Optimal Fixed-Size Controllers for Decentralized POMDPs

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

Abstract

Solving decentralized partially observable Markov decision processes (DEC-POMDPs) is a difficult task. Exact solutions are intractable in all but the smallest problems and approximate solutions provide limited optimality guarantees. As a more principled alternative, we present a novel formulation of an optimal fixed-size solution of a DEC-POMDP as a nonlinear program. We discuss the benefits of this representation and evaluate several optimization methods. While the methods used in this paper only guarantee locally optimal solutions, a wide range of powerful nonlinear optimization techniques may now be applied to this problem. We show that by using our formulation in various domains, solution quality is higher than a current state-of-the-art approach. These results show that optimization can be used to provide high quality solutions to DEC-POMDPs while maintaining moderate memory and time usage.

1 Introduction

While a large body of work has been developed for partially observable Markov decision processes (POMDPs), their multiagent counterpart, decentralized POMDPs (DEC-POMDPs), have only recently gained popularity. The DEC-POMDP models the more general problem of a set of agents which possess joint outcomes based on the actions of all agents. Each agent must then make decisions based on imperfect information of the system state or uncertainty about the other agents. Solutions then seek to maximize shared reward using solely local information for each agent.

Applications include robot control [2, 4] and networking [3, 7]. Robots typically have sensors that provide uncertain and incomplete information about the state of the environment and the location of the other robots. This lack of information must be factored into the planning process. In a decentralized network, each node must make decisions about when and where to send packets without full knowledge of the structure or actions of the rest of the network. Other applications are e-commerce and space exploration systems.

Several exact and approximate algorithms have recently been developed to solve DEC-POMDPs [2, 3, 4, 7, 8]. Current exact algorithms that use dynamic programming often require an intractable amount of space even though an optimal or near-optimal solution may be concise. DEC-POMDP approximation algorithms can operate with a limited amount of memory, but as a consequence may provide solutions that are unboundedly below the optimal. We propose a new approach that addresses the space requirement of DEC-POMDP algorithms while maintaining well-defined optimality guarantees. This approach formulates the optimal fixed-size solution to

the DEC-POMDP as a nonlinear program (NLP), thus allowing a wide range of powerful nonlinear optimization techniques to be applied. While the NLP algorithms used in this paper only guarantee a locally optimal solution, the optimization permits a principled approximation when the optimal solution is infeasible.

In this paper, we first present some background on the DEC-POMDP model and current algorithms used to solve it. We then describe some flaws in these methods and, as an alternative, present a nonlinear program that represents the optimal fixed-size solution. Lastly, experimental results are provided comparing two locally optimal nonlinear optimization methods and a current state-of-the-art DEC-POMDP approximation algorithm. For a range of domains and controller sizes, higher-valued controllers are found with the NLP, suggesting that high quality, concise controllers can be found in many diverse DEC-POMDP domains.

2 DEC-POMDP model and solutions

We first review the decentralized partially observable Markov decision process (DEC-POMDP) model and how to represent solutions as finite state controllers. For clarity, we present the model for two agents as it is straightforward to extend it to n agents.

A two agent DEC-POMDP can be defined with the following tuple:
$M = \langle S, A_1, A_2, P, R, \Omega_1, \Omega_2, O \rangle$
with

- S, the set of states with designated initial state distribution b_0
- A_1 and A_2, the sets of actions
- P, the state transition probabilities: $P(s'|s, a_1, a_2)$, the probability of transitioning to state s' given the previous state was s and actions a_1 and a_2 were taken by agents 1 and 2 respectively
- R, the reward function: $R(s, a_1, a_2)$, the immediate reward for being in state s and agent 1 taking action a_1 and agent 2 taking action a_2
- Ω_1 and Ω_2, the sets of observations
- O, the observation probabilities: $O(o_1, o_2|s', a_1, a_2)$, the probability of agents 1 and 2 seeing observations o_1 and o_2 respectively given agent 1 has taken action a_1 and agent 2 has taken action a_2 causing the state to transition to s'

At each step, every agent chooses an action based on their local observation histories, resulting in an immediate reward and an observation for each agent. Histories are used because the true state is unknown. Solutions to DEC-POMDPs, or policies, are defined by mappings from local observation histories to actions for each agent. We focus on maximizing the infinite horizon discounted reward, thus we employ a discount factor, $0 \leq \gamma < 1$. Also, communication may take place in a DEC-POMDP, but is not explicitly modeled or assumed.

