Monte-Carlo Search for an Equilibrium in Dec-POMDPs

Abstract

Decentralized partially observable Markov decision processes (Dec-POMDPs) formalize the problem of designing individual controllers for a group of collaborative agents under stochastic dynamics and partial observability. Seeking a global optimum is difficult (NEXP complete), but seeking a Nash equilibrium —each agent policy being a best response to the other agents— is more accessible, and allowed addressing infinite-horizon problems with solutions in the form of finite state controllers. In this paper, we show that this approach can be adapted to cases where only a generative model (a simulator) of the Dec-POMDP is available. This requires relying on a simulation-based POMDP solver to construct an agent's FSC node by node. A related process is used to heuristically derive initial FSCs. Experiment with benchmarks shows that MC-JESP is competitive with exisiting Dec-POMDP solvers, even better than many offline methods using explicit models.

1 INTRODUCTION

The framework of Decentralized Partially Observable Markov Decision Processes (Dec-POMDPs) allow modeling collaborative multi-agent systems, the objective being to equip them with individual policies that maximize some common performance criterion. However, solving Dec-POMDPs is challenging since the environment evolves according to all agent's actions, and each agent performs its action only based on its local action-observation histories. To ensure finding a global optimum, one thus needs to reason about all individual policies together. As a consequence of this interdependency, even for a finite-horizon Dec-POMDP, the solving process has been proven to be NEXP in the worst case [Bernstein et al., 2002], and solving an infinite-horizon Dec-POMDP is undecidable [Madani et al., 2003, Oliehoek and Amato, 2016].

Nair et al. propose an alternative approach called JESP (Joint Equilibrium-Based Search for Policies) [Nair et al., 2003], which avoids this interdependency in the solving process by searching for a Nash equilibrium, *i.e.*, each agent's policy is a best response to other agents' policies. JESP operates an iterative optimization process over each agent. In each iteration, it builds agent i's best-response policy considering other agents' policies are fixed. A Nash equilibrium is therefore guaranteed when no further improvement is possible. In JESP, each agent's policy is represented in a tree structure, which limits its usage only to finite-horizon problems. To overcome this limitation, Inf-JESP (Infinite-Horizon JESP) [You et al., 2021] extends JESP to infinite-horizon Dec-POMDPs by representing each agent policy as a finite-state controller (FSC). Two advantages of Inf-JESP are that 1. it often achieves near-global optima despite only searching for local ones, and 2. its FSCs make for interpretable policies if their size is reasonable. However, both methods require an explicit Dec-POMDP model which details the exact environment dynamics.

In this paper, we propose a new algorithm called MC-JESP (Monte-Carlo Joint Equilibrium-based Search for Policies), which is a simulation-based version of Inf-JESP. In each iteration, MC-JESP builds an agent's FSC node by node using a Monte-Carlo (POMDP) planner relying on a black-box Dec-POMDP simulator, along with the other agents' FSCs. Experiments show that MC-JESP is competitive with state-of-the-art infinite-horizon Dec-POMDP solvers based either on exact or generative models.

The structure of this paper is organized as follows: Section 2 discusses related work on solving Dec-POMDPs. Sec. 3 gives background about Dec-POMDPs, POMDPs, FSCs, and Inf-JESP. Our contributions are presented in Sec. 4, and experiments with comparisons to state-of-the-art Dec-POMDP solvers in Sec. 5. Finally, we conclude this work in Sec. 6 and discuss future perspectives.

Submitted to the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023). **To be used for reviewing only**.

2 RELATED WORK

Recently, there has been significant progress in infinite-horizon Dec-POMDP planning, and state-of-the-art methods fall into three main types. The first type of methods estimates the best parameters of finite-state controllers (FSCs) of each agent [Amato et al., 2010a], and addresses Dec-POMDPs as an inference problem via Expectation-Maximization methods [Pajarinen and Peltonen, 2011a,b, Kumar and Zilberstein, 2010, Kumar et al., 2015]. The second type consists in transforming the Dec-POMDP problem into a Markov decision process with a state space of sufficient statistics [MacDermed and Isbell, 2013, Dibangoye et al., 2014, 2016]. The third type searches for Nash equilibrium solutions, *i.e.*, each agent's policy being a best response to the other agents' policies [Nair et al., 2003, Bernstein et al., 2005, You et al., 2021].

However, for large problems or real applications, it may be challenging to represent the system's dynamics explicitly. Often, only a black-box simulator (also called a generative model) is available. Although the algorithms mentioned previously with explicit models cannot be directly applied, most state-of-the-art simulation-based methods are inspired by them. For example, Wu et al. [2013] propose to use a Monte-Carlo Expectation Maximization (MCEM) for estimating the parameters of agents' FSCs with generative models. Liu et al. [2015] improve MCEM by constructing agent FSCs using the stick-breaking prior and allowing a variable FSC size. On the other hand, similar to FB-HSVI [Dibangoye et al., 2016] (which uses explicit models), the simulation-based method oSARSA [Dibangoye and Buffet, 2018] focuses on recasting Dec-POMDPs into occupancy-state MDPs, where each occupancy-state is a sufficient statistics.

Last but not least, some multi-agent reinforcement learning (MARL) algorithms are also interested in solving Dec-POMDPs with black-box simulators. However, most of them [Sunehag et al., 2017, Rashid et al., 2018, Son et al., 2019, Rashid et al., 2020] have not been evaluated on classical Dec-POMDP benchmarks [Seuken and Zilberstein, 2007, Amato and Zilberstein, 2009]. Only a few MARL algorithms conducted experiments on such domains but were limited to finite-horizon settings [Lee and Lee, 2019], or failed to obtain state-of-the-art results [Kraemer and Banerjee, 2016].

3 BACKGROUND

3.1 DEC-POMDP

The problem of finding optimal collaborative behaviors for a group of agents under stochastic dynamics and partial observability is typically formalized as a *decentralized partially observable Markov decision process* (Dec-POMDP).

