it may be difficult for these methods to complete the number of backups necessary to approximate the value function well. As our approach uses finite-state controllers, it is more suitable for finding infinite-horizon policies.

NLP formulation

Unlike BPI, which alternates between policy improvement and evaluation, our nonlinear 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 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 simpler than a general nonlinear program. The QCLP formulation permits a large number of algorithms to be applied. Because the QCLP is a subclass of the general NLP, we will refer to the QCLP formulation as an NLP.

Table 1 describes the NLP 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 which maintains correct values as probability parameters change. The second and third constraints ensure that the x variables represent proper probabilities and the last 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 NLP in Table 1 results in an optimal stochastic controller for the given size and initial state distribution.*

Proof The optimality of the controller follows from the one-to-one correspondence between the objective function of the NLP and the value of the POMDP at the initial state distribution. The Bellman equation constraints restrict the value variables to be consistent with the chosen action selection and transition probabilities. The remaining constraints guarantee that the action selection and transition probabilities are selected from valid distributions. Hence, the optimal solution of this NLP represents the value of a fixed-size controller that is optimal for the given POMDP. □

Methods for solving the NLP

Many efficient constrained optimization algorithms can be used to solve large NLPs. When the objective function and all constraints are linear, this is called a linear program (LP). As our formulation has a linear objective function, but contains some quadratic constraints, it is a quadratically constrained linear program. Unfortunately, this problem is nonconvex. Essentially, this means that there may be multiple local maxima as well as global maxima, thus finding globally optimal solution is often very difficult.

It is worth noting that global optimization techniques can be applied to our NLP by reformulating it as a DC (difference of convex functions) programming problem. Global optimization algorithms can then be used, but because of the large size of the resulting DC program, it is unlikely that current optimal solvers can handle even small POMDPs. Nevertheless, it would be interesting to identify classes of problems for which DC optimization is practical. Details of the transformation from our NLP to DC optimization problem can be found in (Amato, Bernstein, & Zilberstein 2007).

Since it may not be possible or feasible to solve the NLP optimally, locally optimal methods are often more useful in practice. A wide range of nonlinear programming algorithms have been developed that are able to efficiently solve 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 and the problem space can be made convex by approximation or domain information. These methods are much more robust than simpler methods such as gradient ascent, while retaining modest efficiency in many cases.

For this paper, we used a freely available nonlinearly constrained optimization solver called *snopt* on the NEOS server (www-neos.mcs.anl.gov). The algorithm finds solu-

For variables: $x(q', a, q, o)$ and $z(q, s)$, Maximize $\sum_s b_0(s) z(q_0, s)$, subject to

The Bellman constraints:

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

Probability constraints:

$$\sum_{q',a} x(q', a, q, o) = 1, \forall q, o \qquad x(q', a, q, o) \geq 0, \forall q', a, q, o$$

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

Table 1: The NLP defining an optimal fixed-size controller. Variable $x(q', a, q, o)$ represents $P(q', a|q, o)$, variable $z(q, s)$ represents $V(q, s)$, q_0 is the initial controller node and o_k is an arbitrary fixed observation.

tions 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.

Experiments

For experimental comparison, we present an evaluation of the performance of our formulation using an off-the-shelf nonlinear program solver, snopt, as well as leading POMDP approximation techniques. In these experiments we seek to determine how well our formulation performs when used in conjunction with a generic solver such as snopt. The formulation is very general and many other solvers may be applied. We are currently developing a customized solver that would take further advantage of the inherent structure of the NLPs and increase scalability.