As a way to model DEC-POMDP policies with finite memory, finite state controllers provide an appealing solution. Each agent's policy can be represented as a local controller and the resulting set of controllers supply the joint policy. Each finite-state controller can formally be defined by the tuple $\langle Q, \psi, \eta \rangle$, where Q is the finite set of controller nodes, $\psi : Q \to \Delta A$ is the action selection model for each node, and $\eta : Q \times A \times O \to \Delta Q$ represents the node transition model for each node given an action was taken and an observation seen. For two agents, the value for starting in agent 1's node q_1 and agent 2's node q_2 at state s is given by

$$V(q_1, q_2, s) = \sum_{a_1, a_2} P(a_1|q_1)P(a_2|q_2)[R(s, a_1, a_2)+$$

$$\gamma \sum_{s'} P(s'|a_1, a_2, s) \sum_{o_1, o_2} O(o_1, o_2|s', a_1, a_2) \sum_{q_1', q_2'} P(q_1'|q_1, a_1, o_1)P(q_2'|q_2, a_2, o_2)V(q_1', q_2', s')]$$

This is also referred to as the Bellman equation. Note that the values can be calculated offline in order to determine controllers for each agent that can then be executed online for distributed decision making.

3 Previous work

Several exact and approximate DEC-POMDP algorithms have been developed recently, but each has significant drawbacks. Most notably, DEC-POMDP exact algorithms have been developed by Hansen et al. [7] and Becker et al. [2] while approximate algorithms have been developed by Nair et al. [8], Emery-Montemerlo et al. [4] and Bernstein et al. [3].

While Hansen et al.'s approach allows the general DEC-POMDP to be solved optimally, it requires an intractable amount of memory for any but the smallest problems. This method generalizes dynamic programming for the POMDP case and builds up a controller one step at a time. At each step, new nodes are added for all possible action and observation pairs, an exponential increase. Nodes which have lesser value for all possible controllers of the other agents are then removed. This process continues until the controllers are 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.

In order to combat the high complexity of Hansen et al.'s approach, Becker et al. developed an algorithm that is able to take advantage of the structure of certain DEC-POMDPs to find an optimal solution much more efficiently. It was found that if each agent has full local information (such as its own location, battery level, etc.), and independent transitions (i.e., each agent's actions only affect that agent), the problem complexity drops considerably (NP-complete versus NEXP-complete) and coordination is still possible as the rewards in the system remain dependent on the actions of all agents. The algorithm is able to solve larger problems than Hansen's method, but does not scale well as problem size increases. While this approach identifies an interesting DEC-POMDP subclass, there are many instances where these assumptions do not hold.

As an alternative to exact methods, several DEC-POMDP approximation algorithms have been developed. Two of the best techniques are those of Nair et al. and Emery-Montemerlo et al. Both algorithms iterate through the set of agents, holding all policies fixed except one and choosing the best policy for that agent. This continues until there is no change in the policies selected. While this approach will allow a locally optimal solution to be found, it may be very poor and still requires the space of all policies for a single agent to be searched. Nair et al. propose a way to scale up their algorithm to larger problems by incorporating dynamic programming, but effects are limited in practice. Emery-Montemerlo et al. scale up their algorithm by approximating the problem by a series of simpler ones. While this increases solvable problem size, it decreases solution quality.

Bernstein et al. use a different approach called decentralized bounded policy iteration (DEC-BPI) to find an approximate solution. Fixed-size controllers are used for each agent and improved

by a linear program. The linear program attempts to improve each controller by examining the value of taking a different first action and then transitioning into the old controller. If the value is higher for controllers of the other agents, the change is made. This allows memory to remain fixed, but provides only a locally optimal solution. This is due to the linear program considering the old controller values from the second step on, and the fact that improvement must be over all possible controllers of the other agents. As the number of agents or size of controllers grows, this later drawback is likely to severely hinder improvement. Bernstein et al. allow each agent's controller to be correlated by using a shared information in a "correlation device." This improves solution quality, but also increases solution size that may be better used in the each agent's controller.

