

Generalized Dynamic Programming for Decentralized POMDPs

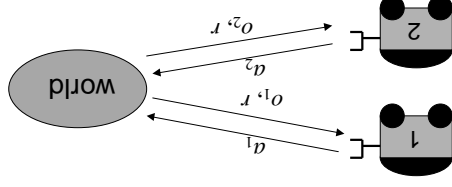
Daniel S. Bernstein

Shlomo Zilberstein

University of Massachusetts Amherst

September 5, 2003

- Multiple agents control a Markov process
- Each agent has a local view of the system
- Cooperative (single reward function)



DEC-POMDP

- Problem: Getting teams of agents to act intelligently under uncertainty about the world and each other
- Examples: networking, robotics, planetary exploration
- Mathematical Abstraction: Decentralized Partially Observable MDP (DEC-POMDP)

Introduction

- A DEC-POMDP is $\langle S, A_1, A_2, P, R, \Omega_1, \Omega_2, O \rangle$, where
 - S is a finite state set
 - A_1, A_2 are finite action sets
 - $P(s, a_1, a_2, s')$ is a transition function
 - $R(s, a_1, a_2)$ is a reward function
 - Ω_1, Ω_2 are finite observation sets
 - $O(s, o_1, o_2)$ is an observation function
- A POMDP is a DEC-POMDP with one agent

DEC-POMDP – Formal Definition

DEC-POMDP – More Definitions

- A local policy is a mapping $\delta_i : \Omega_i^* \rightarrow A_i$
- A joint policy is a pair $\langle \delta_1, \delta_2 \rangle$
- We want a joint policy that maximizes expected long-term reward

$$E \left[\sum_{t=1}^T r_t \right]$$

or

$$E \left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t \right]$$

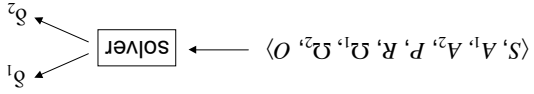
finite-horizon infinite-horizon discounted

Dynamic Programming Approach

- Lots of work on dynamic programming for POMDPs [SS73, CLZ97, KLC98, H98]
- Can we extend this approach to solve DEC-POMDPs?
- Yes, but it will not be straightforward...

The Planning Problem

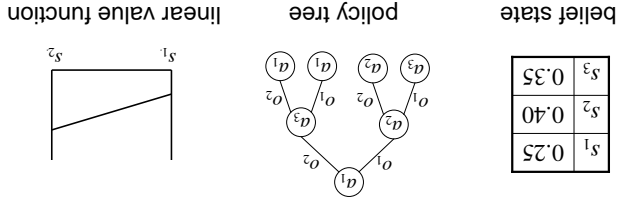
- We consider the problem of finding a solution, given a model and a centralized solver



- An important first step towards decentralized planning and reinforcement learning

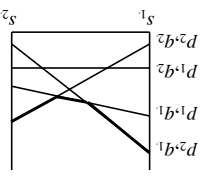
DP for Finite-Horizon POMDPs

- We'll start with some important concepts:

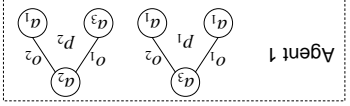


Generalizing to DEC-POMDPs

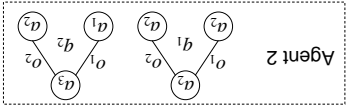
- No result like Astrom's, but we will design an algorithm anyway
- Now we maintain two sets of policy trees
- Values are assigned to *pairs* of trees



Agent 1

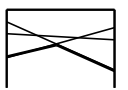


Agent 2



DP for Finite-Horizon POMDPs

- Initialize 1-step policy trees to be actions
- Repeat:
 - Evaluate each of the t -step trees
 - Prune dominated trees
 - Form an exhaustive set of $t+1$ -step trees from the remaining t -step trees
- On each step t , we are guaranteed have an optimal set of t -step trees

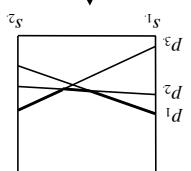


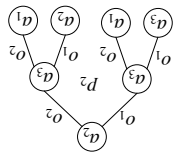
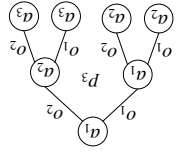
What Makes This Work?

