# A BRIEF INTRODUCTION TO DEC-POMDP

**Shuo Liu**
Computer Science
Northeastern University
shuo.liu2@northeastern.edu

## ABSTRACT

This article introduces Dec-POMDP, the computation, and planning methods.[1]

## 1 DEC-POMDP

**Definition 1.** *A Dec-POMDP is a tuple* $\langle \mathbb{I}, \mathcal{S}, \{\mathbb{A}_i\}, T, R, \{\mathbb{O}_i\}, O, \mathcal{H}, \gamma \rangle$*:*

- $\mathbb{I}$ *is a finite sets of agents,* $|\mathbb{I}| = n$*;*
- $\mathcal{S}$ *is a set of states with designated initial state distribution* $b^0$*;*
- $\mathbb{A}_i$ *is a set of actions for agent* $i$ *with* $\mathbb{A} \doteq \times_i \mathbb{A}_i$ *the set of joint actions;*
- $T$ *is the state transition probability function,* $T$*:* $\mathcal{S} \times \mathbb{A} \times \mathcal{S} \to [0, 1]$*, that specifies the probability of transitioning from state* $s \in \mathcal{S}$ *to* $s' \in \mathcal{S}$ *when the actions* $\boldsymbol{a} \in \mathbb{A}$ *are taken by agents (i.e.,* $T(s, \boldsymbol{a}, s') = P(s'|s, \boldsymbol{a})$*);*
- $R$ *is the joint reward function, where* $R$*:* $\mathcal{S} \times \mathbb{A} \to \mathbb{R}$*;*
- $\mathbb{O}_i$ *is a set of observations for each agent* $i$*, with* $\mathbb{O} \doteq \times_i \mathbb{O}_i$ *the set of joint observations;*
- $O$ *is an observation probability function,* $O$*:* $\mathbb{A} \times \mathcal{S} \times \mathbb{O}$*, that specifies the probability of seeing observation* $\boldsymbol{o}' \in \mathbb{O}$ *given the actions* $\boldsymbol{a} \in \mathbb{A}$ *are taken and state* $s' \in \mathcal{S}$ *is observed (i.e.,* $O(\boldsymbol{a}, s', \boldsymbol{o}') = P(\boldsymbol{o}'|\boldsymbol{a}, s')$*);*
- $\mathcal{H}$ *is the horizon (the number of steps until termination);*
- $\gamma$ *is the discount factor for the return.*

*A solution to a Dec-POMDP is a joint policy* $\boldsymbol{\pi} : \mathbb{H}_i \to \mathbb{A}_i, \forall i \in \mathbb{I}$ *over joint observation-action history* $\boldsymbol{h} = \{\boldsymbol{a}^0, \boldsymbol{o}^1, \cdots \boldsymbol{o}^{\mathcal{H}-1}\}$*, an optimal solution maximizes the expected return,*

$$\boldsymbol{\pi}^* = \underset{\boldsymbol{\pi}}{\operatorname{argmax}} \mathbb{E}\left[\sum_{t=0}^{\mathcal{H}-1} R(\boldsymbol{h}, \boldsymbol{\pi}(\boldsymbol{h}))\middle| b^0\right].$$

In Dec-POMDP, the Bellman recursive formulation of the history V-function is,

$$\begin{aligned} V^{\boldsymbol{\pi}}(\boldsymbol{h}) &= \sum_s P(s|b^0, \boldsymbol{h}) \left[ R(s, \boldsymbol{\pi}(\boldsymbol{h})) + \gamma \sum_{s'} P(s'|s, \boldsymbol{\pi}(\boldsymbol{h})) \sum_{\boldsymbol{o}'} P(\boldsymbol{o}'|\boldsymbol{\pi}(\boldsymbol{h}), s') V^{\boldsymbol{\pi}}(\boldsymbol{h}') \right] \\ &\equiv R(\boldsymbol{h}, \boldsymbol{\pi}) + \gamma \sum_{\boldsymbol{o}'} P(\boldsymbol{o}'|\boldsymbol{h}, \boldsymbol{\pi}) V^{\boldsymbol{\pi}}(\boldsymbol{h}'), \end{aligned} \tag{1}$$