4 Nonlinear optimization approach

Due to the high space complexity of finding an optimal solution for a DEC-POMDP, fixed-size solutions are very appealing. Fixing memory balances optimality and computational concerns and should allow high quality solutions to be found when compared with other current approximation methods. Using Bernstein et al.'s method reduces problem complexity by fixing controller size, but only locally optimal solutions can be found. Also, each agent's controller is improved separately without consideration for the knowledge of the initial problem state, thus reducing solution quality. Both of these limitations can be eliminated by modeling a set of optimal controllers as a nonlinear program (NLP). By setting the value as a variable and using constraints to maintain validity, the parameters can be updated in order to find the globally optimal solution over the infinite horizon of the problem. Also, the NLP improves and evaluates the controllers of all agents at once for a given initial state in order to make the best possible use of the controller size.

Compared with other DEC-POMDP algorithms, the NLP approach makes more efficient use of memory than the exact methods, and unlike approximate methods, provides theoretical guarantees. Rather than adding nodes and then attempting to remove those that will not improve the controller, we search for the best controllers of a fixed size. The NLP is also able to take advantage of the start distribution, thus making better use of its size. Contrary to Becker et al.'s algorithm, it is able to solve the general DEC-POMDP problem and unlike approximate methods, the optimal solution to the NLP provides the optimal fixed size controllers for each agent.

The NLP approach has already shown promise in the POMDP case. In a previous paper [1], we have modeled the optimal fixed-size controller for a given POMDP as a NLP and with locally optimal solution techniques produced consistently higher quality controllers than a current state-of-the-art method. The success of the NLP in the single agent case suggested a similar approach could also be successful in multiagent problems.

4.1 Nonlinear problem model

The nonlinear program seeks to optimize the value of fixed-size controllers given a initial state distribution and the DEC-POMDP model. The parameters of the two agent version of this problem are the joint action selection probabilities at each node of the two controllers $P(a_1, a_2|q_1, q_2)$, the joint node transition probabilities $P(q_1', q_2'|q_1, q_2, a_1, a_2, o_1, o_2)$ and the values of each node in each state, $V(q_1, q_2, s)$. This approach differs from Bernstein et al.'s approach 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 optimiza-

For variables: $x(q_1, q_2, a_1, a_2)$, $y(q_1, q_2, a_1, a_2, o_1, o_2, q_1', q_2')$ and $z(q_1, q_2, s)$
Maximize

$$\sum_s b_0(s) z(q_1^0, q_2^0, s)$$

Given the Bellman constraints:

$$\forall q_1, q_2, s \; z(q_1, q_2, s) = \sum_{a_1, a_2} x(q_1, q_2, a_1, a_2) \Big[R(s, a_1, a_2) +$$

$$\gamma \sum_{s'} P(s'|s, a_1, a_2) \sum_{o_1, o_2} O(o_1, o_2|s', a_1, a_2) \sum_{q_1', q_2'} y(q_1, q_2, a_1, a_2, o_1, o_2, q_1', q_2') z(q_1', q_2', s') \Big]$$

Independence constraints:

$$\forall a_1, q_1, q_2 \; \sum_{a_2} x(q_1, q_2, a_1, a_2) = \sum_{a_2} x(q_1, q_2^c, a_1, a_2)$$

$$\forall a_2, q_1, q_2 \; \sum_{a_1} x(q_1, q_2, a_1, a_2) = \sum_{a_1} x(q_1^c, q_2, a_1, a_2)$$

$$\forall a_1, a_2, q_1, q_2, o_1, o_2, q_1' \; \sum_{q_2'} y(q_1, q_2, a_1, a_2, o_1, o_2, q_1', q_2') = \sum_{q_2'} y(q_1, q_2^c, a_1, a_2^c, o_1, o_2^c, q_1', q_2')$$

$$\forall a_1, a_2, q_1, q_2, o_1, o_2, q_2' \; \sum_{q_1'} y(q_1, q_2, a_1, a_2, o_1, o_2, q_1', q_2') = \sum_{q_1'} y(q_1^c, q_2, a_1^c, a_2, o_1^c, o_2, q_1', q_2')$$

