

Scalable Solution Methods for Dec-POMDPs with Deterministic Dynamics

Yang You¹, Alex Schutz², Zhikun Li², Bruno Lacerda², Robert Skilton¹, Nick Hawes²,

¹ United Kingdom Atomic Energy Authority
² Oxford Robotics Institute, University of Oxford

Abstract

Many high-level multi-agent planning problems, such as multi-robot navigation and path planning, can be modeled with deterministic actions and observations. In this work, we focus on such domains and introduce the class of Deterministic Decentralized POMDPs (Det-Dec-POMDPs)—a subclass of Dec-POMDPs with deterministic transitions and observations given the state and joint actions. We then propose a practical solver, *Iterative Deterministic POMDP Planning* (IDPP), based on the classic Joint Equilibrium Search for Policies framework, specifically optimized to handle large-scale Det-Dec-POMDPs that existing Dec-POMDP solvers cannot handle efficiently.

1 Introduction

Decentralized partially observable Markov decision processes (Dec-POMDPs) are widely used to model multi-agent decision-making under uncertainty and partial observability, where each agent acts based solely on its own action-observation history. While highly expressive, Dec-POMDPs are difficult to solve. Even for finite horizons, solving Dec-POMDPs optimally is NEXP-complete (Bernstein et al. 2002). To reduce this complexity, Besse and Chaib-Draa introduced the Quasi-Deterministic Dec-POMDP (QDet-Dec-POMDP) (Besse and Chaib-Draa 2009), which assumes deterministic transitions but retains stochastic observations. While this quasi-deterministic structure simplifies some aspects of Dec-POMDPs, the stochastic observations can still significantly hinder scalability.

Motivated by the observation that in many real-world robotic mission planning scenarios, high-level decision-making often involves both deterministic action outcomes and effectively deterministic observations, our first contribution is to propose a further simplification of existing models: the Deterministic Decentralized POMDP (Det-Dec-POMDP). In a Det-Dec-POMDP, uncertainty exists only in the initial state distribution, while both the transition and observation models are fully deterministic. This model can be seen as a natural extension of deterministic POMDPs (Det-POMDPs) (Littman 1996) to the multi-agent setting. Leveraging this fully deterministic structure, our second contribution introduces a practical solver called *Iterative Deterministic*

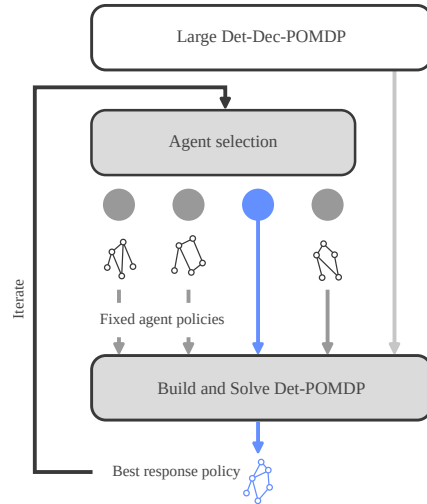


Figure 1: The approach of IDPP for solving large Det-Dec-POMDPs. The joint policy is decomposed into individual agent policies, which are initialized using a heuristic. Policies are improved using an iterative best-response process.

POMDP Planning (IDPP) that specifically optimized for solving large Det-Dec-POMDPs. IDPP improves upon prior JESP frameworks (Nair et al. 2003; You et al. 2021, 2023) by exploiting deterministic system dynamics. In each iteration, it invokes an efficient Det-POMDP planner (Schutz et al. 2025) to compute each agent’s best-response policy, ultimately converging to a Nash equilibrium policy set, as shown in Figure 1. Although simple, this practical enhancement significantly improves scalability and enables efficient planning in large Det-Dec-POMDPs that existing Dec-POMDP solvers struggle to handle. Moreover, to facilitate future research on algorithm scalability, we introduce two scalable Det-Dec-POMDP benchmarks that scale to millions of states and thousands of observations per agent.

2 Related Work

Applications with Deterministic POMDPs. Recent advances in efficient algorithms have driven significant progress in Deterministic POMDP applications. One particularly useful class involves environments where structural elements

are initially uncertain but can be deterministically observed during execution. For example, a robot may have a prior over an object’s location, or a waiter robot may choose to bring the most likely item before confirming a request. These problems can be made deterministic at the action and observation level through appropriate abstractions, such as checking whether a door is open or detecting an object using a reliable classifier. Such examples appear in robot forest path planning (Schutz et al. 2025) and robots navigation under centralized control (Stadler, Banfi, and Roy 2023). Beyond robotics, areas such as circuit synthesis, sorting networks, and communication protocols can also be modeled as Det-POMDPs (Bonet 2009).

Solving General Dec-POMDPs. State-of-the-art Dec-POMDP planning methods broadly fall into three categories. The first frames Dec-POMDPs as inference problems, estimating optimal parameters for each agent’s policy—often represented as finite-state controllers (FSCs) (Amato, Bernstein, and Zilberstein 2010; Pajarinen and Peltonen 2011a,b; Kumar and Zilberstein 2012; Kumar, Zilberstein, and Tous-saint 2015; Song, Liao, and Carin 2016). These approaches are well-suited for infinite-horizon problems and can produce compact policies, but the underlying non-convex optimization often suffers from poor local optima, limiting solution quality. The second transforms Dec-POMDPs into centralized sequential decision problems by constructing sufficient statistic spaces such as occupancy or information states (Szer, Charpillet, and Zilberstein 2012; MacDermed and Isbell 2013; Dibangoye et al. 2016). This enables the use of powerful POMDP solvers to compute optimal joint policies, which are then decomposed into decentralized agent policies. However, the exponential growth of the statistic space may limit scalability. The third category relaxes global optimality by seeking Nash equilibrium solutions (Nair et al. 2003; Bernstein, Hansen, and Zilberstein 2005; Bernstein et al. 2009; You et al. 2021, 2023), where each agent’s policy is a best response to fixed policies of others. This approach often scales better in infinite-horizon problems by reducing the problem to sequential single-agent POMDPs, at the cost of lacking optimality guarantees.