To improve scalability, we also use a version of our NLP formulation in which the action at each node is fixed. As the transitions between the nodes remain stochastic, this only reduces the expressiveness of the controller by a small amount due to the first action being chosen deterministically. While we could have allowed this first action to be stochastic, it was found that choosing this action greedily based on immediate reward at the initial belief often performed reasonably well. To set the other actions of the controller, we cycled through the available actions and assigned the next available action to the next node. That is, if a problem has 4 actions and 9 nodes then each action is represented twice in the controller except for the greedy first action which is represented 3 times. Given a fixed controller size, fixing the action selection will usually perform worse than allowing stochastic actions, but the problem is simpler and thus larger controller sizes can be solved. When there were multiple highest-valued actions, one of these actions is randomly assigned to the first node. The hope is that although fixing actions will result in less concise controllers, higher-valued, larger controllers may be found. Throughout this section, we will refer to the optimization of our NLP using snopt as *NLO* for the fully stochastic case and as *NLO fixed* for the optimization with fixed actions.

In this section, we compare the results obtained using our new formulation with those of biased BPI, PBVI, PERSEUS, HSVI and PBPI. Each of our NLP methods was initialized with ten random deterministic controllers and we report mean values and times after convergence. To slightly increase the performance of the solver upper and lower bounds were added. These represent the value of taking the highest and lowest valued actions respectively for an infinite number of steps. All results were found by using the NEOS server which provides a set of machines with varying CPU speeds and memory limitations.

Benchmark problems

We first provide a comparison of our NLP formulations with leading POMDP approximation methods on common benchmark problems. The first three domains, which were introduced by Littman, Cassandra and Kaelbling (1995), are grid problems in which an agent must navigate to a goal square. The other benchmark problem is a larger grid problem in which the goal is for the agent to catch and tag an opponent which attempts to move away (Pineau, Gordon, & Thrun 2003). Following the convention of previously published results, we consider the versions of these problems that stop after the goal has been reached and a discount factor of 0.95 was used.

Table 2 shows the results from previously published algorithms and our NLP formulations. We provide the mean values and times for the largest controller size that is solvable with less than (approximately) 400MB of memory and under eight hours on the NEOS server. The values of PBPI in the table are the highest values reported for each problem, but the authors did not provide results for the Hallway problem. Because the experiments were conducted on different computers, solution times give a general idea of the speed of the algorithms. It is also worth noting that while most of the other approaches are highly optimized, a generic solver is used with our approach in these experiments.

In general, we see that the nonlinear optimization approach is competitive both in running time and value produced, but does not outperform the other techniques. Our

Tiger-grid $|S| = 36$, $|A| = 5$, $|\Omega| = 17$

	value	size	time
HSVI	2.35	4860	10341
PERSEUS	2.34	134	104
HSVI2	2.30	1003	52
PBVI	2.25	470	3448
PBPI	2.24	3101	51
biased BPI	2.22	120	1000
NLO fixed	2.20	32	1823
BPI	1.81	1500	163420
NLO	1.79	14	1174
Q_{MDP}	0.23	n.a.	2.76

Hallway $|S| = 60$, $|A| = 5$, $|\Omega| = 21$

	value	size	time
PBVI	0.53	86	288
HSVI2	0.52	147	2.4
HSVI	0.52	1341	10836
biased BPI	0.51	43	185
PERSEUS	0.51	55	35
BPI	0.51	1500	249730
NLO fixed	0.49	24	330
NLO	0.47	12	362
Q_{MDP}	0.27	n.a.	1.34

Hallway2 $|S| = 93$, $|A| = 5$, $|\Omega| = 17$

	value	size	time
PERSEUS	0.35	56	10
HSVI2	0.35	114	1.5
PBPI	0.35	320	3.1
HSVI	0.35	1571	10010
PBVI	0.34	95	360
biased BPI	0.32	60	790
NLO fixed	0.29	18	240
NLO	0.28	13	420
BPI	0.28	1500	274280
Q_{MDP}	0.09	n.a.	2.23

