Optimally Solving Simultaneous-Move Dec-POMDPs: The Sequential Central Planning Approach

Johan Peralez[1], **Aurelien Delage**[1], **Jacopo Castellini**[2], **Rafael F. Cunha**[3], **Jilles S. Dibangoye**[3]
[1]Univ Lyon, INSA Lyon, Inria, CITI, EA3720, 69621 Villeurbanne, France
[2]Haute Ecole de Gestion de Genève, University of Applied Sciences and Arts Western, 1227 Carouge, Geneva, Switzerland
[3]Bernoulli Institute, University of Groningen, Nijenborgh 4, NL-9747AG, Groningen, Netherlands
johan.peralez@gmail.com, aurelien.delage@insa-lyon.fr, jacopo.castellini@hesge.ch, {r.f.cunha,j.s.dibangoye}@rug.nl

Abstract

The centralized training for decentralized execution paradigm emerged as the state-of-the-art approach to ϵ-optimally solving decentralized partially observable Markov decision processes. However, scalability remains a significant issue. This paper presents a novel and more scalable alternative, namely the sequential-move centralized training for decentralized execution. This paradigm further pushes the applicability of the Bellman's principle of optimality, raising three new properties. First, it allows a central planner to reason upon sufficient sequential-move statistics instead of prior simultaneous-move ones. Next, it proves that ϵ-optimal value functions are piecewise linear and convex in such sufficient sequential-move statistics. Finally, it drops the complexity of the backup operators from double exponential to polynomial at the expense of longer planning horizons. Besides, it makes it easy to use single-agent methods, *e.g.,* SARSA algorithm enhanced with these findings, while still preserving convergence guarantees. Experiments on two- as well as many-agent domains from the literature against ϵ-optimal simultaneous-move solvers confirm the superiority of our novel approach. This paradigm opens the door for efficient planning and reinforcement learning methods for multi-agent systems.

1 Introduction

The problem of designing a system with multiple agents that cooperate to control a hidden Markov chain is of interest in artificial intelligence, optimal decentralized and stochastic control, and game theory (Yoshikawa and Kobayashi 1978; Radner 1962; Ooi and Wornell 1996; Bernstein et al. 2002; Shoham and Leyton-Brown 2008). Simultaneous-move decentralized partially observable Markov decision processes (Dec-POMDPs) emerged as the standard framework for formalizing and solving such problems when agents act simultaneously. Given the growth of multi-agent intelligence systems, *e.g.,* online interactive services, the Internet of Things, and intelligent transportation systems, the importance of scalable multi-agent solutions is clear. It also highlights the impetus for methods with theoretical guarantees to find safe and secure solutions. However, as the situation currently stands, ϵ-optimally solving Dec-POMDPs is widely believed to be out of reach: finite-horizon cases are NEXP-hard, infinite-horizon cases are undecidable (Bernstein et al. 2002), and approximations remain intractable (Rabinovich, Goldman, and Rosenschein 2003).

There are two distinct but interdependent explanations for the limited scalability of existing algorithms. The more widely known reason is that agents in such a setting can neither see the state of the world nor explicitly communicate with one another due to communication costs, latency, or noise. However, what one agent sees and does directly affects what the others see and do. That **silent coordination problem** makes it hard to define a right notion of state necessary to apply dynamic programming theory (Hansen, Bernstein, and Zilberstein 2004). A standard solution method suggests reducing the original multi-agent problem into a single-agent one, which is then solved using single-agent algorithms based on recent advances in Markov decision processes. This simultaneous-move centralized training for decentralized execution paradigm applies successfully to planning (Szer, Charpillet, and Zilberstein 2005; Oliehoek, Spaan, and Vlassis 2008; Oliehoek et al. 2013; Dibangoye et al. 2016; Sokota et al. 2021) and reinforcement learning (Foerster et al. 2018; Rashid et al. 2018; Bono et al. 2018; Dibangoye and Buffet 2018). The less well-known reason for the poor scaling behavior is the requirement of current methods for simultaneous-move decision-making. In such methods, the simultaneous-move greedy selection of joint decision rules, *i.e.,* simultaneous-move Bellman's backup, is performed collectively for agents—all at once. Unfortunately, this **decision entanglement problem** is double exponential with agents and time, rendering even a single simultaneous-move Bellman's backup prohibitively expensive when facing realistic problems with long planning horizons—let alone finding an optimal solution.

In contrast, this paper builds upon the assumption that decision rules to be executed simultaneously can nonetheless be selected sequentially. A paradigm we refer to as sequential-move centralized training for simultaneous-move decentralized execution. This paradigm pushes further the applicability of Bellman's principle of optimality so that a central planner can still select joint decision rules, but now one private decision rule at a time. Doing so disentangles the decision variables with mutual influence and raises three new properties. First, it allows a sequential-move central planner to reason upon sufficient sequential-move statistics instead of prior simultaneous-move ones.