Definition 1. *A* Dec-POMDP *with* $|\mathcal{I}|$ *agents is represented as a tuple* $M \equiv \langle \mathcal{I}, \mathcal{S}, \mathcal{A}, \Omega, T, O, R, b_0, H, \gamma \rangle$*, where:* $\mathcal{I} = \{1, \ldots, |\mathcal{I}|\}$ *is a finite set of* agents*;* $\mathcal{S}$ *is a finite set of* states*;* $\mathcal{A} = \bigtimes_i \mathcal{A}^i$ *is the finite set of joint actions, with* $\mathcal{A}^i$ *the set of agent* i*'s* actions*;* $\Omega = \bigtimes_i \Omega^i$ *is the finite set of joint observations, with* Ω^i *the set of agent* i*'s* observations*;* $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ *is the* transition function*, with* $T(s, \mathbf{a}, s')$ *the probability of transiting from* s *to* s' *if* $\mathbf{a}$ *is performed;* $O : \mathcal{A} \times \mathcal{S} \times \Omega \to \mathbb{R}$ *is the* observation function*, with* $O(\mathbf{a}, s', \mathbf{o})$ *the probability of observing* $\mathbf{o}$ *if* $\mathbf{a}$ *is performed and the next state is* s'*;* $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ *is the* reward function*, with* $R(s, \mathbf{a})$ *the immediate reward for executing* $\mathbf{a}$ *in* s*;* b_0 *is the* initial probability distribution *over states;* $H \in \mathbb{N} \cup \{\infty\}$ *is the (possibly infinite)* time horizon*;* $\gamma \in [0, 1)$ *is the* discount factor *applied to future rewards.*

An agent's i action *policy* π^i maps its possible action-observation histories to actions. The objective is then to find a joint policy $\pi = \langle \pi^1, \ldots, \pi^{|\mathcal{I}|} \rangle$ that maximizes the expected discounted return from b_0:

$$V_H^\pi(b_0) \stackrel{\text{def}}{=} \mathbb{E}\left[\sum_{t=0}^{H-1} \gamma^{-t} r(S_t, A_t) \mid S_0 \sim b_0, \pi\right].$$

However, we often do not know the exact transition, observation, and reward functions for large problems or real-world applications, but may rely on a generative model (black-box simulator) G, which, given a state-action pair $\langle s, \mathbf{a} \rangle$, samples a triplet $\langle s', \mathbf{o}, r \rangle$.

3.2 POMDP

In this work, we will consider one agent i at a time, and thus end up solving a single-agent partially observable Markov decision problem (POMDP) in each iteration, *i.e.*, the particular case of a single-agent Dec-POMDP ($\mathcal{I} = \{1\}$). In a POMDP, an optimal policy π^* exists whose input is the belief state b, *i.e.*, the probability distribution over states given the current action-observation history. For finite H, the optimal value function (which allows deriving π^*) is recursively defined as:

$$V_h^*(b) = \max_a \left[r(b, a) + \gamma \sum_o Pr(o \mid b, a) V_{h-1}^*(b^{a,o}) \right],$$

where (i) $r(b, a) = \sum_s b(s) \cdot r(s, a)$; (ii) $Pr(o \mid b, a)$ depends on the dynamics; and (iii) $b^{a,o}$ is the belief updated upon performing a and perceiving o.

3.3 FINITE STATE CONTROLLERS

In POMDPs as in Dec-POMDPs, solution policies can also be sought for in the form of *finite state controllers* (FSC)

(also called *policy graphs* [Meuleau et al., 1999]), *i.e.*, automata whose transitions from one internal state to the next depend on the received observations and generate the actions to be performed.

Definition 2. *For some POMDP sets $\mathcal{A}$ and Ω, a (deterministic)* FSC *is specified by a tuple $fsc \equiv \langle N, \eta, \psi \rangle$, where:*

- *N is a finite set of nodes, with n_0 the start node;*
- *$\eta : N \times \Omega \to N$ is the node transition function; $n' = \eta(n, o)$ is the next node and observing o from node n;*
- *$\psi : N \to \mathcal{A}$ is the action-selection function of the FSC; $a = \psi(n)$ is the action triggered when in node n.*

3.4 SOLVING DEC-POMDPS BY FINDING NASH EQUILIBRIA (INFINITE-HORIZON JESP)

Inf-JESP (Infinite-Horizon JESP) [You et al., 2021] is an infinite-horizon Dec-POMDP solver, which is based on Nair et al.'s JESP [2003], but replaces the policy tree representation by a finite-state controller (FSC) for each agent's policy. This modification allows solving infinite-horizon problems rather than finite-horizon ones, and may help scaling up to larger problems. More specifically, in Inf-JESP (Algorithm 1), each iteration derives (line 7) the explicit model of a (best-response) POMDP for agent i by fixing the other agents' FSCs (index "$\neq i$") and using an extended state space $e^t \in \mathcal{E}$, *i.e.*, containing:

- s^t, the current state of the Dec-POMDP problem,
- $\mathbf{n}^t_{\neq i} \equiv \langle n^t_j \rangle_{j \neq i}$, the current nodes of other agents, and
- $\tilde{o}^t_i$, agent i's current observation.

Denoting $\eta_{\neq i}(\mathbf{n}^t_{\neq i}, \mathbf{o}^{t+1}_{\neq i}) = \langle \eta(n^t_j, \tilde{o}^{t+1}_j) \rangle_{j \neq i}$ and $\psi_{\neq i}(\mathbf{n}^t_{\neq i}) = \langle \psi_j(n^t_j) \rangle_{j \neq i}$, this leads to a valid POMDP with the following dynamics:[1]

$$
\begin{aligned}
T_e(e^t, a^t_i, e^{t+1}) &= Pr(e^{t+1} | e^t, a^t_i) \\
&= \sum_{\mathbf{o}^{t+1}_{\neq i}} T(s^t, \langle \psi_{\neq i}(\mathbf{n}^t_{\neq i}), a^t_i \rangle, s^{t+1}) \cdot \mathbf{1}_{\mathbf{n}^{t+1}_{\neq i} = \eta_{\neq i}(\mathbf{n}^t_{\neq i}, \mathbf{o}^{t+1}_{\neq i})} \\
&\quad \cdot O(s^{t+1}, \langle \psi_{\neq i}(\mathbf{n}^t_{\neq i}), a^t_i \rangle, \langle \mathbf{o}^{t+1}_{\neq i}, o^{t+1}_i \rangle), \\
O_e(a^t_i, e^{t+1}_i, o^{t+1}_i) &= Pr(o^{t+1}_i | a^t_i, e^{t+1}_i) \\
&= Pr(o^{t+1}_i | a^t_i, \langle s^{t+1}, \mathbf{n}^{t+1}_{\neq i}, \tilde{o}^{t+1}_i \rangle) = \mathbf{1}_{o^{t+1}_i = \tilde{o}^{t+1}_i}, \\
r_e(e^t, a^t_i) &= r(s^t, a^t_i, \psi_{\neq i}(\mathbf{n}^t_{\neq i})).
\end{aligned}
$$

Then, Inf-JESP solves this explicit POMDP for agent i using an ϵ-optimal offline POMDP solver (SARSOP [Kurniawati et al., 2008]) and derives an FSC fsc'_i that approximates the solution policy (cf. line 8, which does not distinguish both steps). fsc'_i is then evaluated (line 9) and retained only if it improves on i's previous FSC, fsc_i, so that Inf-JESP stops when an approximate Nash equilibrium is obtained, which is detected using a counter $\#ni$.