Tag $|S| = 870$, $|A| = 5$, $|\Omega| = 30$

	value	size	time
PBPI	-5.87	818	1133
PERSEUS	-6.17	280	1670
HSVI2	-6.36	415	24
HSVI	-6.37	1657	10113
biased BPI	-6.65	17	250
BPI	-9.18	940	59772
PBVI	-9.18	1334	180880
NLO fixed	-10.48	5	2117
NLO	-13.94	2	5596
Q_{MDP}	-16.9	n.a.	16.1

Table 2: Values, representation sizes and running times (in seconds) for the set of benchmark problems. Results for other algorithms were taken from the following sources: BPI (Poupart & Boutilier 2003), biased BPI (Poupart 2005), HSVI (Smith & Simmons 2004), HSVI2 (Smith & Simmons 2005), PERSEUS (Spaan & Vlassis 2005), PBPI (Ji *et al.* 2007), PBVI (Pineau, Gordon, & Thrun 2003), Q_{MDP} (Spaan & Vlassis 2005)

method achieves 94%, 92%, 83% of max value in the first three problems, but does so with much smaller representation size. A key aspect of the NLP approach is that high quality solutions can be found with very concise controllers. Thus, limited representation size is very well utilized, likely

Machine $|S| = 256$, $|A| = 4$, $|\Omega| = 16$

	value	size	time
HSVI2	63.17	575	317
NLO fixed	62.65	20	3963
NLO	61.74	10	7350
biased BPI	59.75	20	30831
PERSEUS	39.28	86	2508

Aloha 30 $|S| = 90$, $|A| = 29$, $|\Omega| = 3$

	value	size	time
HSVI2	1212.15	2909	1841
NLO	1211.67	6	1134
NLO fixed	1076.49	26	2014
biased BPI	993.22	22	5473
PERSEUS	853.24	114	2512

Table 3: Values, representation sizes and running times (in seconds) for the machine maintenance and aloha problems.

better than the other approaches in most problems.

The values in the table as well as those in Figure 1 show that our approach is currently unable to find solutions for large controller sizes. For example, in the Tag problem, a solution could only be found for a two node controller in the general case and a five node controller when the actions were fixed. As seen in the figures, there is a near monotonic increase of value in the Hallway problem when compared with either controller size or time. As expected, fixing the actions at each node results in faster performance and increased scalability. This allows higher valued controllers to be produced, but scalability remains limited. As improved solvers are found higher values and larger controllers will be solvable by both approaches. The other benchmark problems display similar trends of near monotonic improvement and the need for increased scalability to outperform other methods on these problems.

Other domains

We also examined the approximate approaches in two other domains, the machine maintenance and aloha problems (Cassandra 1998). The machine maintenance domain has 256 states, 4 actions and 16 observations and the the aloha problem is a simplified networking problem using the slotted Aloha protocol with 90 states, 30 actions and 3 observations. Discount factors of 0.99 and 0.999 were used for the two problems respectively. For HSVI2 and PERSEUS, software was used from web sites of Trey Smith and Matthijs Spaan respectively. For biased BPI results, we used our own implementation and like our NLP results, the mean values and times of optimizing ten fixed-size deterministic controllers are reported. The heuristic parameters were found experimentally by choosing a wide range of values and reporting the best results. Similarly, for PERSEUS, several methods for choosing a set of belief points of size 10000 were used and again, the best results are reported.

On these problems, which we provide results for in Table 6, our NLP formulation provides results that are nearly identical (over 99% of the value) on the machine problem (NLO fixed) and the aloha problem (NLO). Both of our methods also outperform our versions of biased BPI and PERSEUS in

almost all evaluation criteria in both problems. While these two algorithms may be able to increase their performance by adjusting heuristics, we believe that our approach will continue to produce results that will match this performance without the need to adjust heuristics. One reason for this is that these problems are difficult for point-based approaches to solve optimally. This is due to the high discount factor causing many backups to be required and the need for many vectors to closely approximate the optimal value function. For instance, in the aloha domain, after 15 hours HSVI2 was able to produce a value of 1212.92 which required over 15000 vectors to represent. In contrast, while the mean value produced by our nonlinear optimization was 1211.67, a controller with value 1217.11 was found with only six nodes in about 20 minutes. This shows the value of our controller-based technique as a more efficient alternative to point-based methods in problems such as these.