Next, it proves that ϵ-optimal value functions are still piecewise linear and convex in sufficient sequential-move statistics. Finally, it drops the complexity of Bellman's backups from double exponential to polynomial at the expense of longer planning horizons. We show that a SARSA algorithm enhanced by these findings applies while preserving its convergence guarantees. Experiments conducted in standard two- and multi-agent domains from existing literature, particularly against ϵ-optimal solvers such as feature-based heuristic search value iteration and local methods (Tan 1993; Konda and Tsitsiklis 1999), demonstrate the clear advantages of our sequential-move variant of the oSARSA algorithm (Dibangoye and Buffet 2018).

This remainder of this paper is organized as follows. Section 2 examines the dominant paradigm for optimally solving simultaneous-move Dec-POMDPs: simultaneous-move centralized training for decentralized execution. Although this paradigm has demonstrated efficiency in small to medium-sized teams of cooperative agents, it faces scalability issues when applied to larger teams. In Section 3, we advocate for a paradigm shift from simultaneous-move to sequential-move centralized training for decentralized execution. We establish that simultaneous-move Dec-POMDPs can be reduced to sequential-move ones, enabling knowledge transfer from the former to the latter. Section 4 illustrates that all existing structural results from the dominant paradigm are transferable to this new approach. However, this shift also uncovers an exponential drop in the complexity of backup operators. Finally, Section 5 adapts the oSARSA algorithm from the original paradigm to the new framework. The experiments in Section 6 provide substantial evidence for the superiority of the proposed paradigm.

2 Simultaneous-Move Central Planning

This section presents an overview of the dominant paradigm for solving cooperative multi-agent problems: simultaneous-move centralized training for decentralized control. We start by distinguishing between the multi-agent and single-agent formulations of simultaneous-move Dec-POMDPs. The multi-agent framework, while being the original design, faces significant challenges in dynamic programming applications. In contrast, the single-agent formulation enables a central planner to orchestrate the actions of all agents concurrently, thereby illustrating simultaneous-move centralized training for decentralized control. This paradigm leverages single-agent theories and algorithms and has emerged as the dominant approach for (optimally) solving simultaneous-move Dec-POMDPs.

2.1 Simultaneous-Move Multi-Agent Formulation

Definition 1. *A n-agent simultaneous-move Dec-POMDP is given by tuple $\mathtt{M} \doteq (\mathtt{X}, \mathtt{U}, \mathtt{Z}, \mathtt{p}, \mathtt{r})$. , where: $\mathtt{X}$ is a finite set of states, denoted $\mathtt{x}$ or $\mathtt{y}$; $\mathtt{U} \doteq \mathtt{U}^1 \times \cdots \times \mathtt{U}^n$ is a finite set of actions, denoted $\mathtt{u} = (\mathtt{u}^1, \cdots, \mathtt{u}^n)$; $\mathtt{Z} \doteq \mathtt{Z}^1 \times \cdots \times \mathtt{Z}^n$ is a finite set of observations, denoted $\mathtt{z} = (\mathtt{z}^1, \cdots, \mathtt{z}^n)$; $\mathtt{p}$ is the transition function that specifies the probability $\mathtt{p}^{\mathtt{u},\mathtt{z}}_{\mathtt{x},\mathtt{y}}$ of the process state being $\mathtt{y}$ and having observation $\mathtt{z}$ after taking action $\mathtt{u}$ in state $\mathtt{x}$; and $\mathtt{r}$ is the reward function specifying the reward $\mathtt{r}_{\mathtt{x},\mathtt{u}}$ received after taking action $\mathtt{u}$ in state $\mathtt{x}$.*

Throughout the paper, we make the following assumptions: (1) rewards are two-side bounded, *i.e.,* there exists some $c > 0$, such that $\|\mathtt{r}_{\cdot,\cdot}\|_\infty \leq c$; and planning horizon ℓ is finite. This is commonly achieved in infinite-horizon problems with a discount factor $\gamma \in [0, 1)$ by searching for an ϵ-optimal solution (for some $\epsilon > 0$) and setting $\ell = \lceil \log_\gamma (1-\gamma)\epsilon/c \rceil$. Optimally solving $\mathtt{M}$ aims at finding (simultaneous-move) joint policy $\xi = (\mathtt{a}^1_{0:\ell-1}, \cdots, \mathtt{a}^n_{0:\ell-1})$, *i.e.,* an n-tuple of sequences of private decision rules, one sequence of private decision rules $\mathtt{a}^i_{0:\ell-1} = (\mathtt{a}^i_0, \ldots, \mathtt{a}^i_{\ell-1})$ for each agent i. An optimal joint policy maximizes the expected γ-discounted cumulative reward starting from initial state distribution b_0 onward. This value can be expressed as:

$$v^\xi_{\mathtt{M},\gamma,0}(b_0) \doteq \mathbb{E}\{\textstyle\sum_{t=0}^{\ell-1} \gamma^t \mathtt{r}_{\mathtt{x}_t,\mathtt{u}_t} \mid b_0, \xi\}.$$