Another major line of work is multi-agent reinforcement learning (MARL), which tackles Dec-POMDPs through a learning perspective (Sunehag et al. 2017; Rashid et al. 2018; Yu et al. 2021). To handle partial observability, they often incorporate recurrent architectures (Hochreiter and Schmidhuber 1997) that maintain internal memory of action-observation histories. However, these methods often require extensive training time and struggle with sample inefficiency, especially when the agents’ reward signals are sparse.

Motivation and Our Contribution. We observe that many applications of single-agent Det-POMDPs can naturally extend to decentralized multi-agent settings, forming Det-Dec-POMDPs—for example, generalizing single-robot navigation to multi-robot navigation in a forest. However, unlike the single-agent case, scalable solvers for large Det-Dec-POMDPs remain underdeveloped, and general Dec-POMDP methods often struggle with such domains. This lack of efficient methods may, in turn, limit the practical adoption of Det-Dec-POMDPs. In this work, beyond formalizing Det-

Dec-POMDPs, we propose a simple yet practical solution method to enable future applications.

3 Background

Partially Observable Decision Models

A Partially Observable Markov Decision Process (POMDP) models a decision-making problem where the agent cannot directly observe the true underlying state. A POMDP is formally defined as a tuple $\langle \mathcal{S}, \mathcal{A}, \Omega, \mathcal{T}, \mathcal{O}, \mathcal{R}, \gamma, b_0 \rangle$, where \mathcal{S} denotes the set of states, \mathcal{A} the set of actions, and Ω the set of observations. The transition function $\mathcal{T}(s, a, s') = \Pr(s' | s, a)$ specifies the probability of reaching state s' after taking action a in state s , while the observation function $\mathcal{O}(a, s', o) = \Pr(o | s', a)$ gives the probability of observing o after arriving at state s' via action a . The reward function $\mathcal{R}(s, a)$ defines the immediate reward received for taking action a in state s , and $\gamma \in [0, 1)$ is the discount factor that models the agent’s preference for immediate rewards over future ones. The initial belief b_0 denotes the initial state distribution. In POMDPs, at each timestep, the agent updates a belief (a probability distribution over \mathcal{S}) based on the actions taken and observations received. The goal of solving a POMDP is to find a policy that maps beliefs to actions in order to maximize expected discounted rewards over time.

A Decentralized POMDP (Dec-POMDP) extends POMDPs to cooperative multi-agent settings. In a Dec-POMDP, multiple agents jointly control the environment, each making decisions based on their local action-observations. Agents aim to coordinate implicitly through their policies to maximize a shared cumulative reward. Planning in Dec-POMDPs is significantly more challenging than POMDPs due to this decentralized feature and the exponential growth of joint policy spaces.

Recent work has studied subclasses of POMDPs and Dec-POMDPs with deterministic or partially deterministic dynamics (Besse and Chaib-Draa 2009; Bonet 2009). Deterministic POMDPs (Det-POMDPs) have deterministic state transition and observation functions, while Quasi-Deterministic POMDPs (QDET-Dec-POMDPs) feature deterministic transitions but stochastic observations. However, deterministic Dec-POMDPs (Det-Dec-POMDPs), which naturally extend Det-POMDPs to multi-agent settings, have not been specifically studied to the best of our knowledge.

Finite-State Controllers

A Finite-State Controller (FSC) is a compact representation of a policy for agents in POMDPs and Dec-POMDPs. Instead of mapping full histories or beliefs to actions, an FSC encodes a policy as a finite automaton defined by a tuple $(\mathcal{N}, \psi, \eta, n^0)$, where: \mathcal{N} is a finite set of controller nodes (internal states); $\psi : \mathcal{N} \rightarrow \mathcal{A}$ is the action selection function; $\eta : \mathcal{N} \times \mathcal{O} \rightarrow \mathcal{N}$ is the node transition function based on observations, and $n^0 \in \mathcal{N}$ is the initial node. At each time step, the agent selects an action *deterministically* $a = \psi(n)$ based on its current node $n \in \mathcal{N}$, and upon receiving an observation $o \in \mathcal{O}$, it transitions to a new node $n' = \eta(n, o)$ *deterministically*.

FSCs are widely used in infinite-horizon planning (Bai et al. 2011; Lim, Sun, and Hsu 2011; You et al. 2021, 2023)

due to their ability to represent policies compactly and operate without tracking the full belief state or history. Note that, there are also *stochastic* FSCs where action selection and node transition function are modeled with probability distributions. In this paper we stick to the deterministic version of the FSC for simplicity without sacrificing optimality (Oliehoek, Spaan, and Vlassis 2008).

Finding Nash-Equilibrium Solutions

Joint Equilibrium-based Search for Policies (JESP) approaches (Nair et al. 2003; You et al. 2021, 2023) aim to find Nash Equilibrium solutions by iteratively computing one agent’s best-response policy while fixing the policies of all other agents. All JESP methods share a common algorithmic structure, given in Algorithm 1:

Algorithm 1: General JESP Framework

Input: Initial policies for all agents

Output: A Nash equilibrium policy set

- 1 **while** policies have not converged **do**
 - 2 Select the current optimizing agent i ;
 - 3 Fix other agents’ policies $\pi_{\neq i}$ and construct agent i ’s best-response model POMDP_{BR, i} ;
 - 4 Solve POMDP_{BR, i} and update agent i ’s policy π_i ;
-