And probability constraints:

$$\forall q_1 \; \sum_{a_1, a_2} x(q_1, q_2^c, a_1, a_2) = 1$$

$$\forall q_2 \; \sum_{a_1, a_2} x(q_1^c, q_2, a_1, a_2) = 1$$

$$\forall q_1, o_1, a_1 \; \sum_{q_1', q_2'} y(q_1, q_2^c, a_1, a_2^c, o_1, o_2^c, q_1', q_2') = 1$$

$$\forall q_2, o_2, a_2 \; \sum_{q_1', q_2'} y(q_1^c, q_2, a_1^c, a_2, o_1^c, o_2, q_1', q_2') = 1$$

$$\forall q_1, q_2, a_1, a_2 \; x(q_1, q_2, a_1, a_2) \geq 0 \text{ and } \forall q_1, q_2, o_1, o_2, a_1, a_2 \; y(q_1, q_2, a_1, a_2, o_1, o_2, q_1', q_2') \geq 0$$

Table 1: The nonlinear program for finding the optimal fixed-size controller. Variable $x(q_1, q_2, a_1, a_2)$ represents $P(a_1, a_2|q_1, q_2)$, variable $y(q_1, q_2, a_1, a_2, o_1, o_2, q_1', q_2')$ represents $P(q_1', q_2'|q_1, q_2, a_1, a_2, o_1, o_2)$, variable $z(q_1, q_2, s)$ represents $V(q_1, q_2, s)$, q_1^0 is the initial controller node for agent 1, and q_2^0 is the initial controller node for agent 2. Superscripted c's such as q_1^c represent arbitrary fixed values.

tion. These constraints are the Bellman equations given the policy determined by the action and transition probabilities. Constraints are also added to ensure distributed action selection and node transitions for each agent. We must also ensure that all probabilities are valid numbers between 0 and 1.

Table 1 describes the nonlinear program used to find the optimal two agent controller. The

value of designated initial local nodes is maximized given the initial state distribution and the necessary constraints. The independence constraints guarantee that action selection and transition probabilities can be summed out for each agent by ensuring that they do not depend on any information that is not local. Note that we provide the two agent version of the NLP, but it is straightforward to extend it to n agents.

Theorem 1 *An optimal solution of the NLP results in optimal stochastic controllers for the given size and initial state distribution.*

Proof. The optimality of the controllers follows from the NLP constraints and maximization of given initial nodes at the initial state distribution. The Bellman equation constraints restrict the value variables to valid amounts based on the chosen probabilities, the independence constraints guarantee distributed control and the maximum value is found for the initial nodes and state. Hence, this produces optimal controllers.

4.2 Nonlinear solution techniques

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 contains some nonlinear constraints, it is a nonlinearly constrained optimization problem or nonlinear program. In general, our problem is nonconvex making it often intractable to solve exactly. Essentially, nonconvex problems may have multiple local maxima as well as global maxima.

Fortunately, 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.

For this paper, we used freely available nonlinearly constrained optimization solvers called *snopt* [6] and *filter* [5] on the NEOS server (http://www-neos.mcs.anl.gov/neos/). Each algorithm finds solutions by a method of successive approximations called sequential quadratic programming (SQP). SQP uses quadratic approximations which are then more efficiently 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. Filter also uses SQP, but adds a "filter" which tests the current objective and constraint violations against those of previous steps in order to promote convergence. The DEC-POMDP and nonlinear optimization models were described using a standard optimization language AMPL and gradients were calculated by the NEOS solver.

5 Experimental results

In each experiment, we compare Bernstein et al.'s DEC-BPI with NLP solutions using filter and snopt for a range of controller sizes. Each algorithm was run until convergence was achieved with ten different random initial controllers, and the mean values and times are reported. The

# nodes	BPI ind	BPI cor	NLP algs
1	4.687	6.290	9.1
2	4.068	7.749	9.1
3	8.637	7.781	9.1
4	7.857	8.165	9.1

Table 2: Values for Bernstein's independent and correlated controllers compared with NLP results in the broadcast channel domain

times reported for each NLP method can only be considered estimates due to running each algorithm on external machines with uncontrollable load levels. DEC-BPI also allows solutions to be improved by correlating each agent's controller through shared information. Although this technique was not used in the nonlinear algorithms, we include results with a two state correlation device as another source of comparison. Future work could include a similar device for the nonlinear formulation.