Let $\mathbb{N}_{\leq m} \doteq \{0, 1, \ldots, m\}$ and $\mathbb{N}^*_{\leq m} \doteq \mathbb{N}_{\leq m} \backslash \{0\}$. For each agent $i \in \mathbb{N}^*_{\leq n}$, private decision rule $\mathtt{a}^i_t \colon \mathtt{o}^i_t \mapsto \mathtt{u}^i_t$ depends on t-th private history $\mathtt{o}^i_t \doteq (\mathtt{u}^i_{0:t-1}, \mathtt{z}^i_{1:t})$, with 0-th private history being $\mathtt{o}^i_0 \doteq \emptyset$. At each step $t \in \mathbb{N}_{\leq \ell-1}$, we denote $\mathtt{O}^i_t$ and $\mathtt{A}^i_t$ the finite set of agent i' t-th private histories $\mathtt{o}^i_t \in \mathtt{O}^i_t$ and decision rules $\mathtt{a}^i_t \in \mathtt{A}^i_t$, respectively.

Optimally solving $\mathtt{M}$ using dynamic programming theory in its simultaneous-move multi-agent formulation is non-trivial, since it is not clear how to define a right notion of state (Hansen, Bernstein, and Zilberstein 2004). To better understand this, notice that every agent acts simultaneously, but can neither see the actual state of the world nor explicitly communicate its actions and observations with others. At the same time, what one agent sees and does directly affects what the others see and do, which explains the mutual influence of all decision variables $\mathtt{a}_t = (\mathtt{a}^1_t, \cdots, \mathtt{a}^n_t)$ at each time step $t \in \mathbb{N}_{\leq \ell-1}$, *i.e.,* a mapping called t-th joint decision rule $\mathtt{a}_t \colon \mathtt{o}_t \mapsto \mathtt{u}_t$ from joint histories $\mathtt{o}_t \doteq (\mathtt{o}^1_t, \ldots, \mathtt{o}^n_t)$ to joint actions. At each step $t \in \mathbb{N}_{\leq \ell-1}$, we denote $\mathtt{O}_t$ and $\mathtt{A}_t$ the finite set of t-th joint histories $\mathtt{o}_t \in \mathtt{O}_t$ and joint decision rules $\mathtt{a}_t \in \mathtt{A}_t$, respectively.

The motivation for a single-agent reformulation is twofold. The primary reason is that it allows us to reason simultaneously about all mutually dependent decision variables $\mathtt{a}_t = (\mathtt{a}^1_t, \cdots, \mathtt{a}^n_t)$, *i.e.,* a t-th joint decision rule. Besides, the single-agent reformulation eases the transfer of theory and algorithms from single- to multi-agent systems.

2.2 Single-Agent Reformulation

This equivalent single-agent reformulation aims at recasting $\mathtt{M}$ from the perspective of an offline simultaneous-move central planner. At every step, this planner simultaneously acts on behalf of all agents, taking a joint decision rule, but receives no feedback in the worst-case scenario. A statistic of the history of selected joint decision rules $\mathtt{a}_{0:t-1} = (\mathtt{a}_0, \ldots, \mathtt{a}_{t-1})$, *i.e.,* t-th simultaneous-move occupancy state $\mathtt{s}_t \doteq (\Pr\{\mathtt{x}_t, \mathtt{o}_t \mid b_0, \mathtt{a}_{0:t-1}\})_{\mathtt{x}_t \in \mathtt{X}, \mathtt{o}_t \in \mathtt{O}_t}$, is a conditional probability distribution over states and t-th joint histories. It describes a non-observable and deterministic Markov decision process, namely a simultaneous-move occupancy-state MDP (*o*MDP) (Dibangoye et al. 2016).

Definition 2. *A simultaneous-move oMDP* w.r.t. $\mathtt{M}$ *is given by a tuple $\mathtt{M}' \doteq (\mathtt{S}, \mathtt{A}, \mathtt{T}, \mathtt{R})$, where $\mathtt{S} = \cup_{t \in \mathbb{N}_{\leq \ell-1}} \mathtt{S}_t$ is a*

collection of simultaneous-move occupancy states, denoted $\mathtt{s}_t \in \mathtt{S}_t$*;* $\mathtt{A}$ *is the space of actions describing joint decision rules;* $\mathtt{T}\colon \mathtt{S} \times \mathtt{A} \to \mathtt{S}$ *is the deterministic transition rule, where next occupancy state is given by* $\mathtt{s}_{t+1} \doteq \mathtt{T}(\mathtt{s}_t, \mathtt{a}_t)$*,*

$$\mathtt{s}_{t+1}\colon (\mathtt{y}, (\mathtt{o}, \mathtt{u}, \mathtt{z})) \mapsto \textstyle\sum_{\mathtt{x}\in\mathtt{X}} \mathtt{s}_t(\mathtt{x}, \mathtt{o}) \cdot \mathtt{p}_{\mathtt{x},\mathtt{y}}^{\mathtt{a}_t(\mathtt{o}),\mathtt{z}},$$

and $\mathtt{R}\colon \mathtt{S}\times\mathtt{A} \to \mathbb{R}$ *describes the linear reward function*—i.e.,

$$\mathtt{R}\colon (\mathtt{s}_t, \mathtt{a}_t) \mapsto \textstyle\sum_{\mathtt{x}\in\mathtt{X}} \sum_{\mathtt{o}\in\mathtt{O}_t} \mathtt{s}_t(\mathtt{x}, \mathtt{o}) \cdot \mathtt{r}_{\mathtt{x},\mathtt{a}_t(\mathtt{o})}.$$

