

Mixed-integer Linear Programming for Transition-independent Decentralized MDPs

Jianhui Wu
EECS Department, University of Michigan
Ann Arbor, MI 48109 USA
jianhuiw@umich.edu

Edmund H. Durfee
EECS Department, University of Michigan
Ann Arbor, MI 48109 USA
durfee@umich.edu

Categories and Subject Descriptors

I.2.8 [ARTIFICIAL INTELLIGENCE]: Problem Solving, Control Methods, and Search

General Terms

Algorithms

Keywords

MDP, transition-independent decentralized MDP, piecewise linear approximation, mixed integer linear programming

1. INTRODUCTION

Markov decision process (MDP) provides a framework for computing optimal policies for individual agents operating in uncertain environments. However, extending single-agent MDP techniques to multiagent problems is not straightforward. Previous complexity analyses have shown that the general decentralized Markov decision process (Dec-MDP) is NEXP-complete, which means that optimally solving a Dec-MDP is extremely difficult. The class of problems studied in this paper is a subclass of Dec-MDP in which two or more cooperative agents are tied together through the rewards of completing joint tasks but the actions taken by one agent do not impact other agents' transitions. Although this reduces the complexity class to NP-complete [4], efficiently solving such transition independent Dec-MDPs is still nontrivial.

Optimal algorithms, such as the coverage-set algorithm [2], slow down dramatically as the size of the problem increases. Approximation algorithms, such as the JESP algorithm [5], can quickly find a locally optimal joint policy, but the solutions usually lack quality guarantees. Other collaborative multiagent research often makes the simplifying assumption that each agent has the same (complete) world view. Thus, there remains a critical need for new algorithms that can efficiently build optimal or near-optimal policies for transition independent Dec-MDPs. In this paper, we present a novel mathematical programming algorithm for this purpose.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...\$5.00.

2. MDPS AND LINEAR PROGRAMMING

A classical MDP can be defined as a four-tuple $\langle S, A, P, R \rangle$, where S is a finite state space, A is a finite action space, $P = \{p_{i,a,j}\}$ is the state transition probability where $p_{i,a,j}$ is the probability that the agent reaches state j if it executes action a in state i , and $R = \{r_{i,a}\}$ is the reward function where $r_{i,a}$ is the reward that the agent receives if it executes action a in state i .

One approach to solving MDPs is based upon linear programming (LP) [1]. Let $x_{i,a}$ represent the expected number of times action a is executed in state i . Then the objective function $\sum_i \sum_a x_{i,a} \times r_{i,a}$ represents the total expected reward. Now, consider the linear program shown in Eq.1.

$$\max \sum_i \sum_a x_{i,a} \times r_{i,a} \quad \left| \begin{array}{l} \sum_a x_{j,a} = \alpha_j + \sum_i \sum_a p_{i,a,j} \times x_{i,a} \\ x_{i,a} \geq 0 \end{array} \right. \quad (1)$$

where α_j is the probability that the system is initially in state j . The constraint $\sum_a x_{j,a} = \alpha_j + \sum_i \sum_a p_{i,a,j} \times x_{i,a}$ indicates that the expected number of times state j is visited must equal the initial probability distribution at state j plus the expected number of times state j is entered via all possible transitions.

If we solve Eq.1, it is trivial to derive the optimal policy. Specifically, assigning a probability of executing action a in state i as $\pi_{i,a} = \frac{x_{i,a}}{\sum_a x_{i,a}}$ will maximize the total expected reward.

3. ALGORITHM

In the class of problems studied in this paper, multiple cooperative agents aim to maximize their total expected reward in a stochastic environment where actions taken by one agent cannot impact other agents' transitions. Each agent can fully observe its own local state but can only have a partial view of the global environment. An agent receives rewards by completing its individual tasks. Agents can also work together to complete joint tasks. The accomplishment of a joint task needs participation from multiple agents. That is, if every (relevant) agent completes its subtask of the joint task, then the group receives a joint reward; otherwise they get zero reward for that joint task. Since agents might not be able to complete all available tasks due to limitations (such as time), they need to determine the subset of tasks they should pursue, and align their policies.