Conclusions

In this paper, we introduced a new approach for solving POMDPs by defining optimal fixed-size solutions as nonlinear programs. This permits a wide range of powerful NLP solution methods to be applied to these problems. As a controller-based approach, our method may be able to provide solutions that are higher quality than point-based methods such as those with high discount rate or periodic optimal policies. Because our approach optimizes a stochastic controller it is able to make efficient use of the limited representation space. Thus, this new formulation is simple to implement and may allow very concise high quality solutions to a large number POMDPs.

We showed that by using a generic NLP solver, our formulation can provide solutions that are competitive with leading POMDP approximation techniques in terms of running time and solution quality. Given the complexity of the problems that we considered, it is not surprising that no single approach dominates all the others. Nevertheless, our approach can produce good value with significantly smaller memory. And because the controller is more compact, in some cases the approach uses less time despite the use of nonlinear optimization. This opens up new promising research directions that could produce further improvement in both quality and efficiency. As the solver in our experiments has not been optimized for our NLP it is likely that even higher performance can be achieved in the future.

It is also worth noting that we have extended this NLP approach to the multiagent DEC-POMDP framework (Amato, Bernstein, & Zilberstein 2007). A set of fixed-size independent controllers is optimized, which when combined, produce the policy for the problem. We demonstrated that our approach is able to produce solutions that are often significantly higher-valued than the other infinite-horizon DEC-POMDP algorithms for a range of problems.

In the future, we plan to explore more specialized algorithms that can be tailored for our optimization problem. While the performance achieved with a standard nonlinear optimization algorithm is good, specialized solvers might be able to further increase solution quality and scalability. Different methods may be able to take advantage of the specific structure inherent in POMDPs. Also, important subclasses could be identified for which globally optimal solutions can be efficiently provided. These improvements may allow optimal or near optimal fixed-size solutions to be found for important classes of POMDPs.

Acknowledgments

An earlier version of this paper without the fixed action extension and additional discussion and experiments appeared in UAI-07. More details and an extension to the multiagent DEC-POMDP framework can be seen in the tech report. 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, C.; Bernstein, D. S.; and Zilberstein, S. 2007. Optimizing fixed-size stochastic controllers for POMDPs and decentralized POMDPs. Technical Report CS-07-70, University of Massachusetts, Department of Computer Science, Amherst, MA.

Bertsekas, D. P. 2004. *Nonlinear Programming*. Athena Scientific.

Cassandra, A. R. 1998. *Exact and Approximate Algorithms for Partially Observable Markov Decision Processes*. Ph.D. Dissertation, Brown University, Providence, RI.

Hansen, E. A. 1998. Solving POMDPs by searching in policy space. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, 211–219.

Ji, S.; Parr, R.; Li, H.; Liao, X.; and Carin, L. 2007. Point-based policy iteration. In *Proceedings of the Twenty-Second National Conference on Artificial Intelligence*.

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

Pineau, J.; Gordon, G.; and Thrun, S. 2003. Point-based value iteration: an anytime algorithm for POMDPs. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*.

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

Poupart, P. 2005. *Exploiting Structure to Efficiently Solve Large Scale Partial ly Observable Markov Decision Processes*. Ph.D. Dissertation, University of Toronto.

Singh, S.; Jaakkola, T.; and Jordan, M. 1994. Learning without state-estimation in partially observable Markovian decision processes. In *Proceedings of the Eleventh International Conference on Machine Learning*.

Smith, T., and Simmons, R. 2004. Heuristic search value iteration for POMDPs. In *Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence*.

Smith, T., and Simmons, R. 2005. Point-based POMDP algorithms: Improved analysis and implementation. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*.

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