the Bellman recursive formulation of the **history-policy** Q-function is,[2]

$$\begin{aligned} Q^{\boldsymbol{\pi}}(\boldsymbol{h}, \boldsymbol{\pi}) &= \sum_s P(s|b^0, \boldsymbol{h}) \left[ R(s, \boldsymbol{\pi}(\boldsymbol{h})) + \gamma \sum_{s'} P(s'|s, \boldsymbol{\pi}(\boldsymbol{h})) \sum_{\boldsymbol{o}'} P(\boldsymbol{o}'|\boldsymbol{\pi}(\boldsymbol{h}), s') Q^{\boldsymbol{\pi}}(\boldsymbol{h}', \boldsymbol{\pi}(\boldsymbol{h}')) \right] \\ &\equiv R(\boldsymbol{h}, \boldsymbol{\pi}) + \gamma \sum_{\boldsymbol{o}'} P(\boldsymbol{o}'|\boldsymbol{h}, \boldsymbol{\pi}) Q^{\boldsymbol{\pi}}(\boldsymbol{h}', \boldsymbol{\pi}). \end{aligned} \tag{2}$$

[1] This introduction is **heavily** based on Amato et al. (2013).

[2] The definition of the value function is pretty flexible, which can be based on various aspects, such as the value of a state, a belief state, an observation, a state history, an observation history, an action (single-step policy), a policy (action history), observation-action history, and their combinations.

## 2 SUBCLASSES

**Centralized Control** MMDP is a fully observable version of Dec-POMDP, but it does not specify decentralized control. Dec-MDP assumes that the joint observations uniquely determine the state, while agents still act with local observations. Similarly, MPOMDP does not specify whether the control is decentralized, which could have a centralized policy $\mathbb{H} \to \mathbb{A}$.

**Independent Variables** A decentralized control model might be factorized with independent local variables, e.g., transition-independence (TI) $T(s, \boldsymbol{a}, s') = \Pi_{i=1}^{n} T(s_i, a_i, s_i')$ and reward-independence (RI) $R(s, \boldsymbol{\pi}) = f_{\text{mono}}(\langle R(s_i, \pi_i)\rangle_{i=1}^{n})$. Network-distributed POMDP (ND-POMDP) represents the factored one with TI and block-RI, i.e., $R(s, \boldsymbol{\pi}) = f_{\text{mono}}(\langle R(s_{i,\mathcal{N}(i)}, \pi_{i,\mathcal{N}(i)})\rangle_{i=1}^{n})$, where $\mathcal{N}(i)$ are the neighbors of $i$.

**Complexity** The worst-case complexity of finite-horizon problems is: (by Amato et al. (2013))

| Model | Complexity |
|---|---|
| MDP | P-complete |
| MMDP (Cen-MMDP) | P-complete |
| Dec-MDP | NEXP-complete |
| Dec-MDP with TI no RI | NP-complete |
| Dec-MDP with RI no TI | NEXP-complete |
| Dec-MDP with TI and RI | P-complete |
| POMDP | PSPACE-complete |
| MPOMDP (Cen-MPOMDP) | PSPACE-complete |
| Dec-POMDP | NEXP-complete |
| ND-POMDP | NEXP-complete |

**Theorem 1.** *An MDP is P-complete in finite and infinite horizons Papadimitriou & Tsitsiklis (1987).*

**Theorem 2.** *A finite POMDP is PSPACE-complete Papadimitriou & Tsitsiklis (1987).*

**Theorem 3.** *The complexity of an infinite POMDP is undecidable Madani et al. (1999), leading to the undecidability of the infinite Dec-POMDP complexity.*

**Theorem 4.** *A finite Dec-POMDP$_{n\geqslant 2}$ is NEXP-complete, and a finite Dec-MDP$_{n\geqslant 3}$ is also NEXP-complete Bernstein et al. (2002).*