Algorithm 1: Inf-JESP's Local Search

```
1  [Input:] b^0: initial belief | M: Dec-POMDP model |
     fsc: initial FSCs
2  Fct LocalSearch (b_0, M, fsc)
3      v_bestL ← eval(fsc)
4      #ni ← 0 // #(iterations w/o improvement)
5      i ← 1 // Id of current agent
6      repeat // Cycle over agents
7          b^0_BR, M_BR ← getBRpomdp(b^0, M, fsc_≠i)
8          fsc'_i ← ComputeFSC (b^0_BR, M_BR)
9          v ← Eval(fsc'_i, b^0_BR, M_BR)
10         if v > v_bestL then // Keep new FSC if better
11             fsc_i ← fsc'_i
12             v_bestL ← v
13             #ni ← 0
14         else // increment #ni
15             #ni ← #ni + 1
16         i ← (i mod |I|) + 1
17     until #ni = |I| // No improvement in last cycle.
18     return ⟨fsc, v_bestL⟩
```

In practice (see [You et al., 2021] or Sec. 5.1), this search for an equilibrium often allows finding near-global optima either using some random restarts, or initial FSCs obtained through relaxing the Dec-POMDP.

4 MONTE-CARLO JESP

As Inf-JESP, we aim to find Nash equilibrium infinite-horizon solutions by iteratively building each agent's best response to other agents' fixed policies. We thus stick to representing policies as FSCs, and to using the same algorithmic scheme for the local search as presented in Alg. 1.

This requires relying on the same best-response POMDPs, *i.e.*, in particular with the same extended state $s^t_e = \langle s^t, \mathbf{n}^t_{\neq i}, o^t_i \rangle$. However, lacking an explicit Dec-POMDP model, we cannot derive explicit POMDP models. To address this issue, in MC-JESP, we propose an alternative approach that relies on *best-response generative POMDP models* (noted G_{BR}) derived from the Dec-POMDP simulator and other agents' FSCs. In the following, we discuss how to build such models, how to derive solution FSCs, how to obtain initial heuristic FSCs, and what are the properties of the resulting approach.

[1] Note: You et al. [2021] provide formulas for stochastic FSCs.

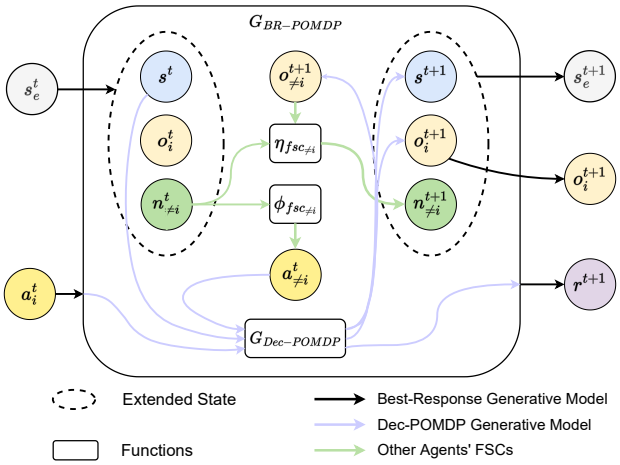


Figure 1: Structure of the best-response POMDP generative model G_{BR}, with inputs and outputs represented as: blue arrows for the Dec-POMDP simulator G; green arrows for agents $\neq i$' FSCs; and black arrows for G_{BR}.

4.1 BEST-RESPONSE GENERATIVE MODEL

A generative POMDP model G_{BR} for agent i has to sample the next extended state s_e^{t+1}, observation o_i^{t+1}, and reward r^{t+1}, given a current extended state s_e^t and action a_i^t. As illustrated in Figure 1 and as detailed in Alg. 2, this can be achieved by relying only on the Dec-POMDP simulator and the other agents' FSCs. The algorithm first decomposes the extended state, and gets other agents' actions $\mathbf{a}_{\neq i}^t$ according to action-selection functions $\psi_{\neq i} \equiv \langle \psi_j \rangle_{j \neq i}$ (lines 4 and 5). Then, in line 6, the joint action $\langle a_i^t, \mathbf{a}_{\neq i}^t \rangle$ is passed to the Dec-POMDP simulator G, which outputs the next state s^{t+1}, joint observation $\langle o_i^{t+1}, \mathbf{o}_{\neq i}^{t+1} \rangle$ and instant reward r^{t+1}. With the other agents' observations $\mathbf{o}_{\neq i}^{t+1}$, line 7 computes their next nodes $\mathbf{n}_{\neq i}^{t+1} \equiv \langle n_j^{t+1} \rangle_{j \neq i}$. In the end, we build the next extended state s_e^{t+1} and return the results (lines 8 and 9). In this algorithm, stochasticity exists only in the Dec-POMDP simulator G, the FSC functions ψ and η being deterministic.

Algorithm 2: i's Best-Response Generative Model

1 [Input:] s_e^t: extended state | a_i^t: agent i's action
2 [Parameters:] G: Dec-POMDP simulator | $fsc_{\neq i} \equiv \langle N_{\neq i}, \psi_{\neq i}, \eta_{\neq i} \rangle$: other agents' FSCs
3 **Fct** $G_{BR}(s_e^t, a_i^t, [G, fsc_{\neq i}])$
4 $\quad \langle s^t, \mathbf{n}_{\neq i}^t, o_i^t \rangle \leftarrow s_e^t$ `// extract` s_e^t`'s 3 components`
5 $\quad \mathbf{a}_{\neq i}^t \leftarrow \psi_{\neq i}(\mathbf{n}_{\neq i}^t)$ `// get action from FSC`
6 $\quad s^{t+1}, \mathbf{o}^{t+1}, r^{t+1} \leftarrow G(s^t, \mathbf{a}^t)$ `// sample transition`
7 $\quad \mathbf{n}_{\neq i}^{t+1} \leftarrow \eta(\mathbf{n}_{\neq i}^t, \mathbf{o}_{\neq i}^{t+1})$ `// evolve FSC`
8 $\quad s_e^{t+1} \leftarrow \langle s^{t+1}, \mathbf{n}_{\neq i}^{t+1}, o_i^{t+1} \rangle$ `// build` s_e^{t+1}
9 $\quad$ **return** $s_e^{t+1}, o_i^{t+1}, r^{t+1}$ `// return step results`

4.2 COMPUTING AGENT i'S FSC USING MONTE-CARLO METHODS

In the previous section, we demonstrate how to build the best-response generative model G_{BR} for agent i considering others' fixed FSCs. However, unlike in Inf-JESP, state-of-the-art point-based POMDP solvers (see, *e.g.*, [Pineau et al., 2003, Smith and Simmons, 2004, Kurniawati et al., 2008]) require exact models, and thus can not be used in MC-JESP. We thus rely on a simulation-based solver, *i.e.*, POMCP [Silver and Veness, 2010], which is an online algorithm, *i.e.*, it focuses on returning the best action for the current belief. Therefore, the question is how to use a simulator (G_{BR}) and an online simulation-based solver to obtain agent i's FSC. To answer it, we propose an algorithm that 1. uses this Monte-Carlo planner (POMCP) to compute the best action for a given FSC node, which is labeled by a unique belief; and 2. expands reachable beliefs, *i.e.*, creates new FSC nodes using computed actions to gradually build a complete FSC. Moreover, to control the computational cost, we explicitly bound the FSC size with a given parameter $N_{max\text{-}fsc} \in \mathbb{N}$.