Infinite-Horizon JESP (InfJESP) (You et al. 2021) extends JESP to infinite-horizon Dec-POMDPs by representing each agent’s policy as a finite-state controller and constructing the best-response POMDP accordingly. Each agent’s best-response model POMDP_{BR, i} is solved using SARSOP (Kurniawati, Hsu, and Lee 2008), enabling planning over infinite horizons. In InfJESP, agent i ’s POMDP_{BR, i} uses an extended state space $e^t \in \mathcal{E}$ containing: • s^t , the current environment state; • $n_{\neq i}^t = \langle n_j^t \rangle_{j \neq i}$, the current nodes of other agents’ FSCs; • \tilde{o}_i^t , agent i ’s current observation. This extended state enables defining a valid best-response POMDP with the following dynamics:

$$\begin{aligned} \mathcal{T}_e(e^t, a_i^t, e^{t+1}) &= Pr(e^{t+1} | e^t, a_i^t) \\ &= \sum_{o_{\neq i}^{t+1}} T(s^t, \langle \psi_{\neq i}(n_{\neq i}^t), a_i^t \rangle, s^{t+1}) \cdot \mathbf{1}_{n_{\neq i}^{t+1} = \eta_{\neq i}(n_{\neq i}^t, o_{\neq i}^{t+1})} \\ &\quad \cdot O(s^{t+1}, \langle \psi_{\neq i}(n_{\neq i}^t), a_i^t \rangle, \langle o_{\neq i}^{t+1}, o_i^{t+1} \rangle), \\ \mathcal{O}_e(a_i^t, e_i^{t+1}, o_i^{t+1}) &= Pr(o_i^{t+1} | a_i^t, e_i^{t+1}) \\ &= Pr(o_i^{t+1} | a_i^t, \langle s^{t+1}, n_{\neq i}^{t+1}, \tilde{o}_i^{t+1} \rangle) = \mathbf{1}_{o_i^{t+1} = \tilde{o}_i^{t+1}}, \\ r_e(e^t, a_i^t) &= r(s^t, a_i^t, \psi_{\neq i}(n_{\neq i}^t)). \end{aligned}$$

where:

- $\psi_{\neq i}(n_{\neq i}^t) = \langle \psi_j(n_j^t) \rangle_{j \neq i}$ denotes the other agents’ action selections.
- $\eta_{\neq i}(n_{\neq i}^t, o_{\neq i}^{t+1}) = \langle \eta_j(n_j^t, \tilde{o}_j^{t+1}) \rangle_{j \neq i}$ denotes the other agents’ FSC node transitions.

Note that, in this expression, the extended state e^t explicitly includes agent i ’s current observation \tilde{o}_i^t , which results in a deterministic observation function \mathcal{O}_e when constructing

agent i ’s best-response POMDP for any Dec-POMDP. In MCJESP (You et al. 2023), the best-response model POMDP_{BR, i} is represented implicitly via a generative model $G_{\text{POMDP}_{\text{BR},i}}$, rather than explicitly enumerating all transitions. Agent i ’s FSC policy is then optimized through Monte Carlo search (Silver and Veness 2010); specifically, each FSC node is associated with a belief, and MCJESP uses POMCP to compute the best action for that node, updating agent i ’s FSC in a node-by-node manner. This allowing MCJESP to scale to larger Dec-POMDPs by avoiding the computational overhead associated with explicit dynamics representation.

4 Deterministic Dec-POMDPs

This section formally defines the deterministic Dec-POMDP (Det-Dec-POMDP), an extension of the single-agent Det-POMDP to decentralized multi-agent settings.

Definition 1. A *Det-Dec-POMDP* is a tuple $\langle \mathcal{I}, \mathcal{S}, \mathcal{A}, \Omega, \mathcal{T}, \mathcal{O}, \mathcal{R}, \gamma, b_0 \rangle$, where: • \mathcal{I} is the finite set of agents, with $i \in \mathcal{I}$; • \mathcal{S} is the set of states, $s \in \mathcal{S}$; • $\mathcal{A} = \times_{i \in \mathcal{I}} \mathcal{A}_i$ is the set of joint actions, where \mathcal{A}_i is the action set of agent i ; • $\Omega = \times_{i \in \mathcal{I}} \Omega_i$ is the set of joint observations, where Ω_i is the observation set of agent i ; • $\mathcal{T}(s, a, s') : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \{0, 1\}$ is the deterministic transition function, mapping a state and joint action to a unique next state; • $\mathcal{O}(a, s', o) : \mathcal{A} \times \mathcal{S} \times \Omega \rightarrow \{0, 1\}$ is the deterministic observation function, mapping a joint action and next state to a unique joint observation; • $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the immediate reward function; • $\gamma \in (0, 1)$ is the discount factor for future rewards; • $b_0 \in \Delta(\mathcal{S})$ is the initial belief, i.e., a probability distribution over initial states.

Each agent i selects its actions according to a local policy $\pi_i : (\mathcal{A}_i \times \Omega_i)^* \rightarrow \mathcal{A}_i$, mapping its local action-observation history to an action. In a Det-Dec-POMDP, uncertainty arises only from the initial state; thereafter, the system evolves deterministically according to \mathcal{T} and \mathcal{O} .

In a single-agent Det-POMDP, the agent’s belief about the true state becomes increasingly concentrated (the support of the belief monotonically decreases) as it gathers deterministic observations over time. At each step, the agent can rule out states that are inconsistent with its action-observation history, gradually refining its belief until it converges to the true state. Therefore, by exploiting both the deterministic dynamics and the concentrating nature of the belief, one can develop highly efficient planning methods for solving Det-POMDPs (Schutz et al. 2025). However, even under deterministic dynamics, solving a Det-Dec-POMDP remains significantly more challenging than solving a Det-POMDP because each agent must reason not only about its own observations but also anticipate all other agents’ possible histories and their induced behaviors, leading to a combinatorial explosion in the joint history and policy spaces.

5 Iterative Deterministic POMDP Planning

One major scalability bottleneck in existing state-of-the-art Det-POMDP planning methods is the requirement to construct and reason over sufficient statistics (Szer, Charpillet, and Zilberstein 2012; Dibangoye et al. 2016), such as distribution over these joint histories (also known as an occupancy

state), to compute each agent’s optimal actions. This involves evaluating an exponentially growing number of joint histories, which quickly becomes intractable as the problem size increases, especially in domains with thousands of observations per agent, even under finite-horizon settings. Importantly, this issue persists in Det-Dec-POMDPs. Although one might expect determinism to simplify planning, it does not alleviate the exponential growth in the joint history space. This is because initial state uncertainty still leads to many possible observation sequences for each agent, resulting in a large number of possible joint histories that must be considered during computation—especially when a long horizon is required to complete the task.