Let $p_{i,a,j}^m$, $r_{i,a}^m$, and α_j^m , respectively, represent the transition probabilities, rewards (or costs) for individual activities, and initial probability distribution for agent m , let $p_{i,a}^{m,k}$

represent the probability that agent m completes its subtask of joint task k when executing action a in state i , and let R^k represent the reward for accomplishing joint task k (i.e., $R^k > 0$). Our problem can then be formulated as

$$\begin{aligned} \max \quad & \sum_m \sum_i \sum_a x_{i,a}^m \times r_{i,a}^m + \sum_k \rho^k \times R^k \\ \text{s.t.} \quad & \sum_a x_{j,a}^m = \alpha_j^m + \sum_i \sum_a p_{i,a,j}^m \times x_{i,a}^m \\ & \rho^{m,k} = \sum_i \sum_a p_{i,a}^{m,k} \times x_{i,a}^m \\ & \rho^k = \prod_m \rho^{m,k} \\ & x_{i,a}^m \geq 0 \end{aligned} \quad (2)$$

where $x_{i,a}^m$ is the expected number of times that action a is executed in state i for agent m . $\rho^{m,k} = \sum_i \sum_a p_{i,a}^{m,k} \times x_{i,a}^m$ is the expected number of times agent m completes its subtask of joint task k . Note that it is generally not difficult to restrict an autonomous agent from completing a particular task more than once, either through an additional bit in the state representation or through a fixed acyclic ordering for completing the tasks, and so $\rho^{m,k}$ can also represent the probability of agent m completing its subtask of joint task k . $\rho^k = \prod_m \rho^{m,k}$ is the probability that the group successfully completes joint task k .

The term $\sum_m \sum_i \sum_a x_{i,a}^m \times r_{i,a}^m$ in the objective function of Eq.2 is the total expected reward (or cost) of individual activities, and the term $\sum_k \rho^k \times R^k$ is the total expected reward from accomplishing joint tasks. The constraint $\sum_a x_{j,a}^m = \alpha_j^m + \sum_i \sum_a p_{i,a,j}^m \times x_{i,a}^m$ is a multiagent version of the constraint in Eq.1.

It is important to emphasize that, unlike an LP formulation (e.g., Eq.1), Eq.2 uses nonlinear constraints to represent cooperative activities of agents (instead of using the joint state space) and thus avoids exponential increase in the size of the mathematical formulation. In the following two subsections, we present discretization and piecewise linear approximation techniques, and then reformulate Eq.2 into a mixed integer linear program (MILP) that can be efficiently solved by state-of-art algorithms [3].

3.1 Discretization

We now introduce a discretization technique that, although it cannot eliminate the nonlinearities in Eq.2 by itself, paves the way for using piecewise linear approximation technique presented in Section 3.2.

Let $\mathbb{H}$ denote an ordered set of constants (named *thresholds*) in which $0 \leq h_1 < h_2 < \dots < h_i < \dots \leq 1$, and define binary variables Θ_h^k . Then, with additional constraints Eq.3,

$$\begin{cases} \frac{\rho^k}{h} \geq \Theta_h^k \\ \sum_h \Theta_h^k \leq 1 \\ \Theta_h^k \in \{0, 1\} \end{cases} \quad (3)$$

the objective function of Eq.2 can be approximated as

$$\sum_m \sum_i \sum_a x_{i,a}^m \times r_{i,a}^m + \sum_k \sum_{h \in \mathbb{H}} h \times \Theta_h^k \times R^k \quad (4)$$

The constraint $\frac{\rho^k}{h} \geq \Theta_h^k$ is equivalent to $\Theta_h^k = 0$ when $\rho^k < h$, and $\Theta_h^k \in \{0, 1\}$ when $\rho^k \geq h$. The constraint $\sum_h \Theta_h^k \leq 1$ guarantees that, for each joint task k , there is at most one nonzero integer in the set $\{\Theta_h^k \mid h \in \mathbb{H}\}$. These two constraints work together to make $\sum_{h \in \mathbb{H}} h \times \Theta_h^k \leq \rho^k$. Since rewards of joint tasks are always positive, $\sum_{h \in \mathbb{H}} h \times \Theta_h^k$ equals the largest h in the set $\{h \mid h \in \mathbb{H} \text{ and } h \leq$

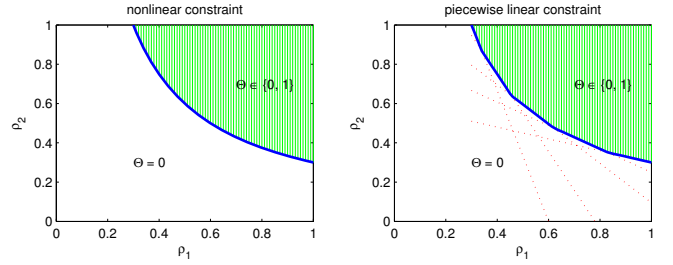


Figure 1: An example of piecewise linear approximation ($h = 0.3$, two agents).