In the proposed algorithm, each FSC node is attached to 1. an approximate belief b (with at least $N_{min\text{-}part}$ particles), 2. a preferred action a_i, and 3. a weight w that estimates the probability to reach that node at least once during execution. As detailed in Algorithm 3, this information is first gathered for initial belief b_0 by calling POMCP to get agent i's best action a_i^0 in line 4. Line 5 then creates a start node n_0 with b_{BR}^0, a_i^0, and a weight $w = 1$. This start node is added to the FSC under construction (N) and an open list (L).

Now, while L is not empty, its node n with largest weight w is popped out (line 9), so as to first develop the nodes that may have the highest impact on the value at the root. Expanding it requires mapping each observation o_i feasible when performing $n.a_i$ from $n.b$ to a particle set. This is achieved through sampling by **getNxtBeliefs** until each feasible o_i (according to the samples) is attached to at least $N_{min\text{-}part}$ particles (line 29), which returns a set Ω_i' of feasible observations, and a mapping B_i' from these observations to particle sets. Then, for each individual observation o_i, the algorithm needs to create a transition to an appropriate node, which may already exist or needs to be created, as explained in the following. If o_i is assumed impossible when performing $n.a_i$ in $n.b$ ($o_i \notin n.\Omega_i$), then a self-loop is added (line 13).[2] Otherwise, line 15 gets the belief b_{BR}' attached to o_i, and line 16 computes an associated weight w'. If (i) a belief ϵ-close to b_{BR}' (in 1-norm) exists in N, or (ii) the FSC has reached its size limit $N_{max\text{-}fsc}$ (line 17), then we take as next node n' the one in the FSC minimizing $\|b_{BR}' - n'.b\|_1$ and update its weight (lines 18–19). Otherwise a next node n' is created using an action selected by

[2]Note that o_i could become feasible due to future changes in other agents' FSCs.

POMCP (lines 21–22), and added to both N and L (lines 22–24). In line 25, whatever the origin of n', an edge $n \to n'$ is created in the FSC with a label o_i.

Note that, for a fixed $N_{max\text{-}fsc}$ value, a small ϵ may prevent from representing long trajectories, while a large ϵ may induce excessive node merging.

Algorithm 3: Compute agent i's FSC

1 [Input:] b^0_{BR}: G_{BR}'s initial (extended) belief state | G_{BR}: best response generative model for agent i
2 [Parameters:] $N_{max\text{-}fsc}$: max FSC size for agent i | $N_{min\text{-}part}$: min number of particles in each belief | ϵ: min. distance between beliefs
3 **Fct ComputeFSC**(b^0_{BR}, G_{BR})
4 $a^0_i \leftarrow POMCP(b^0_{BR}, G_{BR})$
5 $n_0 \leftarrow node(b^0_{BR}, a^0_i, w = 1)$
6 $N \leftarrow \{n_0\}$ `// init FSC & open list`
7 $L[w] \leftarrow n_0$ `// (sorted by ↘ weight)`
8 **while** $|L| > 0$ **do**
9 $n \leftarrow L.popfront()$
10 $\Omega'_i, B'_i \leftarrow$ **getNxtBeliefs**($n.b, G_{BR}, n.a_i$)
11 **for** $o_i \in \Omega_i$ **do** `// For each obs. of i:`
12 **if** $o_i \notin \Omega'_i$ **then** `// o_i unexpected:`
13 $\eta(n, o_i) \leftarrow n$ `// add self-loop`
14 **else** `// Else: create next node`
15 $b'_{BR} \leftarrow B'_i[o_i]$
16 $w' \leftarrow \frac{|B'_i[o_i]|}{|B'_i|} \cdot n.w$
17 **if** $(b'_{BR} \in N(\epsilon)) \vee (|N| = N_{max\text{-}fsc})$ **then**
`// Similar node exists in FSC or FSC full?`
`// Yes: Merge with closest node in FSC`
18 $n' \leftarrow N.findClosest(b'_{BR})$
19 $n'.w \leftarrow n'.w + w'$
20 **else** `// No: Add new node`
21 $a'_i \leftarrow POMCP(b'_{BR}, G_{BR})$
22 $n' \leftarrow node(b'_{BR}, a'_i, w')$
23 $N \leftarrow N \cup \{n'\}$
24 $L[w'] \leftarrow n'$
25 $\eta(n, o_i) \leftarrow n'$ `// Add transition to FSC.`
26 **Fct getNxtBeliefs** (b^t_{BR}, G_{BR}, a^t_i)
27 $\Omega^{t+1}_i \leftarrow \emptyset$
28 $B^{t+1}_i \leftarrow \emptyset$
29 **repeat**
30 $e^t \sim b^t_{BR}$
31 $\langle e^{t+1}, o^{t+1}_i, r^{t+1} \rangle \sim G_{BR}(e^t, a^t_i)$
32 **if** $o^{t+1}_i \notin \Omega^{t+1}_i$ **then**
33 $\Omega^{t+1}_i \leftarrow \Omega^{t+1}_i \cup \{o^{t+1}_i\}$
34 $B^{t+1}[o^{t+1}_i] \leftarrow B^{t+1}[o^{t+1}_i] \cup \{e^{t+1}\}$
35 **until** *Timeout()* $\vee$ $(MinBeliefParticles(B^{t+1}) > N_{min\text{-}part})$
36 **return** Ω^{t+1}_i, B^{t+1}_i

4.3 HEURISTIC INITIALIZATION

Although MC-JESP monotonically improves the value of the joint policy at each iteration, random initializations often lead to poor local optima. To benefit from a heuristic initialization that allows finding good solutions quickly and reliably, we adapt Inf-JESP's heuristics as we adapted the computation of an agent's FSC in the previous section: using particle sets as beliefs, calling a simulation-based solver, and bounding the number of nodes. In addition, to derive the next belief, we marginalize over possible joint observations $o_{\neq i}$, rather than reasoning on them separately as You et al. [2021] did (*e.g.*, considering only the most probable one).