In this work, we aim to efficiently address large-scale Det-Dec-POMDPs. To tackle the scalability bottleneck, we propose a practical, optimized variant of the JESP approach (You et al. 2021, 2023), which finds Nash equilibrium solutions while leveraging recent advances in solving deterministic POMDPs efficiently.

Best-Response Det-POMDP for Agent i

We follow the same theoretical framework as InfJESP (You et al. 2021), where each agent i ’s decision-making problem is formulated as a best-response POMDP when the FSC policies of the other agents are fixed. Specifically, agent i makes decisions by reasoning over a belief defined on an extended state space \mathcal{E} , where each $e^t \in \mathcal{E}$ is a tuple $\langle s^t, n_{\neq i}^t, \tilde{o}_i^t \rangle$.

Theorem 1. *In a Det-Dec-POMDP, when the FSC policies of all agents except agent i are fixed, the best-response model for agent i , denoted $\text{POMDP}_{\text{BR},i}$, is a Det-POMDP.*

Proof. When the policies $\pi_{\neq i}$ of the other agents are fixed, the transition function of agent i ’s best-response model $\text{POMDP}_{\text{BR},i}$ is given by:

$$\mathcal{T}_e(e^t, a_i^t, e^{t+1}) = \sum_{o_{\neq i}^{t+1}} \mathcal{T}(s^t, \langle \psi_{\neq i}(n_{\neq i}^t, a_i^t), s^{t+1} \rangle \cdot \mathbf{1}_{n_{\neq i}^{t+1} = \eta_{\neq i}(n_{\neq i}^t, o_{\neq i}^{t+1})} \cdot \mathcal{O}(s^{t+1}, \langle \psi_{\neq i}(n_{\neq i}^t, a_i^t), \langle o_{\neq i}^{t+1}, o_i^{t+1} \rangle \rangle)$$

In this function, the actions of the other agents are deterministically chosen by $\psi_{\neq i}(n_{\neq i}^t)$, so the first part, \mathcal{T} , corresponds to the transition function of the Det-Dec-POMDP, which deterministically maps the current state and joint action to the next state. The second part, the transition of the other agents’ FSC nodes, is also deterministic by the FSC definition. The third part, $\mathcal{O}(s^{t+1}, \langle \psi_{\neq i}(n_{\neq i}^t, a_i^t), \langle o_{\neq i}^{t+1}, o_i^{t+1} \rangle \rangle)$, is deterministic because, in a Det-Dec-POMDP, \mathcal{O} maps the next state and joint action to a unique joint observation. This implies that there exists only one possible $o_{\neq i}^{t+1}$, and thus the summation can be eliminated. Therefore, the entire transition function \mathcal{T}_e is deterministic, i.e., $\mathcal{T}_e(e^t, a_i^t, e^{t+1}) \in \{0, 1\}$ and equals 1 for exactly one e^{t+1} . Moreover, \mathcal{O}_e is already defined as a deterministic observation function in $\text{POMDP}_{\text{BR},i}$ ’s formulation. Thus, $\text{POMDP}_{\text{BR},i}$ is a deterministic POMDP with deterministic dynamics. \square

This reduction allows solving a Det-Dec-POMDP as a sequence of Det-POMDPs, where each agent computes a

best response to fixed policies of others, leveraging efficient Det-POMDP solvers that exploit deterministic structure.

Main Algorithm

The main procedure of *Iterative Deterministic POMDP Planning* (IDPP) for solving Det-Dec-POMDPs is shown in Algorithm 2. This procedure is adapted from the (MC)JESP scheme, with key modifications highlighted in blue. The process begins with a heuristic initialization step (line 1), which is described in detail in Section 5. It then enters an iterative best-response loop where agents update their policies until convergence. In each iteration, IDPP constructs a best-response deterministic POMDP for the selected agent i (line 6) and solves it efficiently using Det-MCVI (Schutz et al. 2025) (line 7), a solver specifically tailored for deterministic POMDPs. This design leverages the structural determinism of Det-Dec-POMDPs to significantly improve computational efficiency. Although the adaptation appears simple, it results in a powerful and scalable framework for solving large-scale multi-agent decision-making problems modeled as Det-Dec-POMDPs.

Algorithm 2: Main Algorithm for Solving Det-Dec-POMDP

Input: Deterministic Dec-POMDP model G
Output: Nash equilibrium policy set $\{\pi_i\}_{i \in \mathcal{I}}$

- 1 $\{\pi_i\} \leftarrow \text{HeuristicInitFSCs}(G)$;
- 2 $V \leftarrow \emptyset$;
- 3 **while** V not converged **do**
- 4 $i \leftarrow \text{GetNextAgent}()$;
- 5 $\pi_{\neq i} \leftarrow \text{FixOthersPolicies}(i, \{\pi_j\})$;
- 6 $G_{\text{BR},i} \leftarrow \text{BuildBRDetPOMDP}(G, \pi_{\neq i})$;
- 7 $\pi_i^* \leftarrow \text{SolveDetPOMDP}(G_{\text{BR},i})$;
- 8 $\pi_i \leftarrow \pi_i^*$;
- 9 $V \leftarrow \text{Evaluate}(G, \{\pi_j\})$;
- 10 **return** $\{\pi_j\}$

Heuristic Initialization

In JESP-style methods (Nair et al. 2003; You et al. 2021, 2023), initializing agents with non-myopic policies can significantly reduce iterations, especially when coordination is essential. Prior work (e.g., InfJESP and MCJESP) uses centralized heuristics that plan joint policies over the joint observation space, which becomes inefficient in large domains.

We propose a new heuristic that avoids joint observations by planning over each agent’s local observation space. When other agents’ policies are fixed, agent i plans using a Det-POMDP with extended state $e^t = \langle s^t, n_{\neq i}^t, \tilde{o}_i^t \rangle$ (Section 5). At initialization, we replace $n_{\neq i}^t$ with a default MDP policy π_{MDP} that maps states to actions: $a_{\neq i}^t \leftarrow \pi_{\text{MDP}}(s^t)$. This yields a deterministic model $\text{POMDP}_{\text{init},i}$ with state $e^t = \langle s^t, \pi_{\text{MDP}}, \tilde{o}_i^t \rangle$, which agent i uses to compute its initial policy. Since all components evolve deterministically, $\text{POMDP}_{\text{init},i}$ is itself a Det-POMDP. The full initialization procedure is given in Algorithm 3, and π_{MDP} is easily computed using standard value iteration.