5.1 Broadcast channel

A small DEC-POMDP used by Bernstein et al. was a simplified two agent networking example. This problem has 4 states, 2 actions and 5 observations. At each time step, each agent must choose whether or not to send a message. If both agents send, there is a collision and neither gets through. A reward of 1 is given for every step a message is successfully sent over the channel and all other actions receive no reward. Agent 1 has a 0.9 probability of having a message in its queue on each step and agent 2 has only a 0.1 probability. The domain is initialized with only agent 1 possessing a message and a discount factor of 0.9 was used.

Table 2 shows the values produced by Bernstein et al.'s approach. Both nonlinear algorithms produce the same value, 9.1 for each controller size. In all cases this is a higher value than that produced by Bernstein's independent and correlated approaches. It turns out that the policy found by the NLP approach is for only agent 1 to send messages. Since it has a 0.9 probability of having a message to send, a vast majority of the time, the channel is used. A more sophisticated policy might be possible, but the fact that Bernstein et al.'s algorithm did not find this simple policy while each NLP method did shows the promise of the NLP approach.

The time used by each algorithm is shown in Table 3. As expected, as controller size grows, the computation time also grows. Both NLP algorithms require more time than either version of BPI, and the mean time used by filter in this domain is higher than that of snopt while the same controller values are produced. As noted above, solution quality is also higher using nonlinear optimization. Either NLP approach can produce a one node controller in an amount of time similar to each DEC-BPI method and that solution is both more concise and higher valued.

# nodes	snopt	filter	DEC-BPI	DEC-BPI corr
1	1s	1s	< 1s	< 1s
2	2s	3s	< 1s	2s
3	14s	764s	2s	7s
4	188s	4061s	5s	24s

Table 3: Broadcast channel mean optimization times using NLP methods snopt and filter as well as those for DEC-BPI

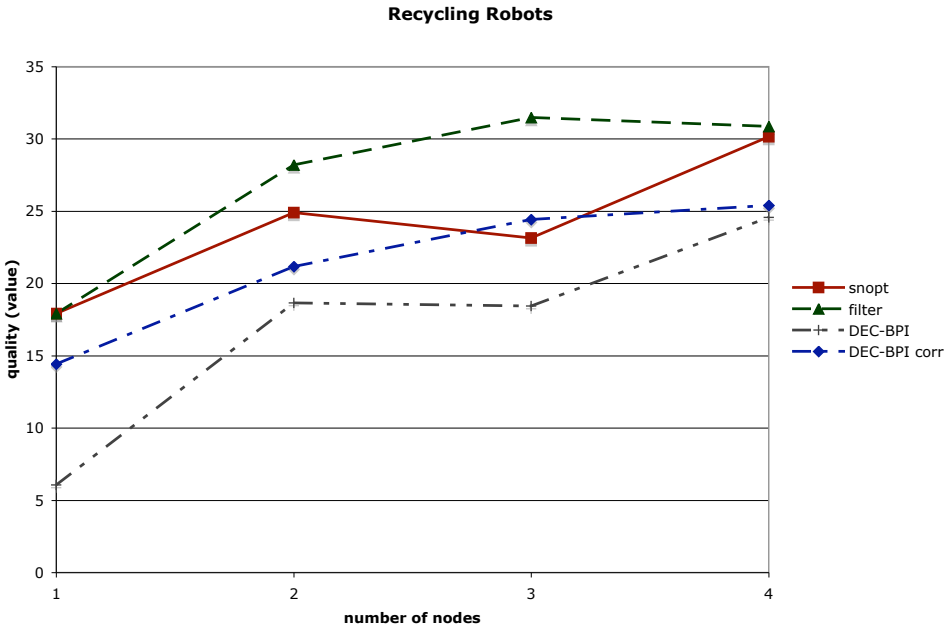

Figure 1: Recycling robots values using NLP methods snopt and filter as well as those using DEC-BPI