This heuristic initialization assumes that 1. agent i's decisions are made as if all agents where sharing their observations, thus acting as a single agent; this means making decisions (picking joint actions $\mathbf{a}$) by solving a Multi-agent POMDP (MPOMDP) [Pynadath and Tambe, 2002] relaxation of the original Dec-POMDP, which can be done here with an (online) simulation-based POMDP solver; and 2. agent i's belief b over the hidden state is updated assuming (a) that the other agents ($\neq i$) also act according to the computed MPOMDP policy at b, but (b) using only i's observation, o_i, while marginalizing over others' observations ($o_{\neq i}$, which are actually not known to i at execution time) to ignore them. This *one-sided-observation belief update* is computed as follows:

$$b^{\mathbf{a},o_i}(s') \stackrel{\text{def}}{=} Pr(s'|o_i, \mathbf{a}, b) = \frac{Pr(s', o_i, \mathbf{a}, b)}{Pr(o_i, \mathbf{a}, b)}$$
$$= \frac{\sum_{\mathbf{o}_{\neq i}} O(\langle o_i, \mathbf{o}_{\neq i} \rangle | \mathbf{a}, s') \sum_s T(s, \mathbf{a}, s') b(s)}{\sum_{s', \mathbf{o}_{\neq i}} O(\langle o_i, \mathbf{o}_{\neq i} \rangle | \mathbf{a}, s') \sum_s T(s, \mathbf{a}, s') b(s)}.$$

Following this idea, we derive the FSC heuristic initialization process for agent i detailed in Alg. 4 which, as shown in red, differs from Alg. 3 in two aspects:

- the Dec-POMDP simulator G is used as an MPOMDP simulator for POMCP to get the best joint action with a given belief (lines 3 and 21); and
- in line 10's **getNxtBeliefs** function, the next estimated beliefs are obtained by repeatedly sampling transitions using the computed joint action $\langle n.a_i, n.\mathbf{a}_{\neq i} \rangle$ (and Dec-POMDP simulator G), and collecting particle sets for each encountered individual observation o_i, ignoring $\mathbf{o}_{\neq i}$, which is equivalent to a marginalization.

4.4 OBSERVATIONS

With an increasing time budget, POMCP asymptotically converges to optimal decisions. By (i) increasing POMCP's time budget to infinity, (ii) increasing $N_{min\text{-}part.}$ and $N_{max\text{-}fsc}$ to infinity, and (iii) setting $\epsilon = 0$, each iteration of the local search would thus return the best response (possibly infinite)

Algorithm 4: Build a heuristic FSC for agent i

1 [Input:] b^0: initial state distribution | i: agent index
2 [Parameters:] G: Dec-POMDP simulator | $N_{max\text{-}fsc}$: max. FSC size | ϵ: min. distance between beliefs
3 $\langle a_i^0, \mathbf{a}_{\neq i}^0 \rangle \leftarrow POMCP(b^0, G)$
4 $n_0 \leftarrow node(b^0, a_i^0, \mathbf{a}_{\neq i}^0, w = 1)$
5 $N \leftarrow \{n_0\}$ `// init FSC & open list`
6 $L[w] \leftarrow n_0$
7 **while** $|L| > 0$ **do** `// loop over open nodes`
8 $\quad L.sort()$
9 $\quad n \leftarrow L.popfront()$
10 $\quad \Omega_i', B_i' \leftarrow$ **getNxtBeliefs**$(n.b, G, \langle n.a_i, n.\mathbf{a}_{\neq i} \rangle)$
11 $\quad$ **for** $o_i \in \Omega$ **do** `// For each obs. of i:`
12 $\quad\quad$ **if** $o_i \notin \Omega_i'$ **then** `// o_i unexpected: add self-loop`
13 $\quad\quad\quad \eta(n, o_i) \leftarrow n$
14 $\quad\quad$ **else** `// Else: create next node:`
15 $\quad\quad\quad b' \leftarrow B_i'[o_i]$
16 $\quad\quad\quad w' \leftarrow n.w * \frac{|B_i'[o_i]|}{|B_i'|}$
17 $\quad\quad\quad$ **if** $(b' \in N(\epsilon)) \vee (|N| = N_{max\text{-}fsc})$ **then**
$\quad\quad\quad$ `// Similar node exists in FSC or FSC full?`
$\quad\quad\quad$ `// Yes: Merge with closest node in FSC`
18 $\quad\quad\quad\quad n' \leftarrow N.findClosest(b')$
19 $\quad\quad\quad\quad n'.w \leftarrow n'.w + w'$
20 $\quad\quad\quad$ **else** `// No: Add new node`
21 $\quad\quad\quad\quad \langle a_i', \mathbf{a}_{\neq i}' \rangle \leftarrow POMCP(b', G)$
22 $\quad\quad\quad\quad n' \leftarrow node(b', a_i', \mathbf{a}_{\neq i}', w')$
23 $\quad\quad\quad\quad N \leftarrow N \cup \{n'\}$
24 $\quad\quad\quad\quad L[w] \leftarrow n'$
25 $\quad\quad\quad \eta(n, o_i) \leftarrow n'$

FSC. As a consequence, assuming also an exact evaluation of FSCs, the local search would be guaranteed to find a Nash equilibrium.

In practice, we only obtain approximate Nash equilibria. Also, due to randomization in POMCP and in FSC evaluations through simulations, restarts of the full process lead to different search trajectories, but always stop in finitely many iterations with probability 1. The next section looks at the results obtained in practice through experiments.

5 EXPERIMENTS

We evaluate our contributions on five state-of-the-art Dec-POMDP benchmarks (*cf.* `http://masplan.org/problem_domains`): Decentralized Tiger [Nair et al., 2003], Recycling Robots [Amato et al., 2012], Meeting in a 3×3 grid [Amato et al., 2009], Cooperative Box Pushing [Seuken and Zilberstein, 2007], Mars Rover [Amato and Zilberstein, 2009]. We compare MC-JESP with state-of-the-art Dec-POMDP solvers relying on either: *explicit models:* (which benefit from more information) FB-HSVI [Dibangoye et al., 2016], Peri [Pajarinen and Peltonen, 2011b], PeriEM [Pajarinen and Peltonen, 2011b], PBVI-BB [MacDermed and Isbell, 2013], MealyNLP [Amato et al., 2010b] and Inf-JESP; or *generative models:* MCEM [Wu et al., 2013], Dec-SBPR [Liu et al., 2015] and oSARSA [Dibangoye and Buffet, 2018].

For MC-JESP, 1. POMCP is used as our simulation-based POMDP planner with a timeout of 1 s; 2. we consider three different maximum FSC sizes: 10, 30, and 50, respectively; 3. the threshold distance between beliefs is set to $\epsilon = 0.1$; and 4. FSC evaluations (line 9 of Alg. 1) use 10^6 simulations that stop when $\gamma^t < 10^{-4}$. For MC-JESP and Inf-JESP's empirical results, having access to the exact model in each benchmark problem, we use Hansen's [1998] policy evaluation for FSCs applied to a best-response POMDP. The experiments with MC-JESP were conducted on a laptop with a 2.3 GHz i9 CPU. The source code is in the supplementary material.