Algorithm 3: HeuristicInitFSCs

Input: The Det-Dec-POMDP model G **Output:** All agents' initial policies

- 1 $\pi_{\text{MDP}} \leftarrow \text{GetDefaultPolicyMDP}(G)$;
 - 2 $\{\pi_{\text{init}}\} \leftarrow \emptyset$;
 - 3 **for** each agent $i \in \mathcal{I}$ **do**
 - 4 $G_{\text{POMDP}_{\text{init},i}} \leftarrow \text{BuildInitDetPOMDP}(G, i, \pi_{\text{MDP}})$;
 - 5 $\pi_{\text{init},i} \leftarrow \text{SolveDetPOMDP}(G_{\text{POMDP}_{\text{init},i}})$;
 - 6 $\{\pi_{\text{init}}\} \leftarrow \{\pi_{\text{init}}\} \cup \pi_{\text{init},i}$;
 - 7 **return** $\{\pi_{\text{init}}\}$
-

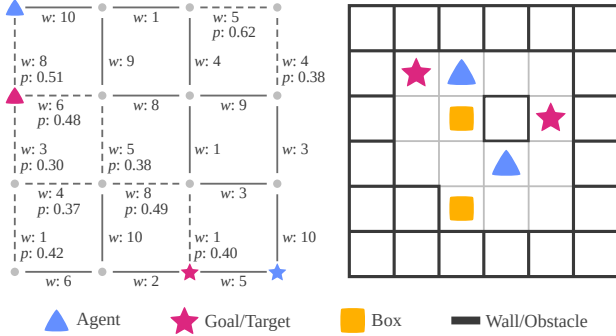


Figure 2: LEFT: An instance of MACTP $\langle 4, 2, 10 \rangle$. RIGHT: An instance of Collecting $\langle 4, 4, 2, 2 \rangle$.

It is important to note that this heuristic initialization does not guarantee optimality. Each agent assumes others follow a fixed MDP policy, which may yield suboptimal initial policies in tightly coordinated scenarios. However, since these serve only as starting points, the subsequent IDPP process (Algorithm 2) iteratively refines them via best-response updates, eventually converging to a Nash equilibrium.

6 Experiments

Experiment Setting. In this work, we introduce two Det-Dec-POMDP benchmarks: *Multi-Agent Canadian Traveler Problem* (MACTP) and *Collecting* as in Figure 2. Each is configurable with parameters such as grid size and number of agents. For each problem instance, we perform 10 runs for each algorithm to compute the average return and time used. Each run’s joint policies are evaluated with 10^5 episodes from a random starting state sampled from b_0 . All environments are initialized with the same random seed to ensure each algorithm is solving the exact same instance.

Compared Algorithms. We compare the proposed method, IDPP, against the following baselines:

- **InfJESP** (You et al. 2021) and **MCJESP** (You et al. 2023): State-of-the-art Dec-POMDP planners that compute Nash equilibrium solutions, representing infinite-horizon policies with finite-state controllers (FSCs). We use the official implementations provided by the authors.
- **MAA*** (Szer, Charpillat, and Zilberstein 2012): A heuristic search-based Dec-POMDP solver capable of comput-

ing optimal solutions for finite-horizon settings. We use the implementation from the MADP toolbox (Oliehoek et al. 2017).

- **Deterministic POMDP Heuristic:** The baseline described in Section 5, which decomposes the problem into independent Det-POMDPs for each agent, then solves each using DetMCVI (Schutz et al. 2025) to produce individual FSCs.
- **IQL** (Tan 1993) and **MAPPO** (Yu et al. 2021): Two popular MARL algorithms. Both are implemented in Python using PyTorch and employ LSTM-based policies to address partial observability.

All planning methods are implemented in C++, and a time limit of 10,000 seconds is imposed to evaluate their efficiency. MARL methods adopt a fixed training budget of 10,000 episodes (each episode lasting up to 100 steps), rather than enforcing a time constraint. All methods are evaluated based on their *discounted accumulated rewards*.

Multi-Agent Canadian Traveler Problem

The Multi-Agent Canadian Traveler Problem (MACTP) is generated by the tuple $\langle N, n_a, n_e \rangle$, where N is the grid size, n_a is the number of agents, and n_e is the number of stochastic edges. In MACTP, each edge i ’s weight d_i is randomly initialized from $\{1, \dots, 10\}$. Stochastic edges may be blocked or unblocked with a given probability; the edge’s true status can only be observed when an agent reaches one of its incident vertices. Stochastic edges and their blockage probabilities are also randomly initialized. Goal vertices are assigned among the final $N^2 - n_e$ to N^2 nodes for each agent. Each agent takes actions from \mathcal{A} , which is the action space consisting of $\{\uparrow, \rightarrow, \downarrow, \leftarrow, \circ\}$, where \circ denotes a wait action. The state space \mathcal{S} combines the agents’ positions and the edge states, resulting in $|\mathcal{S}| = (N^2)^{n_a} \times 2^{n_e}$. At each time step, each agent observes the traversability of nearby edges and the locations of other agents. A reward of 500 is given when each agent reaches its corresponding goal position, and a cost equal to the edge distance d_i is incurred for each successful movement action. Therefore, the objective is for the agents to explore the environment and identify the shortest traversable paths to their goals.

In the MACTP domain, initial-state uncertainty creates a wide range of possible configurations. For instance, with two edges having blockage probabilities 0.3 and 0.4, the initial belief spans four blockage states (e.g., $Pr(\text{both blocked}) = 0.12$, $Pr(\text{edge1 blocked, edge2 not}) = 0.18$, etc.). Once blockage probabilities and edge weights are generated (using a fixed seed), they are known to the agents—only the blockage configuration remains uncertain. Although agents do not explicitly share observations, they infer environmental information by observing each other’s movements: an agent moving toward a vertex suggests traversable edges, while stopping or circling near an edge indicates possible blockage. This observational cooperation enables agents to indirectly gather knowledge and adapt their behavior accordingly.