Optimally solving $\mathtt{M}'$ aims at finding a simultaneous-move joint policy ξ via an optimal value function $(\upsilon^*_{\mathtt{M}',\gamma,t})_{t\in\mathbb{N}_{\leq\ell}}$, *i.e.,* mappings from states to reals, $\upsilon^*_{\mathtt{M}',\gamma,t}(\mathtt{s}_t) \doteq \max_{\mathtt{a}_{t:\ell-1}\in\mathtt{A}_{t:\ell-1}} \upsilon^{\mathtt{a}_{t:\ell-1}}_{\mathtt{M}',\gamma,t}(\mathtt{s}_t)$, and solution of Bellman's optimality equations, *i.e.,*

$$\upsilon^*_{\mathtt{M}',\gamma,t}(\mathtt{s}_t) = \max_{\mathtt{a}_t\in\mathtt{A}_t} \mathtt{q}^*_{\mathtt{M}',\gamma,t}(\mathtt{s}_t, \mathtt{a}_t), \tag{1}$$

$$\mathtt{q}^*_{\mathtt{M}',\gamma,t}(\mathtt{s}_t, \mathtt{a}_t) \doteq \mathtt{R}(\mathtt{s}_t, \mathtt{a}_t) + \gamma\upsilon^*_{\mathtt{M}',\gamma,t+1}(\mathtt{T}(\mathtt{s}_t, \mathtt{a}_t)), \tag{2}$$

with boundary condition $\upsilon^*_{\mathtt{M}',\gamma,\ell}(\cdot) = \mathtt{q}^*_{\mathtt{M}',\gamma,\ell}(\cdot,\cdot) = 0$. One can greedily select an optimal joint policy given the optimal value function. Unfortunately, solving Bellman's optimality equations (1) is not trivial since simultaneous-move occupancy-state space $\mathtt{S}$ grows exponentially with time and number of agents. Instead, Dibangoye et al. (2016) build on the piecewise linearity and convexity in the simultaneous-move occupancy-state space of the ϵ-optimal value function. Also, they introduced simultaneous-move backup operators that can circumvent the exhaustive enumeration of all joint decision rules using mixed-integer linear programs. They provided equivalence relations among private histories to enhance value generalization. Altogether, these operations made it possible to use single-agent algorithms, *e.g.,* oSARSA (Dibangoye and Buffet 2018) and point-based value iteration (Dibangoye et al. 2016) algorithms, to solve $\mathtt{M}$ while preserving the convergence guarantees. However, scalability remains a major issue. The following lemma suggests there is room for greater scalability by breaking down the decision-making process one step and one agent at a time.

Lemma 1. *[Proof in Appendix C.1] Let* $\sigma\colon \mathbb{N}^*_{\leq n} \mapsto \mathbb{N}^*_{\leq n}$ *be a permutation over agents. Bellman's optimality equations (1) can be re-written in sequential form with no loss in optimality,* i.e., *for all* $t \in \mathbb{N}_{\leq\ell}$ *and any* t*-th simultaneous-move occupancy state* $\mathtt{s}_t$*,*

$$\upsilon^*_{\mathtt{M}',\gamma,t}(\mathtt{s}_t) = \max_{\mathtt{a}_t^{\sigma(1)}\in\mathtt{A}_t^{\sigma(1)}} \cdots \max_{\mathtt{a}_t^{\sigma(n)}\in\mathtt{A}_t^{\sigma(n)}} \mathtt{q}^*_{\mathtt{M}',\gamma,t}(\mathtt{s}_t, \mathtt{a}_t).$$

The main issue with the dominant paradigm is the necessity for simultaneous decision-making rather than the double exponential growth of joint policies over time and the number of agents. If simultaneous planning is conducted, then backups are performed collectively for all agents, all at once. In the much larger problems that are our long-term objective, the double exponential growth of joint policies would render even a single simultaneous-move backup, *cf.* Bellman's optimality equations (1), prohibitively expensive. Instead, Lemma 1 suggests that planning can take place in an offline sequential central manner while preserving optimality, even though agents (online) execute actions in a decentralized and simultaneous fashion. Doing so will disentangle the decision variables. Even more importantly, it will make it possible to implement efficient basic operations, *e.g.,* occupancy-state estimation, value generalization, and backup, of exact single-agent algorithms. The following section introduces the sequential-move central planner approach to optimally solving Dec-POMDPs, assuming for the sake of clarity that the permutation over agents σ is fixed throughout the planning process. We shall demonstrate the sequential-move centralized training for decentralized execution paradigm opens the doors for more scalable single-agent algorithms to apply to multi-agent problems.