**Fact 1.** *A Dec-MDP with TI and RI can be solved independently, resulting in P-complete.*

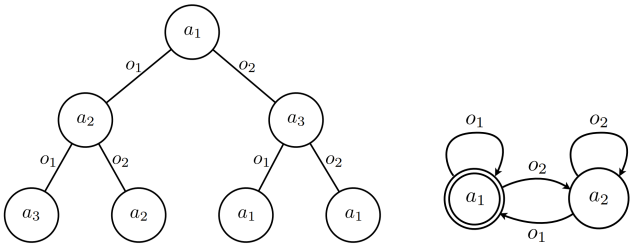
**Theorem 5.** *A Dec-MDP with TI and joint reward is NP-complete, a Dec-MDP with RI but no TI is NEXP-complete Becker et al. (2004)*

**Fact 2.** *An ND-POMDP has the same worst-case complexity as a Dec-POMDP Nair et al. (2005).*

## 3 PLANNING METHODS

### 3.1 POLICY STRUCTURE

Calculating a shared belief state in Dec-POMDP is hard, because the policy can not be recovered from the value function. The policies are normally maintained in a policy tree or FSC. Policies can be extracted by starting at the root (or initial node) and continuing to the subtree (or next node) based on observations, and can be evaluated by summing the rewards weighted by transition probability.




(a) Policy tree. (b) FSC.

### 3.2 Optimal Approaches

**Bottom-up** DP uses joint belief to find optimal solutions with policy pruning Hansen et al. (2004).

$$V^{t+1}(b^t) = \max_{a \in \mathcal{A}} \left\{ \sum_{s \in \mathcal{S}} b^t(s) \left[ R(s,a) + \sum_{o \in \mathcal{O}} \mathcal{P}(o \mid s,a) V^t(b^{t+1}) \right] \right\}. \tag{3}$$

**Algorithm 1** DP Backup with Policy Pruning in Dec-POMDP

**Input:** Depth-$t$ policy trees $Q_i^t$ and value vectors $V_i^t$ for each $i$

Perform exhaustive backups to get $Q_i^{t+1}$, and compute $V_i^{t+1}$ accordingly for each $i$
**repeat**
  Find a policy tree $q_j \in Q_i^{t+1}$ that satisfies $\exists\, v_k \in \{V_i^{t+1} \setminus v_j\}, b^{t+1} v_k \geq b^{t+1} v_j, \forall\, b^{t+1}$
  $Q_i^{t+1} \leftarrow \{Q_i^{t+1} \setminus q_j\}$, and $V_i^{t+1} \leftarrow \{V_i^{t+1} \setminus v_j\}$ accordingly
**until** no more pruning for all $i$

**Output:** Depth-$t+1$ policy trees $Q_i^{t+1}$ and value vectors $V_i^{t+1}$ for each $i$

**Top-down** The policy tree can also be built using heuristic search like MAA* Szer et al. (2005).

**Algorithm 2** MAA* Heuristic Search

**Initialize:** Joint policy tree root $\Pi \leftarrow \times_i \mathbb{A}_i$
**Input:** Depth-$t$ joint policy tree $\Pi$

Select $a^* = \text{argmax}_{a \in \Pi} F^{\mathcal{H}}(b^0, a)$ (heuristic and value)
Expand $a^*$ to $a^{\circledast}$, and $\Pi \leftarrow \{\Pi \cup a^{\circledast}\}$
**for** Each $a \in \Pi$ **do**
  **if** $F^{\mathcal{H}}(s^0, a^{\circledast}) \geqslant F^{\mathcal{H}}(s^0, a)$ **then**
    $\Pi \leftarrow \{\Pi \setminus a\}$
  **end if**
**end for**
**if** $a^*$ is fully expanded **then**
  $\Pi \leftarrow \{\Pi \setminus a^*\}$
**end if**

**Input:** Updated joint policy set $\Pi$.

### 3.3 Approximation Approaches

The algorithms below improve scalability to larger problems over optimal methods, but do not possess any bounds on solution quality.

**MBDP** Memory-bounded dynamic programming (MBDP) techniques mitigate the scalability problem of DP (which generates and evaluates all joint policy trees before pruning) by keeping a fixed number of policy trees for each agent at each step Seuken & Zilberstein (2007a). Several approaches have improved on MBDP by limiting Seuken & Zilberstein (2007b) or compressing Carlin & Zilberstein (2008) observations, replacing exhaustive backup with branch-and-bound search in the space of joint policy trees Dibangoye et al. (2009) as well as constraint optimization Kumar & Zilberstein (2010) and linear programming Wu et al. (2010) to increase the efficiency of selecting the best trees at each step.