Collecting Problem

The Collecting Problem is a Det-Dec-POMDP generated by the tuple $\langle H, W, n_a, n_b \rangle$, where a team of n_a agents operates on a structured grid of size $(H+2) \times (W+2)$, surrounded by untraversable wall cells. Within the interior $H \times W$ region, n_b obstacle cells and n_b goal cells are randomly placed, but their positions are fixed and known to all agents at the start of the problem. The agents must cooperatively pick up and deliver n_b indistinguishable boxes to the n_b designated goal squares. Each successful delivery yields a reward of +100. Agents choose actions from the set $\mathcal{A} := \{\uparrow, \rightarrow, \downarrow, \leftarrow, \odot\}$. Boxes are picked up or dropped automatically when an agent occupies the same cell. To resolve potential conflicts, agent actions are executed in a fixed sequential order, making the problem asymmetric. Each agent receives an observation o_i comprising the 3×3 grid centered on its location, where each cell may indicate a wall, an empty space, a box, an agent, or a goal. While the placement of obstacles and goal locations is fixed and known to agents, there is uncertainty regarding the initial state of the system. Specifically, agents’ starting positions and the initial locations of the boxes are unknown, which introduces uncertainty in the initial belief. This uncertainty is resolved as agents gather more information through their observations during executions. The approximate state space size is:

$$|\mathcal{S}| \approx (C \times 2)^{n_a} \times \binom{C}{n_b},$$

where $C = H \times W - n_b$ is the number of free (e.g., non-wall, non-obstacle) cells, and the factor 2 captures the binary carrying status of each agent.

In this problem, each agent can carry at most one box, and all boxes are indistinguishable, with any box deliverable to any goal location. Therefore, agents must coordinate to avoid redundant deliveries and resolve path conflicts, while considering the uncertainty from the initial state.

Results

We first evaluate the optimal Dec-POMDP algorithm MAA*. As a finite-horizon method, we test MAA* with horizons of 10 and 20. However, we are unable to obtain a valid policy even for horizon 10 due to MAA*’s excessive memory consumption. We track the memory usage of MAA* over the first 60 seconds and compare it with other planning methods on the problem MACTP(3, 2, 5). As shown in Figure 3, MAA* quickly exhausts memory and the program terminates shortly afterward. In contrast, IDPP maintains consistently low memory usage, as it avoids reasoning over every possible joint history. This suggests that building the sufficient statistics to compute optimal policies may be impractical in large (Det-)Dec-POMDPs.

The performance of other algorithms is summarized in Table 1. InfJESP outperforms existing methods on smaller problems (e.g., MACTP(3, 2, 5) and Collecting(4, 3, 2, 2)) by using the exact model and SARSOP to optimally compute each agent’s best response. However, it does not scale well to larger instances. MCJESP, InfJESP’s successor, achieves better scalability while maintaining good performance by

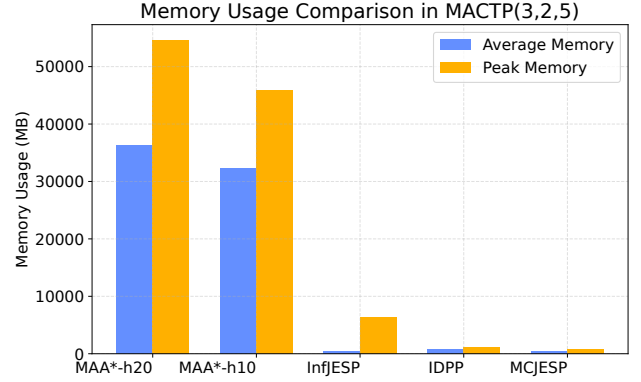


Figure 3: Comparison of average and peak memory usage (in MB) for different planning algorithms on the MACTP(3, 2, 5) problem. For MAA*, memory usage is recorded over the first 60 seconds, after which the program was terminated due to memory exhaustion in both horizon-10 and horizon-20 settings. For all other planners, memory usage is tracked until the problem is successfully solved. Algorithms are presented in descending order of peak memory usage.

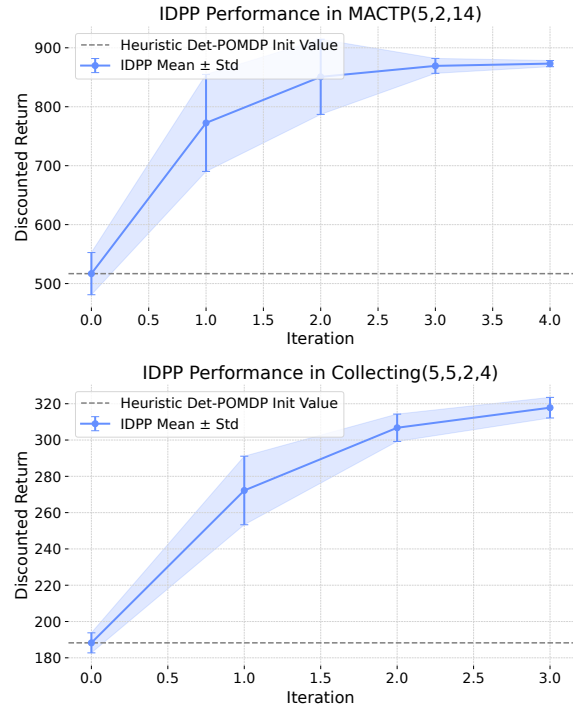


Figure 4: IDPP’s performance across iterations in problem instances MACTP(5, 2, 14) and Collecting(5, 5, 2, 4).

constructing each agent’s FSC node-by-node using Monte Carlo planning (POMCP) within a fixed time budget (1 second per node in our experiments). More planning time may improve results further.