3 Sequential-Move Central Planning

This section presents an alternative paradigm to (optimally) solving simultaneous-move Dec-POMDPs aimed at overcoming the limitations inherent to the dominant approach. We will establish that simultaneous-move Dec-POMDPs reduce to sequential-move Dec-POMDPs. This reduction enables us to apply techniques designed for sequential-move settings to their simultaneous counterparts, thereby introducing the sequential-move centralized training for decentralized control. For clarity, we will distinguish between multi-agent and single-agent formulations of sequential-move Dec-POMDPs, while also establishing their equivalence, which enables the transfer of theories and algorithms from single-agent to sequential-move multi-agent scenarios.

3.1 Sequential-Move Multi-Agent Formulation

This subsection starts with a fairly general definition of n-agent sequential-move Dec-POMDPs. Theorem 1 provides the formal proof that one can convert any simultaneous-move Dec-POMDP into an equivalent sequential-move one, allowing us to solve the former using the latter.

Definition 3. *A* n*-agent sequential-move Dec-POMDP is given by a tuple* $M \doteq (X, U, Z, \rho, p, r, \ell')$*, where:* X *is a finite set of states, denoted* x *or* y*;* $U(\tau)$ *is a finite set of individual actions available to any arbitrary agent at decision epoch* τ*, denoted* u*;* Z^i *is a finite set of individual observations of agent* i*, denoted* z^i*, with* $Z \doteq Z^1 \times Z^2 \times \cdots \times Z^n$ *and* $z \doteq (z^1, \cdots, z^n)$*;* $\rho\colon \mathbb{N}_{\leq\ell'-1} \to \mathbb{N}^*_{\leq n}$ *is the agent function*[1]*, which assigns to each decision epoch* $\tau \in \mathbb{N}_{\leq\ell'-1}$ *an agent* $i \in \mathbb{N}^*_{\leq n}$ *who chooses an action at that step;* $p^{uz}_{xy}(\tau)$ *is the* τ*-th dynamics function that specifies the probability of the state being* y *and having observation* z *after taking action* u *in state* x*;* $r_{x,u}(\tau)$ *is the* τ*-th reward function that specifies the reward received after agent* $\rho(\tau)$ *takes action* u *in state* x*; and* ℓ' *is the planning horizon.*

In sequential-move multi-agent formulation M, agents may start with noisy information regarding the state of the world, specified by the initial state distribution b_0. At each decision epoch τ, the world is in some state x_τ. Agent $\rho(\tau)$ chooses any action u_τ available in action space $U(\tau)$. Upon

[1]It is worth noticing that agent function ρ prescribes $(H^1, \ldots, H^n)$, with H^i being the set of points in time agent i takes actions, so that $\cap_{i\in\mathbb{N}^*_{\leq n}} H^i = \emptyset$, with $\cup_{i\in\mathbb{N}^*_{\leq n}} H^i = \mathbb{N}_{\leq\ell'-1}$.

the execution of action u_τ in state x_τ, the world immediately responds with the corresponding reward $r_{x_\tau,u_\tau}(\tau)$. It further responds at the subsequent decision epoch by randomly moving into a new state $x_{\tau+1}$ and provides each agent i with sequential observation $z^i_{\tau+1}$, according to the dynamics function $p^{u_\tau,z_{\tau+1}}_{x_\tau,x_{\tau+1}}(\tau)$. The process repeats iteratively until the planning horizon is exhausted. Optimally solving M aims at finding sequential-move joint policy π, *i.e.,* a sequence of private decision rules, $\pi \doteq (a_\tau)_{\tau\in\mathbb{N}_{\leq \ell'-1}}$, which maximizes the expected cumulative γ-discounted reward starting from initial state distribution b_0 onward, *i.e.,*

$$\upsilon^\pi_{M,\lambda,0}(b_0) \doteq \mathbb{E}\{\textstyle\sum_{\tau=0}^{\ell'-1} \lambda(\tau)\cdot r_{x_\tau,u_\tau}(\tau) \mid b_0,\pi\}, \qquad (3)$$

where $\lambda\colon \mathbb{N}_{\leq \ell'-1} \to (0,1)$ is the discount function. At every decision epoch τ, agent $i = \rho(\tau)$ acts upon private decision rule $a_\tau\colon o^i_\tau \mapsto u_\tau$, which depends on sequential private histories. Sequential private histories are defined as $o^i_{\tau+1} = (o^i_\tau, u_\tau, z^i_{\tau+1})$ if agent $i = \rho(\tau)$; and $o^i_{\tau+1} = (o^i_\tau, z^i_{\tau+1})$ otherwise, with initial sequential private history being $o^i_0 \doteq \emptyset$. At each step $\tau \in \mathbb{N}_{\leq \ell'-1}$, we denote O^i_τ and A_τ the finite set of agent i's τ-th private histories $o^i_\tau \in O^i_\tau$ and decision rules $a_\tau \in A_\tau$, respectively.