5.1 COMPARISON WITH STATE-OF-THE-ART ALGORITHMS

Table 1 presents the results for the 5 problems, the solvers being ordered from best to worse value. Among x restarts of MC-JESP, the best value is reported in MC-JESP(M-x), and the average value in MC-JESP(M-1_x) (to look at the benefit of restarting). For Inf-JESP, we report the best values among its 3 possible initializations [You et al., 2021]. For MC-JESP, we report the best values over the 3 possible max. FSC sizes. The columns provide: • (*Alg.*) the different algorithms at hand, with a $*$ exponent for those who rely on an explicit model; • (*FSC size*) the final FSC size (for Inf-JESPs and MC-JESP); • (*Iterations*) the number of iterations required to converge (for Inf-JESPs and MC-JESP); • (*Time*) the running time; • (*Value*) the final value (lower bounds for Inf-JESPs and MC-JESP, the true value being at most 0.01 higher). In terms of final value achieved, MC-JESP(M-1_{20}) finds good solutions in all cases except DecTiger (which is a small but difficult coordination problem), and MC-JESP(M-20) obtains results very close to FB-HSVI's near-optimal solutions, which rely on an explicit Dec-POMDP model, for all benchmark problems. MC-JESP even sometimes achieves better results than some other explicit model-based algorithms. Also, compared with other simulation-based methods, it dominates in large problems (Box-pushing and Mars Rovers), while other simulation-based methods fail.

However, compared with the explicit model-based algorithms, MC-JESP requires more solving time. For example, in large problems such as Mars Rovers, MC-JESP takes 349 s on average to solve the task, while Inf-JESP takes 122 s. But this is not surprising since MC-JESP only uses a black-box simulator. A key question is how to determine

Table 1: Comparison of different algorithms in terms of final FSC size, number of iterations, time, and value, on 5 infinite-horizon benchmarks with $\gamma = 0.9$ for all domains. The solvers are listed in a decreasing order of value.

Alg.	FSC size	Iterations	Time (s)	Value
DecTiger ($\lvert\mathcal{I}\rvert = 2, \lvert\mathcal{S}\rvert = 2, \lvert\mathcal{A}^i\rvert = 3, \lvert\mathcal{Z}^i\rvert = 2$)				
FB-HSVI*			154	13.45
Peri*			220	13.45
INF-JESP*	6×6	27	213	13.44
MC-JESP(M-20)	30×30	5	5620	13.44
PeriEM*			6450	9.42
oSARSA				-0.20
MC-JESP(M-1_{20})	24×25	4	281	-2.33
Dec-SBPR			96	-18.63
MCEM			20	-32.31
Recycling ($\lvert\mathcal{I}\rvert = 2, \lvert\mathcal{S}\rvert = 4, \lvert\mathcal{A}^i\rvert = 3, \lvert\mathcal{Z}^i\rvert = 2$)				
FB-HSVI*			3	31.93
MC-JESP(M-20)	17×17	3	5260	31.92
Peri*			77	31.84
PeriEM*			272	31.80
INF-JESP*	2×2	3	3	31.62
Dec-SBPR			147	31.26
MC-JESP(M-1_{20})	19×20	4	263	30.74
Grid3*3 ($\lvert\mathcal{I}\rvert = 2, \lvert\mathcal{S}\rvert = 81, \lvert\mathcal{A}^i\rvert = 5, \lvert\mathcal{Z}^i\rvert = 9$)				
INF-JESP*	8×10	3	2	5.81
MC-JESP(M-20)	50×50	4	11 900	5.81
FB-HSVI*			67	5.80
MC-JESP(M-1_{20})	50×50	6	595	5.80
Peri*			9714	4.64
Box-pushing ($\lvert\mathcal{I}\rvert = 2, \lvert\mathcal{S}\rvert = 100, \lvert\mathcal{A}^i\rvert = 4, \lvert\mathcal{Z}^i\rvert = 5$)				
FB-HSVI*			1715	224.43
MC-JESP(M-20)	50×50	6	9740	223.84
MC-JESP(M-1_{20})	50×50	5	487	220.94
INF-JESP*	250×408	6	963	220.25
Peri*			5675	148.65
oSARSA				144.57
PeriEM*			7164	106.65
Dec-SBPR			290	77.65
Mars Rover ($\lvert\mathcal{I}\rvert = 2, \lvert\mathcal{S}\rvert = 256, \lvert\mathcal{A}^i\rvert = 6, \lvert\mathcal{Z}^i\rvert = 8$)				
FB-HSVI*			74	26.94
INF-JESP*	125×183	6	122	26.91
MC-JESP(M-20)	50×50	5	6980	26.45
MC-JESP(M-1_{20})	50×50	3	349	25.89
Peri*			6088	24.13
Dec-SBPR			1286	20.62
PeriEM*			7132	18.13

whether restarting can be beneficial.

5.2 A CLOSER LOOK AT MC-JESP'S BEHAVIOR

We study MC-JESP's performance with three different maximum FSC sizes in Fig. 2 (red for 10, green for 30, and blue for 50). Right parts present the distribution over final values of MC-JESP with 20 restarts. In the five problems at hand, MC-JESP with max FSC size 50 (blue) has distributions more concentrated on good values than others, and most values are close to FB-HSVI's ones (thus, near-optimal values). These distributions show that few restarts are needed to reach good solutions with high probability if we give large enough FSC sizes.

The left parts of Fig. 2 present the evolution of the values during each iteration of MC-JESP with the three maximum FSC sizes. The average is computed over all runs, even if they have already converged. This figure first shows that MC-JESP monotonically increases during each run, and most runs converge to good local optima in a few iterations. Second, we observe that, for large problems (Box-Pushing and Mars Rovers), there are already significant drops from MC-JESP in the first iteration with an FSC size limit decreasing from 50 to 10. This indicates that, for large problems, we must give large enough FSC size limits, while this is not necessary for small problems.