- This can be viewed as value iteration in an equivalent belief-state MDP [A65]
- So all results for continuous MDPs apply
- Guaranteed optimality from *any* initial belief state
- Can work with value functions, forget trees
- Can easily extend to the infinite-horizon case

DP for Finite-Horizon POMDPs

The value function for a set of trees is always piecewise linear and convex (PWLC)



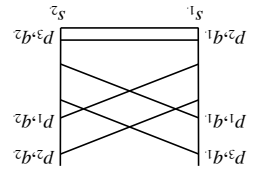



Generalized DP for DEC-POMDPs

- Initialize 1-step policy trees to be actions
- Repeat:
 - Evaluate all pairs of t -step trees from current sets
 - Prune useless trees (see next slide)
 - Form exhaustive sets of $t+1$ -step trees from remaining t -step trees
- If pruning is done correctly, then on each step t , we are guaranteed have optimal sets of t -step trees

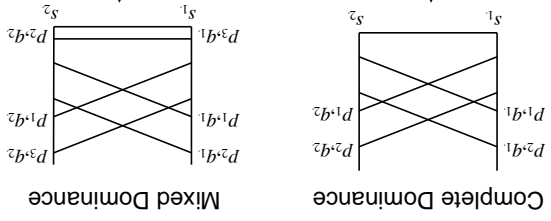
Implications

- A t -step tree that appears useless could be a subtree of a useful $t+1$ -step tree



- Now we cannot just work with value functions (must keep trees in memory)

Two Possible Pruning Rules



Works in general
Always prunes at least as much, but can lead to suboptimal solutions

Extending to Infinite-Horizon

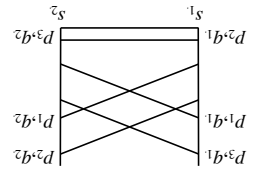
- This result makes it hard to extend infinite-horizon value iteration
- However, we have extended Hansen's policy iteration [H98], which works explicitly with finite-state controllers
- Can get convergence to optimality, but limited pruning of nodes and no convergence test

Generalized DP for DEC-POMDPs

- Initialize 1-step policy trees to be actions
- Repeat:
 - Evaluate all pairs of t -step trees from current sets
 - Prune useless trees (see next slide)
 - Form exhaustive sets of $t+1$ -step trees from remaining t -step trees
- If pruning is done correctly, then on each step t , we are guaranteed have optimal sets of t -step trees

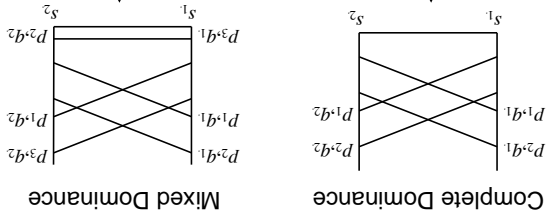
Implications

- A t -step tree that appears useless could be a subtree of a useful $t+1$ -step tree



- Now we cannot just work with value functions (must keep trees in memory)

Two Possible Pruning Rules



Works in general
Always prunes at least as much, but can lead to suboptimal solutions

Extending to Infinite-Horizon

- This result makes it hard to extend infinite-horizon value iteration
- However, we have extended Hansen's policy iteration [H98], which works explicitly with finite-state controllers
- Can get convergence to optimality, but limited pruning of nodes and no convergence test

Ongoing Work

- Other pruning strategies
- Empirical results to complement the theory
- Taking advantage of more structure (e.g. hierarchy, factored state space)
- Distributed planning and reinforcement learning

Conclusion

- We can extend dynamic programming to DEC-POMDPs, but we must be careful
- Still plenty of questions to be answered...