Despite its simplicity, our heuristic initialization (Sec-

		MACTP $\langle N, n_a, n_e \rangle$				Collecting $\langle W, H, n_a, n_t \rangle$		
Instance		$\langle 3, 2, 5 \rangle$	$\langle 4, 2, 8 \rangle$	$\langle 4, 2, 12 \rangle$	$\langle 5, 2, 14 \rangle$	$\langle 4, 3, 2, 2 \rangle$	$\langle 4, 4, 2, 3 \rangle$	$\langle 5, 5, 2, 4 \rangle$
$ \mathcal{S} $		$\sim 2.6\text{k}$	$\sim 65.5\text{k}$	$\sim 1.05\text{M}$	$> 10\text{M}$	$\sim 4.7\text{k}$	$\sim 52.9\text{k}$	$\sim 2.82\text{M}$
$ b_0 $		2^5	2^8	2^{12}	2^{14}	30	112	$\sim 2.73\text{k}$
$ \mathcal{O}_i $		279	656	928	$\sim 1.83\text{k}$	166	593	$\sim 1.75\text{k}$
IQL	Return	849.60 ± 43.08	696.33 ± 45.92	526.31 ± 64.60	455.62 ± 83.22	121.95 ± 9.69	205.44 ± 11.64	216.23 ± 14.28
	Time	–	–	–	–	–	–	–
MAPPO	Return	808.84 ± 39.10	542.82 ± 46.48	427.97 ± 42.19	252.84 ± 23.79	125.17 ± 7.60	189.42 ± 6.34	197.63 ± 5.17
	Time	–	–	–	–	–	–	–
Det-POMDP Heur.	Return	682.13 ± 0.54	705.57 ± 8.26	603.95 ± 8.05	516.84 ± 35.77	118.60 ± 2.11	197.92 ± 2.33	186.46 ± 4.39
	Time	0.4 ± 0.2	8.2 ± 1.4	138.6 ± 34.6	710.3 ± 46.8	0.7 ± 0.5	15.3 ± 1.1	1064.8 ± 109.8
InfJESP	Return	918.37 ± 0.28	†	†	†	186.39 ± 0.72	†	†
	Time	23.8 ± 1.4	‡	‡	‡	44.6 ± 6.4	‡	‡
MCJESP	Return	874.39 ± 34.93	743.76 ± 70.83	459.64 ± 104.58	531.13 ± 82.86	177.51 ± 5.50	252.97 ± 16.51	237.09 ± 32.80
	Time	159.2 ± 71.1	197.6 ± 41.0	1381.3 ± 130.4	4115.8 ± 659.6	383.6 ± 52.9	628.8 ± 42.3	8509.6 ± 684.3
IDPP (Ours)	Return	912.71 ± 0.32	867.58 ± 2.78	798.47 ± 2.40	873.16 ± 5.21	184.42 ± 1.16	267.71 ± 0.32	315.56 ± 2.81
	Time	1.8 ± 0.4	15.3 ± 2.6	570.8 ± 110.3	1706.6 ± 298.4	2.4 ± 0.5	44.6 ± 1.4	4662.3 ± 488.6

Table 1: Performance (avg. return \pm std) and computation time (in seconds) of algorithms on MACTP and Collecting instances. Each instance is defined by its structural parameters. † indicates infeasibility due to memory constraints; ‡ means infeasible due to time limits, and – means not applicable.

tion 5) provides competitive performance relative to MCJESP at significantly lower computational cost. This demonstrates its effectiveness as a strong starting point for IDPP, which further improves solutions as shown in Figure 4. On large instances where InfJESP fails, IDPP consistently outperforms other methods with significant less computation time. By leveraging a deterministic POMDP solver in each iteration, IDPP achieves more accurate and efficient planning than MCJESP, leading to higher-quality Nash equilibrium policies. However, we note this advantage applies specifically to Det-Dec-POMDPs.

Finally, MARL methods MAPPO and IQL, relying solely on partial observations and sparse rewards, successfully learn policies for most tasks. This highlights the power of recurrent networks in handling partial observability. However, due to function approximation errors, their performance is generally inferior to model-based planners. Interestingly, IQL outperforms MAPPO in discounted return across most problems, despite MAPPO’s reputation as a state-of-the-art MARL method. Analysis reveals both succeed in task completion in most runs, but MAPPO tends to generate longer trajectories, lowering its discounted return. This may stem from (1) discrete action spaces favoring Q-learning methods like IQL, and (2) policy gradient methods like MAPPO emphasizing long-term optimization.

7 Discussion of Contributions and Limitations

In this article, we introduce the class of Deterministic Decentralized POMDPs (Det-Dec-POMDPs), a natural extension of Deterministic POMDPs (Bonet 2009) to the multi-agent setting. This model is also a further simplification of Quasi-Deterministic Dec-POMDPs (Besse and Chaib-Draa 2009),

assuming deterministic observations. Such a framework is well suited for problems where uncertainty stems solely from the initial state, and both actions and observations are deterministic—such as high-level task planning in multi-robot systems, including some navigation and path planning applications. We then propose IDPP, a practical JESP variant aimed at efficiently solving large-scale Det-Dec-POMDPs. The main idea is intuitive and effective: “choosing the right tool for the right subproblem,” where we decompose the large Det-Dec-POMDP into a sequence of individual agents’ Det-POMDPs to leverage powerful Det-POMDP solvers. As a result, IDPP becomes a highly efficient Det-Dec-POMDP solver that outperforms existing methods, to our knowledge, in this specific problem class. Moreover, we contribute two scalable benchmarks to facilitate research on scalability.

While IDPP is efficient for Det-Dec-POMDPs, it is not suitable for general Dec-POMDPs with stochastic transitions or observations. Therefore, it should be applied only when environment dynamics are deterministic. Another limitation is that our current IDPP implementation is single-threaded, so further speedups may be achieved through parallelization.

8 Conclusion

Many high-level robotic decision-making problems can be naturally modeled with deterministic actions and observations, where uncertainty primarily stems from the initial state. Motivated by this, we introduce the Det-Dec-POMDP framework to capture such structure, along with IDPP, a practical solver adapted from a JESP variant for solving large Det-Dec-POMDPs to support future applications. Our work may open a promising direction for planning in multi-agent partially observable domains where full stochasticity is unnecessary.