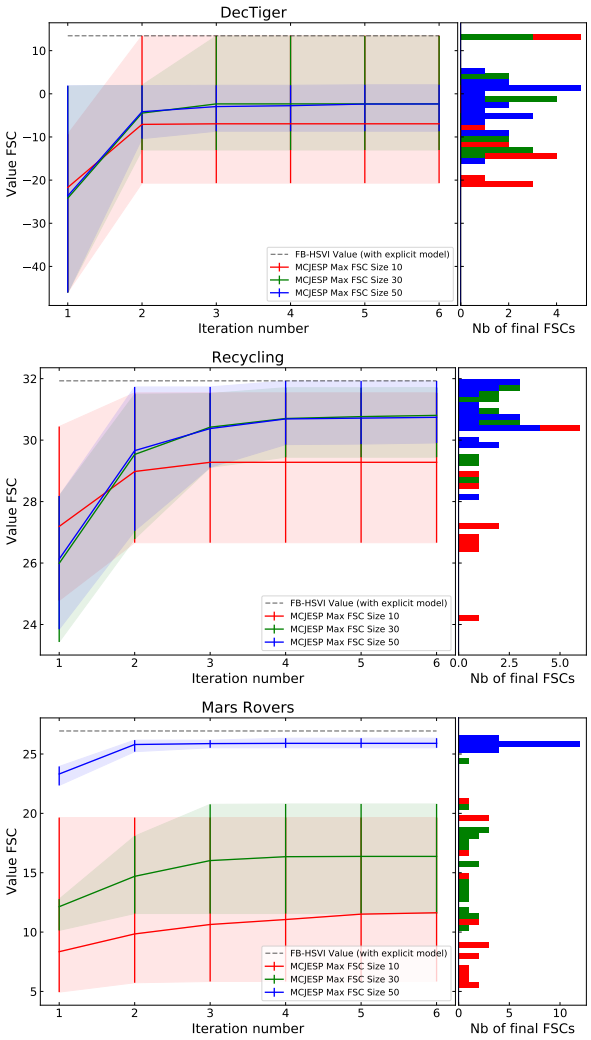
Last but not least, in Dec-Tiger, although some restarts of MC-JESP end with optimal values, we observe that the average value is still relatively low compared with FB-HSVI. Therefore, we conducted another experiment to investigate the impact of different POMCP timeouts (note that there is a fixed timeout of 1 s for the experiments illustrated in Figure 2). To that end, we limit the FSC size in each iteration to at most 50 nodes, and we test MC-JESP with five POMCP timeouts (1 s, 5 s, 10 s, 20 s, and 30 s). The distribution of final values is shown in Fig. 3. We observe that the average value increases and the variability shrinks when we give more time to POMCP. However, it also indicates that, when we increase the time budget, we have a lower chance of getting "lucky" good values.

6 CONCLUSION

In this work, based on Inf-JESP, we propose a novel infinite-horizon Dec-POMDP solver called MC-JESP, which only requires a black-box Dec-POMDP simulator, and returns FSCs, *i.e.*, representations that can make for interpretable policies. We describe how to obtain a best-response generative model (the simulator of the POMDP faced by some agent i assuming known FSCs for other agents), and the process to extract an FSC for each agent. Moreover, a heuristic initialization method for MC-JESP is also provided.

Through experiments, we prove that MC-JESP preserves Inf-JESP's competitive results (though at the cost of an increased computation time), performing better than many explicit model-based algorithms, and outperforming other simulation-based algorithms in most cases. Because it seeks Nash equilibria, this approach could better scale up to large problems than approaches directly seeking global optima.

Several improvements of MC-JESP could be envisioned, such as: 1. robustly comparing FSCs fsc_i and fsc'_i, while minimizing computation time through hypothesis testing; 2. if using large FSCs, using space partitioning (*e.g.*, k-d



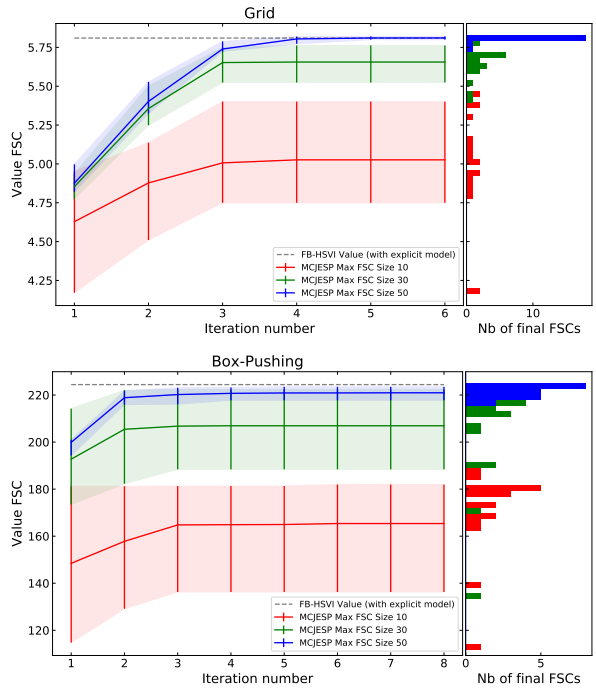


Figure 2: Values of the joint policy for the Dec-Tiger, Grid, Recycling, Box-Pushing, and Mars Rover problems (from top to bottom). The left part of each figure presents the evolution (during a run) of the value of the joint policy at each iteration of MC-JESP(1_{20}) (avg + 10th and 90th percentiles) with different bounded FSC sizes (10, 30, and 50, respectively). The dashed line represents FB-HSVI's final value. The right part presents the value distribution after convergence of MC-JESP(1_{20}).

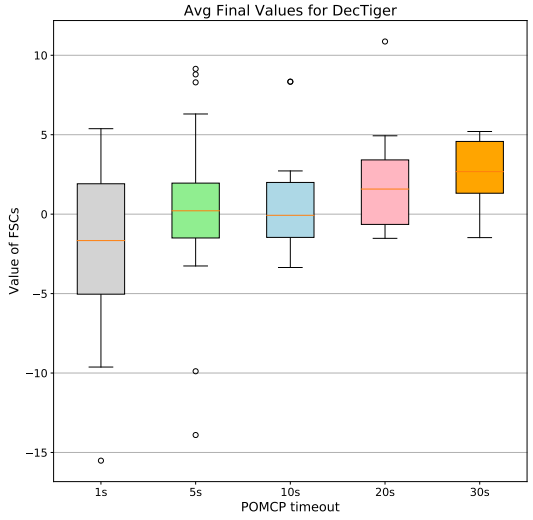


Figure 3: Values of the joint policy for the Dec-Tiger problem for different POMCP timeout values.

trees [Bentley, 1975] or cover trees [Beygelzimer et al., 2006]) to speed up the search for nearest nodes; and 3. re-using POMCP trees from one node to the next, or to initialize **getNxtBeliefs**, although doing so may significantly increase memory usage.

Also, preliminary experiments show that MC-JESP works on a continuous-state meet-in-a-grid problem, the main issue being to replace the distance between sets of discrete particles (i.e., just comparing two vectors representing discrete distributions) by a distance over continuous particles (which requires taking the distance between states into account). For future works, we plan to extend MC-JESP to problems with continuous actions and observations. This would require not only relying on algorithms such as Sunberg and Kochenderfer's POMCPOW [2018], but also, more importantly, deriving FSCs that can handle continuous observations.