Upon executing a sequential joint policy, agents collectively build a sequential joint history satisfying the following recursive formula: $o_{\tau+1} = (o_\tau, u_\tau, z_{\tau+1})$. At each step $\tau \in \mathbb{N}_{\leq \ell'-1}$, we denote O_τ the finite set of sequential joint histories $o_\tau \in O_\tau$. Furthermore, each sequential joint history o_τ includes the sequential private history o^i_τ of each agent i. We denote the sequential private policy of agent i extracted from sequential joint policy π as $\pi^i \doteq (a_\tau)_{\tau\in H^i}$. Perhaps surprisingly, we establish that one can always convert any simultaneous-move Dec-POMDP into a sequential-move one with no loss in optimality.

Theorem 1. *[Proof in Appendix C.2] For every simultaneous-move Dec-POMDP* $\mathtt{M}$*, there exists a corresponding sequential-move Dec-POMDP* M *along with the appropriate performance criterion (3), whose optimal solution is also optimal for* $\mathtt{M}$*.*

Theorem 1 establishes that any simultaneous-move Dec-POMDP can be solved as a sequential-move Dec-POMDP. One can similarly recast any simultaneous-move Dec-POMDP into an extensive-form game (Kovařík et al. 2022). The main similarity is that both transformations allow the sequencing of decision-making one agent at a time. Yet, these transformations remain fundamentally different, mainly as sequential-move Dec-POMDPs make applying theory and algorithms derived from Markov decision processes easy.

3.2 Sequential-Move Single-Agent Reformulation

In the remainder, we assume a sequential central planner who knows model M and selects what to do on behalf of all agents. Even though agents act and receive feedback about the state of the world simultaneously, as in $\mathtt{M}$, our sequential central planner can select the simultaneous private decision rule of one agent at a time by relying on the equivalent model M. To this end, it essentially traverses the space of joint decision rules by performing expansions of joint decision rules, one private decision rule, or one agent, at a time. This sequential expansion is illustrated in Figure 1, *cf.* Appendix. The selection of a private decision rule of an agent depends on the choice of private decision rules from previous points in time and the initial belief state about the process. We call sequential exhaustive data the overall data available to the sequential central planner at each sequential decision epoch.

Definition 4. *For all sequential decision epoch* $\tau \in \mathbb{N}_{\leq \ell'-1}$*, sequential exhaustive data* $\iota_\tau \doteq (b_0, a_{0:\tau-1})$*.*

Sequential exhaustive data satisfy recursion $\iota_{\tau+1} = (\iota_\tau, a_\tau)$, with boundary condition $\iota_0 \doteq (b_0)$. It is worth noticing that variables $(\iota_\tau)_{\tau\in\mathbb{N}_{\leq \ell'-1}}$ taking values in sequential exhaustive data set $\boldsymbol{S}$ define a sequential-move data-driven Markov decision process $\boldsymbol{M}$. In such a process, states are sequential exhaustive data and actions are private decision rules, as illustrated in the influence diagram in Figure 1, *cf.* Appendix. This process is deterministic since the next sequential exhaustive data is given by concatenating the previous one ι_τ along with the sequential action choice a_τ, *i.e.,* $\boldsymbol{p}\colon (\iota_\tau, a_\tau) \mapsto \iota_{\tau+1}$. Furthermore, upon taking sequential action a_τ at sequential exhaustive data ι_τ, the expected reward is $\boldsymbol{r}\colon (\iota_\tau, a_\tau) \mapsto \mathbb{E}\{\lambda(\tau)\cdot r_{x_\tau,u_\tau}(\tau) \mid \iota_\tau, a_\tau\}$.

Definition 5. *A sequential-move data-driven Markov decision process* w.r.t. M *is given by* $\boldsymbol{M} \doteq \langle \boldsymbol{S}, A, \boldsymbol{p}, \boldsymbol{r}\rangle$ *where:* $\boldsymbol{S} \doteq \cup_{\tau\in\mathbb{N}_{\leq \ell'-1}} \boldsymbol{S}_\tau$ *defines the state space, where* $\boldsymbol{S}_\tau$ *is the set of all sequential exhaustive data at sequential decision epoch* $\tau \in \mathbb{N}_{\leq \ell'-1}$*;* $A \doteq \cup_{\tau\in\mathbb{N}_{\leq \ell'-1}} A_\tau$ *denotes the action space, where* A_τ *is the action space available at sequential decision epoch* $\tau \in \mathbb{N}_{\leq \ell'-1}$*;* $\boldsymbol{p}\colon \boldsymbol{S}\times A \to \boldsymbol{S}$ *prescribes the next state* $\iota_{\tau+1} = \boldsymbol{p}(\iota_\tau, a_\tau)$ *after taking action* a_τ *at state* ι_τ*;* $\boldsymbol{r}\colon \boldsymbol{S}\times A \to \mathbb{R}$ *specifies the immediate reward* $\boldsymbol{r}(\iota_\tau, a_\tau)$ *upon taking action* a_τ *in state* ι_τ*; and* $\iota_0 \doteq (b_0)$*.*