**JESP** The joint equilibrium search for policies (JESP) Nair et al. (2003) uses alternating best response. Initial policies are generated for all agents, and then all but one is held fixed. The remaining agent can then calculate the best response (local optimum) to the fixed policies. The policy of this agent becomes fixed and the next agent calculates the best response. These best-response calculations to fixed other agent policies continue until no agent changes its policy.

## REFERENCES


Christopher Amato, Girish Chowdhary, Alborz Geramifard, N. Kemal re, and Mykel J. Kochenderfer. Decentralized control of partially observable markov decision processes. In *52nd IEEE Conference on Decision and Control*, pp. 2398–2405, 2013. doi: 10.1109/CDC.2013.6760239.

Raphen Becker, Shlomo Zilberstein, Victor Lesser, and Claudia V. Goldman. Solving transition independent decentralized markov decision processes. *Journal of Artificial Intelligence Research*, 22:423–455, 2004.

Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of Operations Research*, 27(4): 819–840, 2002.

A. Carlin and S. Zilberstein. Value-based observation compression for dec-pomdps. In *Proceedings of the Seventh International Conference on Autonomous Agents and Multiagent Systems*, 2008.

J. S. Dibangoye, A.-I. Mouaddib, and B. Chaib-draa. Point-based incremental pruning heuristic for solving finite-horizon dec-pomdps. In *Proceedings of the Eighth International Conference on Autonomous Agents and Multiagent Systems*, 2009.

Eric A Hansen, Daniel S Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI*, volume 4, pp. 709–715, 2004.

A. Kumar and S. Zilberstein. Point-based backup for decentralized pomdps: complexity and new algorithms. In *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems*, pp. 1315–1322, 2010.

Omid Madani, Steve Hanks, and Anne Condon. On the undecidability of probabilistic planning and infinite-horizon partially observable markov decision problems. AAAI '99/IAAI '99, pp. 541548. American Association for Artificial Intelligence, 1999. ISBN 0262511061.

Ranjit Nair, Milind Tambe, Makoto Yokoo, David Pynadath, and Stacy Marsella. Taming decentralized pomdps: Towards efficient policy computation for multiagent settings. In *IJCAI*, volume 3, pp. 705–711, 2003.

Ranjit Nair, Milind Tambe, Makoto Yokoo, David V. Pynadath, and Stacy Marsella. Networked distributed pomdps: A synthesis of distributed constraint optimization and pomdps. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-05)*, pp. 133–139. AAAI Press, 2005.

Christos H. Papadimitriou and John N. Tsitsiklis. The complexity of markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987. doi: 10.1287/moor.12.3.441.

Sven Seuken and Shlomo Zilberstein. Memory-bounded dynamic programming for decpomdps. *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2009–2015, 2007a.

Sven Seuken and Shlomo Zilberstein. Improved memory-bounded dynamic programming for decentralized pomdps. pp. 344–351, 2007b.

Daniel Szer, François Charpillet, and Shlomo Zilberstein. Maa*: A heuristic search algorithm for solving decentralized pomdps. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 576–590. AUAI Press, 2005.

F. Wu, S. Zilberstein, and X. Chen. Point-based policy generation for decentralized pomdps. In *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems*, pp. 1307–1314, 2010.

## A COMPLEXITY CLASSES

Assuming $c$ and $k$ are constants, $\mathcal{C}$ is a complexity class, the table shows complexity terminologies.

| | |
|---|---|
| P | the set of problems solvable in polynomial time, e.g., $O(n^k)$ |
| NP | the set of problems solvable nondeterministically in polynomial time |
| EXP | the set of problems solvable in exponential time, e.g., $O(c^{n^k})$ |
| NEXP | the set of problems solvable nondeterministically in exponential time |
| PSPACE | the set of problems solvable in polynomial space (P and NP $\subset$ PSPACE), e.g., $O(c^{n^k})$ |
| $\mathcal{C}$-hard | a problem that all problems in $\mathcal{C}$ are reducible to within polynomial time |
| $\mathcal{C}$-complete | a problem that is contained in $\mathcal{C}$ and $\mathcal{C}$-hard |