$\rho^k\}$ when Eq.4 is maximized. That is, $\sum_{h \in \mathbb{H}} h \times \Theta_h^k$ can nicely approximate ρ^k when the maximum gap between two adjacent thresholds in $\mathbb{H}$ is sufficiently small.

3.2 Piecewise Linear Approximation

With the discretization technique, we are ready to remove the nonlinearities. Note that, in our new formulation (i.e., adding Eq.3 into Eq.2, and replacing Eq.2's objective function with Eq.4), ρ^k are trivial variables, which can be taken away by combining constraints $\rho^k = \prod_m \rho^{m,k}$ and $\frac{\rho^k}{h} \geq \Theta_h^k$ into a new constraint $\frac{\prod_m \rho^{m,k}}{h} \geq \Theta_h^k$.

In general, the nonlinear constraint $\frac{\prod_m \rho^{m,k}}{h} \geq \Theta_h^k$ can be closely approximated with a sufficient number of linear constraints. Figure 1 shows a simple example where a nonlinear constraint $\frac{\rho^1 \times \rho^2}{h} \geq \Theta_h$ ($h = 0.3$) is approximated by five linear constraints.

Explicitly, for M agents, we can build a piecewise linear matrix for threshold h as:

$$\mathbb{A}_h = \begin{bmatrix} \frac{1}{\bar{q}_1^1 \times M} & \cdots & \frac{1}{\bar{q}_1^M \times M} & \cdots & \frac{1}{\bar{q}_1^M \times M} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{1}{\bar{q}_i^1 \times M} & \cdots & \frac{1}{\bar{q}_i^M \times M} & \cdots & \frac{1}{\bar{q}_i^M \times M} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{1}{\bar{q}_n^1 \times M} & \cdots & \frac{1}{\bar{q}_n^M \times M} & \cdots & \frac{1}{\bar{q}_n^M \times M} \end{bmatrix}$$

where $\vec{\bar{q}}_i = \langle \bar{q}_i^1, \dots, \bar{q}_i^M \rangle$ represents a point on the function $\prod_m \rho^m = h$.¹ $\frac{\prod_m \rho^{m,k}}{h} \geq \Theta_h^k$ can then be approximated as $\mathbb{A}_h \times \vec{\rho}^k \geq \Theta_h^k$, where $\vec{\rho}^k = \langle \rho^{1,k}, \dots, \rho^{M,k} \rangle$.

3.3 MILP-HCS Algorithm

Using the approximation techniques presented above, a transition-independent Dec-MDP problem can then be formulated as the following mixed integer linear program.

$$\begin{aligned} \max \quad & \sum_m \sum_i \sum_a x_{i,a}^m \times r_{i,a}^m + \sum_k \sum_{h \in \mathbb{H}} h \times \Theta_h^k \times R^k \\ \text{s.t.} \quad & \sum_a x_{j,a}^m = \alpha_j^m + \sum_i \sum_a p_{i,a,j}^m \times x_{i,a}^m \\ & \rho^{m,k} = \sum_i \sum_a p_{i,a}^{m,k} \times x_{i,a}^m \\ & \mathbb{A}_h \times \vec{\rho}^k \geq \Theta_h^k \\ & \sum_h \Theta_h^k \leq 1 \\ & x_{i,a}^m \geq 0 \\ & \Theta_h^k \in \{0, 1\} \end{aligned} \quad (5)$$

¹In this work, we use a greedy algorithm to find n points $\vec{\bar{q}}_i$ for $\mathbb{A}_h$.

We can then use state-of-art MILP solvers for it, and derive a joint policy in a similar way to that described in Section 2. That is, agent m executes action a at state i with probability $\pi_{i,a}^m = \frac{x_{i,a}^m}{\sum_a x_{i,a}^m}$.

Not surprisingly, with the approximation techniques, our MILP-based algorithm cannot guarantee optimal solutions. However, we can prove that, in the worst case, the algorithm suffers a loss of $\sum_k (e + g) \times R^k$ from the optimal total expected reward, where g denotes the maximum error caused by the discretization, which can be reduced by increasing the size of $\mathbb{H}$, and e denotes the maximum error caused by the piecewise linear approximation, which can be reduced by using more points $\vec{v}_i$ in $\mathbb{A}_h$.

To further improve performance, we integrate the MILP algorithm with hill-climbing search, and iterate MILP and hill-climbing search with gradually increasing approximation accuracy. The details are presented in Algorithm 1.