5.2 Recycling robots

As another comparison, we have extended the Recycling Robot problem [9] to the multiagent case. The robots have the task of picking up cans in an office building. They have sensors to find a can and motors to move around the office in order to look for cans. The robots are able to control a gripper arm to grasp each can and then place it in an on-board receptacle. Each robot has three high level actions: (1) search for a small can, (2) search for a large can or (3) recharge the battery. In our two agent version, the larger can can only be retrieved if both robots pick it up at the same time. Each agent can decide to independently search for a small can or to attempt to cooperate in order to receive a larger reward. If only one agent chooses to retreive the large can, no reward is given. For each agent that picks up a small can, a reward 2 is given and if both agents cooperate to pick the large can, a reward of 5 is given. The robots have

# nodes	snopt	filter	DEC-BPI	DEC-BPI corr
1	1s	1s	< 1s	< 1s
2	2s	4s	< 1s	1s
3	26s	64s	1s	3s
4	523s	635s	3s	10s

Table 4: Recycling robots mean optimization times using NLP methods snopt and filter as well as those for DEC-BPI

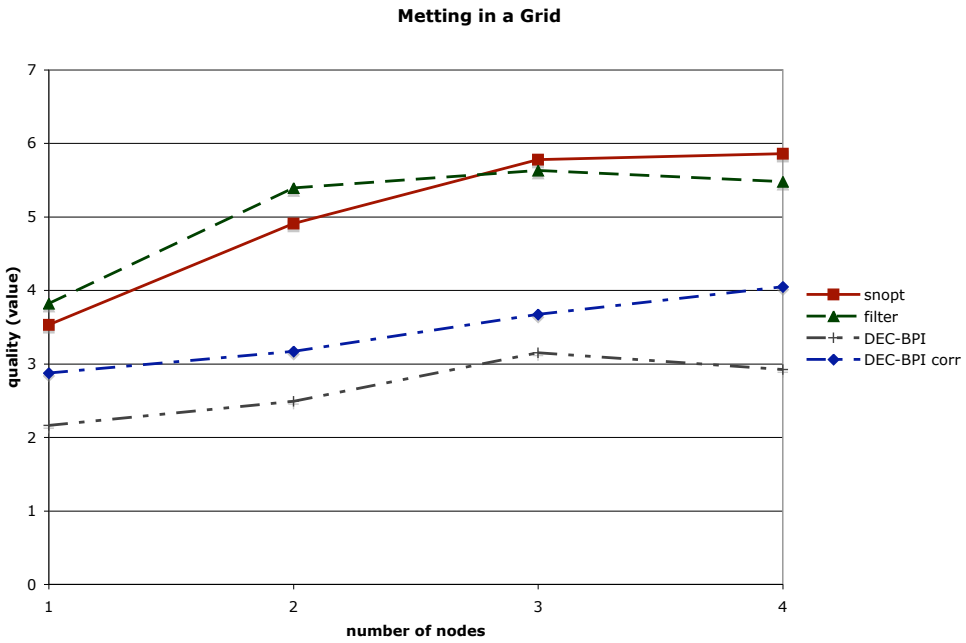

Figure 2: Metting in a 2x2 grid values using NLP methods snopt and filter as well as those using DEC-BPI

the same battery states of high and low, with an increased likelihood of transitioning to a lower state after attempting to pick up the large can. Each robot's battery power depends only on its own actions and each agent can fully observe its own level, but not that of the other agent. If the robot exhausts the battery, it is picked up and plugged into the charger and then continues to act. The two robot version used in this paper has 4 states, 3 actions and 2 observations.

We can see in Figure 1 that in this domain higher quality controllers are produced by using nonlinear optimization. Both NLP methods permit higher mean values than independent controllers using BPI for all controller sizes. The correlated controllers achieve values comparable to snopt for three node controllers, but are otherwise lower then either NLP method. Filter is particularly effective with two and three node controllers and provides values similar to snopt in other cases. Value improvements using filter compared to DEC-BPI with independent controllers range from 195% for one node controllers to 26% for four node controllers.

The times depicted in Table 4 show that even a locally optimal solution of the NLP requires more time than DEC-BPI for each controller size. We also notice that smaller controllers are able to produce higher value using the NLP, allowing more concise and efficient solutions. For instance, either NLP algorithm generates higher quality two node controllers in similar time to independent DEC-BPI controllers and less time than correlated DEC-BPI controllers with four nodes.