Similarly to M, optimally solving $\boldsymbol{M}$ aims at finding sequential joint policy π, which maximizes the cumulative γ-discounted reward starting from ι_0 onward, and given by $\upsilon^\pi_{\boldsymbol{M},\gamma,0}(\iota_0) \doteq \mathbb{E}\{\sum_{\tau=0}^{\ell'-1} \boldsymbol{r}(\iota_\tau, a_\tau) \mid \iota_0, \pi\}$. Building upon theory and algorithms for MDPs, one can extract an optimal sequential joint policy π^* from the solution of Bellman's optimality equations, *i.e.,*

$$\upsilon^*_{\boldsymbol{M},\lambda,\tau}(\iota_\tau) = \max_{a_\tau\in A_\tau} q^*_{\boldsymbol{M},\lambda,\tau}(\iota_\tau, a_\tau)$$
$$q^*_{\boldsymbol{M},\lambda,\tau}(\iota_\tau, a_\tau) = \boldsymbol{r}(\iota_\tau, a_\tau) + \upsilon^*_{\boldsymbol{M},\lambda,\tau+1}(\boldsymbol{p}(\iota_\tau, a_\tau))$$

with boundary condition $\upsilon^*_{\boldsymbol{M},\lambda,\ell'}(\cdot) = q^*_{\boldsymbol{M},\lambda,\ell'}(\cdot,\cdot) \doteq 0$. To transfer knowledge from sequential-move data-driven MDP $\boldsymbol{M}$ to corresponding sequential-move Dec-POMDP M, it will prove useful to show $\boldsymbol{M}$ and M are equivalent.

Lemma 2. *[Proof in Appendix C.3] If we let* $\boldsymbol{M}$ *be a sequential-move data-driven MDP* w.r.t. *original problem* M*, then any optimal sequential joint policy* $\pi^*_{\boldsymbol{M}}$ *for* $\boldsymbol{M}$ *is also an optimal solution for* M*.*

$\boldsymbol{M}$ differs from M in that the search space of $\boldsymbol{M}$ is made explicit, *i.e.,* a state in $\boldsymbol{M}$ (or a sequential exhaustive data) is a point in the search space. Consequently, space $\boldsymbol{S}$ is too large to be generated and stored in memory. It is not useful to remember all the states the sequential central planner experienced. Instead, one can rely on more concise representations, often referred to as statistics. Analogously to

Dibangoye et al. (2016), we introduce a statistic that summarizes the sequential exhaustive data, and name that statistic sequential-move occupancy state (SOC).

Definition 6. *A SOC $s_\tau \in \triangle(X \times O_\tau)$ at step $\tau \in \mathbb{N}_{\leq \ell'-1}$ is defined as a posterior probability distribution of hidden states and sequential joint histories given sequential exhaustive data ι_τ,* i.e., $s_\tau \colon (x_\tau, o_\tau) \mapsto \Pr\{x_\tau, o_\tau | \iota_\tau\}$.

Previously, Dibangoye et al. (2016) introduced the concept of (simultaneous-move) occupancy state to select a simultaneous joint decision rule on behalf of all agents, all at once. Instead, SOCs make it possible to select a private decision rule for each agent, one agent at a time. However, not all statistics can replace the sequential exhaustive data while preserving the ability to find an optimal solution. Statistics that exhibit such property include sufficient statistics of sequential exhaustive data for optimal decision-making in sequential-move Dec-POMDP M. Before further proving the sufficiency of SOCs, we start with two simple yet important preliminary lemmas.

Lemma 3. *[Proof in Appendix C.4] SOCs $(s_\tau)_{\tau \in \mathbb{N}_{\leq \ell'-1}}$ describe a Markov process. In other words, there exists $T \colon (s_\tau, a_\tau) \mapsto s_{\tau+1}$,* i.e., *for every state-history pair $\theta = (x_{\tau+1}, \langle o_\tau, u_\tau, z_{\tau+1} \rangle)$,*

$$s_{\tau+1}(\theta) = \textstyle\sum_{x_\tau \in X} p^{a_\tau(o_\tau^{\rho(\tau)}), z_{\tau+1}}_{x_\tau, x_{\tau+1}}(\tau) \cdot s_\tau(x_\tau, o_\tau).$$

Lemma 3 describes the transition rule of a continuous-state deterministic Markov decision process M', in which states and actions are SOCs and private decision rules, respectively. Next, we define the expected immediate reward model of M' that SOCs and private decision rules induce.