Acknowledgment

This work has been supported by the EPSRC Energy Programme under UKAEA/EPSC Fusion Grant 2022/2027 No. EP/W006839/1.

References

- Amato, C.; Bernstein, D. S.; and Zilberstein, S. 2010. Optimizing Fixed-Size Stochastic Controllers for POMDPs and Decentralized POMDPs. *Autonomous Agents and Multi-Agent Systems*, 21(3): 293–320.
- Bai, H.; Hsu, D.; Lee, W. S.; and Ngo, V. A. 2011. Monte Carlo Value Iteration for Continuous-State POMDPs. In *Algorithmic Foundations of Robotics IX: Selected Contributions of the Ninth International Workshop on the Algorithmic Foundations of Robotics*, 175–191. Springer.
- Bernstein, D. S.; Amato, C.; Hansen, E. A.; and Zilberstein, S. 2009. Policy iteration for decentralized control of Markov decision processes. *Journal of Artificial Intelligence Research*, 34: 89–132.
- Bernstein, D. S.; Givan, R.; Immerman, N.; and Zilberstein, S. 2002. The Complexity of Decentralized Control of Markov Decision Processes. *Mathematics of operations research*, 27(4): 819–840.
- Bernstein, D. S.; Hansen, E. A.; and Zilberstein, S. 2005. Bounded policy iteration for decentralized POMDPs. In *Proceedings of the nineteenth international joint conference on artificial intelligence (IJCAI)*, 52–57.
- Besse, C.; and Chaib-Draa, B. 2009. Quasi-deterministic partially observable markov decision processes. In *International Conference on Neural Information Processing*, 237–246. Springer.
- Bonet, B. 2009. Deterministic POMDPs Revisited. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI)*, 59–66. AUAI Press.
- Dibangoye, J. S.; Amato, C.; Buffet, O.; and Charpillet, F. 2016. Optimally solving Dec-POMDPs as continuous-state MDPs. *Journal of Artificial Intelligence Research*, 55: 443–497.
- Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural Computation*, 9(8): 1735–1780.
- Kumar, A.; and Zilberstein, S. 2012. Anytime planning for decentralized POMDPs using expectation maximization. *arXiv preprint arXiv:1203.3490*.
- Kumar, A.; Zilberstein, S.; and Toussaint, M. 2015. Probabilistic inference techniques for scalable multiagent decision making. *Journal of Artificial Intelligence Research*, 53: 223–270.
- Kurniawati, H.; Hsu, D.; and Lee, W. S. 2008. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and systems*, volume 2008. Citeseer.
- Lim, Z.; Sun, L.; and Hsu, D. 2011. Monte Carlo Value Iteration with Macro-Actions. In Shawe-Taylor, J.; Zemel, R.; Bartlett, P.; Pereira, F.; and Weinberger, K., eds., *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc.
- Littman, M. L. 1996. *Algorithms for Sequential Decision-Making*. Brown University. ISBN 0-591-16350-0.
- MacDermid, L. C.; and Isbell, C. L. 2013. Point based value iteration with optimal belief compression for Dec-POMDPs. *Advances in neural information processing systems*, 26.
- Nair, R.; Tambe, M.; Yokoo, M.; Pynadath, D.; and Marsella, S. 2003. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *IJCAI*, volume 3, 705–711.
- Oliehoek, F. A.; Spaan, M. T.; Terwijn, B.; Robbel, P.; et al. 2017. The MADP Toolbox: An open source library for planning and learning in (multi-) agent systems. *Journal of Machine Learning Research*, 18(89): 1–5.
- Oliehoek, F. A.; Spaan, M. T.; and Vlassis, N. 2008. Optimal and approximate Q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research*, 32: 289–353.
- Pajarinen, J.; and Peltonen, J. 2011a. Efficient planning for factored infinite-horizon DEC-POMDPs. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, 325.
- Pajarinen, J.; and Peltonen, J. 2011b. Periodic finite state controllers for efficient POMDP and DEC-POMDP planning. *Advances in neural information processing systems*, 24.
- Rashid, T.; Samvelyan, M.; de Witt, C. S.; Farquhar, G.; Foerster, J.; and Whiteson, S. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning*.
- Schutz, A.; You, Y.; Mattamala, M.; Caliskanelli, I.; Lacerda, B.; and Hawes, N. 2025. A Finite-State Controller Based Offline Solver for Deterministic POMDPs. *arXiv preprint arXiv:2505.00596*.
- Silver, D.; and Veness, J. 2010. Monte-Carlo planning in large POMDPs. *Advances in neural information processing systems*, 23.
- Song, Z.; Liao, X.; and Carin, L. 2016. Solving DEC-POMDPs by Expectation Maximization of Value Function. In *AAAI Spring Symposia*.
- Stadler, M.; Banfi, J.; and Roy, N. 2023. Approximating the value of collaborative team actions for efficient multiagent navigation in uncertain graphs. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 33, 677–685.
- Sunehag, P.; Lever, G.; Gruslyns, A.; Czarnecki, W. M.; Zambaldi, V.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo, J. Z.; Tuyls, K.; et al. 2017. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*.
- Szer, D.; Charpillet, F.; and Zilberstein, S. 2012. MAA*: A Heuristic Search Algorithm for Solving Decentralized POMDPs. *arXiv:1207.1359*.
- Tan, M. 1993. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, 330–337.

You, Y.; Thomas, V.; Colas, F.; and Buffet, O. 2021. Solving infinite-horizon Dec-POMDPs using finite state controllers within JESP. In *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, 427–434. IEEE.

You, Y.; Thomas, V.; Colas, F.; and Buffet, O. 2023. Monte-Carlo search for an equilibrium in Dec-POMDPs. In *Uncertainty in Artificial Intelligence*, 2444–2453. PMLR.

Yu, C.; Gleave, A.; Dennis, M.; Kant, N.; Levine, S.; and Russell, S. 2021. The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games. In *Advances in Neural Information Processing Systems*.