Algorithm 1 MILP algorithm integrated with hill-climbing search (MILP-HCS)

```

1: Initialize approximation accuracy parameters  $e$ ,  $g$ , and
   randomly choose a joint policy  $randomSol$ 
2:  $initSol \leftarrow HillClimbSearch(problem, randomSol)$ 
3: repeat
4:    $milp \leftarrow CreateMILP(problem, e, g)$ 
5:    $nearOptimalSol \leftarrow SolveMILP(milp, initSol)$ 
6:    $sol \leftarrow HillClimbSearch(problem, nearOptimalSol)$ 
7:    $initSol \leftarrow sol$ 
8:    $e, g \leftarrow ImproveAccuracy(e, g)$ 
9: until computational time uses up or user interrupts
10: return  $sol$ 

```

4. EMPIRICAL RESULTS

In this section, we give a preliminary empirical evaluation of our algorithm through comparing its performance with two other well-known approaches: the coverage-set algorithm [2] and the random restart JESP algorithm [5].

We evaluate these algorithms on problem instances with two cooperative agents where each autonomous agent has a ordered set of 40 tasks, half of which (randomly chosen) are its individual tasks and the others are its subtasks for joint tasks. The rewards for the individual tasks are evenly distributed in the range $[0, 1]$ and the rewards for the joint tasks are evenly distributed in the range $[0, 3]$. The cooperative agents try to maximize their total expected reward within 20 time steps. The state for each agent is composed of its current task and the remaining time before the deadline. At each state, each agent can choose to execute the current task or to skip it. If the agent chooses to skip, then it moves to the next task immediately. The uncertainty in the problem is in the time required to execute the tasks. In this paper, we assume that, if an agent decides to execute a task, it has 25% probability of accomplishing it within one, two, three, and four time steps respectively.

We ran experiments on a Pentium IV machine, and used *cplex* 9.1 (www.ilog.com) as our MILP solver. The empirical results shown in Figure 2 were collected from 100 randomly generated test problems. Figure 2 illustrates anytime performance of these three algorithms, in which we can see that MILP-HCS is able to find good solutions much faster than the other two algorithms.

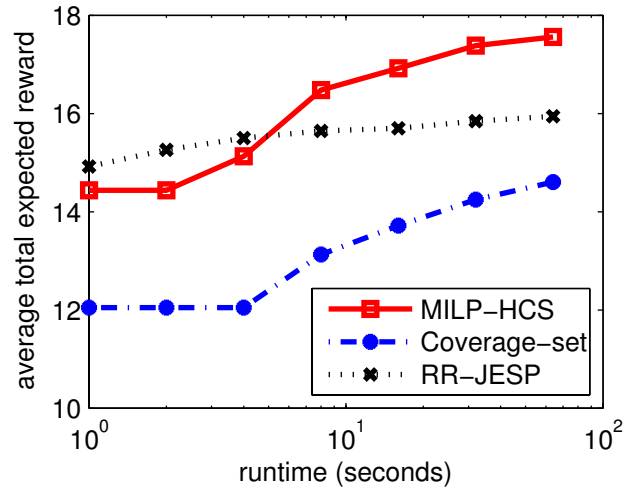


Figure 2: Anytime performance of three algorithms.

5. CONCLUSION

We have described a novel MILP-HCS algorithm for solving transition-independent Dec-MDPs, and shown through experiments that this algorithm is able to find a satisfactory solution much faster than previous approaches. This work links transition-independent Dec-MDPs with mixed integer linear programming, and lays the foundation for further efforts of exploiting established MILP techniques.

6. ACKNOWLEDGEMENTS

This material is based upon work supported in part by the DARPA/IPTO COORDINATORS program and the Air Force Research Laboratory under Contract No. FA8750-05-C-0030. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

7. REFERENCES

- [1] E. Altman. *Constrained Markov Decision Processes*. Chapman and HALL/CRC, New York, 1999.
- [2] R. Becker, S. Zilberstein, V. R. Lesser, and C. V. Goldman. Solving transition independent decentralized Markov decision processes. *J. Artif. Intell. Res. (JAIR)*, 22:423–455, 2004.
- [3] W. Cook, W. Cunningham, W. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*. John Wiley & Sons, New York, 1998.
- [4] C. V. Goldman and S. Zilberstein. Decentralized control of cooperative systems: Categorization and complexity analysis. *J. Artif. Intell. Res. (JAIR)*, 22:143–174, 2004.
- [5] R. Nair, M. Tambe, M. Yokoo, D. V. Pynadath, and S. Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *IJCAI*, pages 705–711, 2003.