# nodes	snopt	filter	DEC-BPI	DEC-BPI corr
1	3s	2s	1s	2s
2	4s	5s	8s	31s
3	54s	110s	39s	151s
4	873s	2098s	118s	638s

Table 5: Meeting in a 2x2 grid mean optimization times using NLP methods snopt and filter as well as those for DEC-BPI

5.3 Meeting in a grid

The last domain is much larger than those above. In this problem, two agents must meet in a 2 by 2 grid with no obstacles. The available actions are left, right, up and down and stay. Only walls to the left and right can be observed. Action transitions are noisy and a reward of 1 is given for each step the agents share the same square. This is a domain with 16 states, 5 actions and 2 observations.

A graph of the mean values for each controller size is shown in Figure 2. In this case we see a dramatic difference between the quality of DEC-BPI solutions and each nonlinear algorithm. While both NLP algorithms perform similarly, they produce controllers with nearly twice the value of independent BPI controllers for each size and 20% to 70% increases over the correlated controllers.

The mean time taken for convergence is shown in Figure 5. This domain shows a more favorable performance by nonlinear optimization techniques. Both NLP algorithms require less time than either DEC-BPI method for two node controllers, and when correlated controllers are considered, snopt and filter require less time in two of four problem instances. Also, two and three node controllers produced by the NLP have higher value while requiring less time than either DEC-BPI approach using four nodes.

6 Conclusion

We introduced a novel formulation of a fixed-size solution for a DEC-POMDP as a type of nonlinear program. The above experiments using nonlinear optimization show a significant improvement over the current state-of-the art. Consistently higher valued controllers are produced by using a NLP than those produced by Bernstein et al.'s method. While the time taken to find even a locally optimal solution to the NLP can be higher, the fact that higher values can be found with smaller controllers using the NLP suggests using more powerful optimization techniques on smaller controllers can be very productive. The combination of start state knowledge and more advanced optimization allows us to make efficient use of the limited space of the controllers. These results show that this method can allow compact optimal or near-optimal controllers to be found for large DEC-POMDPs.

In the future, we plan to explore more specialized algorithms for the NLP representation and conduct a more exhaustive comparison with other DEC-POMDP solution methods. The general nonlinear optimization algorithms performed well in this paper, but algorithms more suitable for the problem may greatly increase scalability. Comparisons with other DEC-POMDP algorithms will further evaluate the usefulness of this approach in various domains.

7 Acknowledgments

We would like to thank Marek Petrik for helpful discussions of this work. This work was supported in part by the Air Force Office of Scientific Research (Grant No. FA9550-05-1-0254) and by the National Science Foundation (Grant No. 0535061). Any opinions, findings, conclusions or recommendations expressed in this manuscript are those of the authors and do not reflect the views of the US government.

References

[1] C. Amato, D. S. Bernstein, and S. Zilberstein. Solving POMDPs using quadratically constrained linear programs. In *Proceedings of the Ninth International Symposium on Artificial Intelligence and Mathematics*, Fort Lauderdale, FL, 2006.

[2] R. Becker, S. Zilberstein, V. Lesser, and C. Goldman. Solving transition-independent decentralized Markov decision processes. *Journal of AI Research*, 22:423–455, 2004.

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

[4] R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun. Approximate solutions for partially observable stochastic games with common payoffs. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, New York, NY, 2004.

[5] R. Fletcher, S. Leyffer, and P. L. Toint. On the global convergence of a filter-SQP algorithm. *SIAM Journal of Optimization*, 13:44–59, 2002.

[6] P. E. Gill, W. Murray, and M. Saunders. Snopt: An SQP algorithm for large-scale constrained optimization. *SIAM Review*, 47:99–131, 2005.

[7] E. A. Hansen, D. S. Bernstein, and S. Zilberstein. Dynamic programming for partially observable stochastic games. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, San Jose, CA, 2004.

[8] R. Nair, D. Pynadath, M. Yokoo, M. Tambe, and S. Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, Acapulco, Mexico, 2003.

[9] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.