References

Christopher Amato and Shlomo Zilberstein. Achieving goals in decentralized POMDPs. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*, pages 593–600, 2009.

Christopher Amato, Jilles Steeve Dibangoye, and Shlomo Zilberstein. Incremental policy generation for finite-horizon DEC-POMDPs. In *Nineteenth International Conference on Automated Planning and Scheduling*, 2009.

Christopher Amato, Daniel S. Bernstein, and Shlomo Zilberstein. Optimizing fixed-size stochastic controllers for POMDPs and decentralized POMDPs. *Journal of Autonomous Agents and Multi-Agent Systems*, 21(3), 2010a. doi: 10.1007/s10458-009-9103-z.

Christopher Amato, Blai Bonet, and Shlomo Zilberstein. Finite-state controllers based on Mealy machines for centralized and decentralized POMDPs. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010b.

Christopher Amato, Daniel Bernstein, and Shlomo Zilberstein. Optimizing memory-bounded controllers for decentralized POMDPs. *Proceedings of the Conference on Uncertainty in Artificial Intelligence, (UAI-07)*, 06 2012.

Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, sep 1975. ISSN 0001-0782. doi: 10.1145/361002.361007. URL https://doi.org/10.1145/361002.361007.

Daniel S. Bernstein, Eric A. Hansen, and Shlomo Zilberstein. Bounded policy iteration for decentralized POMDPs. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 1287–1292. Morgan Kaufmann Publishers Inc., 2005.

D.S. Bernstein, R. Givan, N. Immerman, and Shlomo Zilberstein. The complexity of decentralized control of Markov decision processes. *Math. of Operations Research*, 27(4), 2002.

Alina Beygelzimer, Sham Kakade, and John Langford. Cover trees for nearest neighbor. In *Proceedings of the Twenty-Third International Conference on Machine Learning*, pages 97–104, 2006.

Jilles Dibangoye and Olivier Buffet. Learning to act in decentralized partially observable MDPs. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1233–1242. PMLR, 10–15 Jul 2018.

Jilles Dibangoye, Chris Amato, Olivier Buffet, and François Charpillet. Optimally solving Dec-POMDPs as continuous-state MDPs. *Journal of Artificial Intelligence Research*, 55, 2016.

Jilles S Dibangoye, Olivier Buffet, and François Charpillet. Error-bounded approximations for infinite-horizon discounted decentralized POMDPs. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 338–353. Springer, 2014.

Eric Hansen. An improved policy iteration algorithm for partially observable MDPs. In *Advances in Neural Information Processing Systems 10*, 1998.

Landon Kraemer and Bikramjit Banerjee. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 190:82–94, 2016. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2016.01.031. URL https://www.sciencedirect.com/science/article/pii/S0925231216000783.

Akshat Kumar and Shlomo Zilberstein. Anytime planning for decentralized POMDPs using expectation maximization. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 294–301, July 2010. URL https://dl.acm.org/doi/abs/10.5555/3023549.3023584.

Akshat Kumar, Shlomo Zilberstein, and Marc Toussaint. Probabilistic inference techniques for scalable multiagent decision making. *Journal of Artificial Intelligence Research*, 53, 2015. doi: 10.1613/jair.4649.

Akshat Kumar, Hala Mostafa, and Shlomo Zilberstein. Dual formulations for optimizing Dec-POMDP controllers. In *Proceedings of the Twenty-Sixth International Conference on Automated Planning and Scheduling (ICAPS)*, pages 202–210. AAAI Press, 2016. URL http://www.aaai.org/ocs/index.php/ICAPS/ICAPS16/paper/view/13124.

Hanna Kurniawati, David Hsu, and Wee Sun Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Proc. of Robotics: Science and Systems (RSS)*, 2008.

Hyun-Rok Lee and Taesik Lee. Improved cooperative multi-agent reinforcement learning algorithm augmented by mixing demonstrations from centralized policy. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1089–1098, 2019.

Miao Liu, Christopher Amato, Xuejun Liao, Lawrence Carin, and Jonathan P How. Stick-breaking policy learning in Dec-POMDPs. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, 2015.

Liam C. MacDermed and Charles Isbell. Point based value iteration with optimal belief compression for Dec-POMDPs. In *Advances in Neural Information Processing Systems 26*, 2013.

Omid Madani, Steve Hanks, and Anne Condon. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artificial Intelligence*, 147(1): 5–34, 2003. ISSN 0004-3702. Planning with Uncertainty and Incomplete Information.

N. Meuleau, K.-E. Kim, L.P. Kaelbling, and A.R. Cassandra. Solving POMDPs by searching the space of finite policies. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, 1999.

Ranjit Nair, Milind Tambe, Makoto Yokoo, David Pynadath, and Stacy Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, volume 3, pages 705–711. Citeseer, 2003.

Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. Springer Publishing Company, Incorporated, 1st edition, 2016. ISBN 3319289276.

Joni Pajarinen and Jaakko Peltonen. Efficient planning for factored infinite-horizon DEC-POMDPs. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, 2011a.

Joni Pajarinen and Jaakko Peltonen. Periodic finite state controllers for efficient POMDP and DEC-POMDP planning. In *Advances in Neural Information Processing Systems 24*, 2011b.

Joelle Pineau, Geoff Gordon, and Sebastian Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 1025–1030. Morgan Kaufmann Publishers Inc., 2003.

David V. Pynadath and Milind Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, 16, 2002. ISSN 1076-9757. doi: 10.1613/jair.1024.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International conference on machine learning*, pages 4295–4304. PMLR, 2018.

Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. Weighted Qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. *Advances in neural information processing systems*, 33:10199–10210, 2020.

Sven Seuken and Shlomo Zilberstein. Improved memory-bounded dynamic programming for decentralized POMDPs. *ArXiv*, abs/1206.5295, 2007.

David Silver and Joel Veness. Monte-Carlo planning in large POMDPs. In *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010.

Trey Smith and Reid Simmons. Heuristic search value iteration for pomdps. In *Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence*, pages 520–527. AUAI Press, 2004. ISBN 0974903906.

Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International conference on machine learning*, pages 5887–5896. PMLR, 2019.

Zachary N. Sunberg and Mykel J. Kochenderfer. Online algorithms for POMDPs with continuous state, action, and observation spaces. In *Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling*, 2018.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

Feng Wu, Shlomo Zilberstein, and Nicholas R. Jennings. Monte-Carlo expectation maximization for decentralized POMDPs. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, page 397–403. AAAI Press, 2013. ISBN 9781577356332.

Yang You, Vincent Thomas, Francis Colas, and Olivier Buffet. Solving infinite-horizon Dec-POMDPs using finite state controllers within JESP. In *Proceedings of the 33rd IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, 2021.