Lemma 4. *[Proof in Appendix C.5] SOCs $(s_\tau)_{\tau \in \mathbb{N}_{\leq \ell'-1}}$ are sufficient statistics of sequential exhaustive data $(\iota_\tau)_{\tau \in \mathbb{N}_{\leq \ell'-1}}$ for estimating expected immediate reward. In other words, there exists a mapping $R \colon (s_\tau, a_\tau) \mapsto \mathbb{E}\{\lambda(\tau) \cdot r_{x_\tau, u_\tau}(\tau) \mid s_\tau, a_\tau\}$ such that for every sequential exhaustive data ι_τ, corresponding SOC s_τ, and any private decision rule a_τ, we have:*

$$R(s_\tau, a_\tau) \doteq \boldsymbol{r}(\iota_\tau, a_\tau).$$

We are now ready to define sequential-move occupancy Markov decision process (*so*MDP) M' the Markov decision process that SOCs describe via dynamics and reward models T (*cf.* Lemma 3) and R (*cf.* Lemma 4), respectively.

Definition 7. *A soMDP* w.r.t. M *is given by a tuple $M' \doteq (S, A, T, R, \ell')$ where: $S \subset \triangle(X \times O)$ is the space of SOCs; A is as in M; $T \colon S \times A \to S$ is the transition rule; and $R \colon S \times A \to \mathbb{R}$ is the immediate reward model.*

The goal of *so*MDP M' is to find an optimal sequential joint policy π which maximizes the cumulative γ-discounted reward starting from s_0 onward, *i.e.,* $v^\pi_{M',\lambda,0}(s_0) \doteq \mathbb{E}\{\sum_{\tau=0}^{\ell'-1} R(s_\tau, a_\tau) \mid s_0, \pi\}$. The optimal state- and action-value functions $(v^*_{M',\lambda,\tau}, q^*_{M',\lambda,\tau})_{\tau \in \mathbb{N}_{\leq \ell'-1}}$ of M' are solutions of Bellman's optimality equations: for every $s_\tau \in S$,

$$v^*_{M',\lambda,\tau}(s_\tau) = \max_{a_\tau \in A_\tau} q^*_{M',\lambda,\tau}(s_\tau, a_\tau),$$
$$q^*_{M',\lambda,\tau}(s_\tau, a_\tau) = R(s_\tau, a_\tau) + v^*_{M',\lambda,\tau+1}(T(s_\tau, a_\tau)),$$

with boundary condition $v^*_{M',\lambda,\ell'}(\cdot) = q^*_{M,\lambda,\ell'}(\cdot, \cdot) \doteq 0$. Given the optimal value functions, one can extract an optimal sequential joint policy π^* as follows: for every decision epoch $\tau \in \mathbb{N}_{\leq \ell}$, and SOC $s^*_\tau \doteq (\Pr\{x_\tau, o_\tau | s_0, \pi^*\})_{x_\tau \in X, o_\tau \in O}$, we have $a^*_\tau = \arg\max_{a_\tau \in A_\tau} q^*_{M',\lambda,\tau}(s^*_\tau, a_\tau)$. Interestingly, an optimal sequential joint policy for *so*MDP M' is also an optimal joint policy for the original sequential-move Dec-POMDP M.

Theorem 2. *[Proof in Appendix C.6] Let M be a sequential-move Dec-POMDP. Let $\boldsymbol{M}$ be a sequential-move data-driven MDP* w.r.t. M. *SOCs $(s_\tau)_{\tau \in \mathbb{N}_{\leq \ell'-1}}$ are statistics sufficient to replace the sequential exhaustive data $(\iota_\tau)_{\tau \in \mathbb{N}_{\leq \ell'-1}}$ via soMDP M'* w.r.t. M *while still preserving the ability to optimally solve $\boldsymbol{M}$ (resp. M).*

Theorem 2 demonstrates that—by optimally solving any problem M', $\boldsymbol{M}$ or M along with the appropriate performance criteria—we are guaranteed to find an optimal solution for the others.

4 Exploiting Structural Results

The previous section shows that—to solve simultaneous-move Dec-POMDP M—one can equivalently solve the corresponding *so*MDP M'. This section shows optimal sequential state- and action-value functions $(v^*_{M',\lambda,\tau}, q^*_{M',\lambda,\tau})_{\tau \in \mathbb{N}_{\leq \ell'-1}}$ for M' are piecewise linear and convex functions of states and actions in M'. These convexity properties enable us to generalize values experienced in a few states and actions over the entire state and action spaces. Doing so speeds up convergence towards an optimal solution for M' (respectively, M). Similar properties exist for simultaneous-move Dec-POMDPs as well. However, maintaining sequential action-value functions will prove more tractable—*i.e.,* point-based simultaneous-move backups are double exponential, whereas sequential-move ones exhibit only polynomial-time complexity in the worst case.

Before we state one of the main results of this paper—the piecewise linearity and convexity of optimal sequential state- and action-value functions over state and action spaces—we introduce the following preliminary definition of sequential state-action occupancy measures and related properties. Let $\hat{s}_\tau$ and $\hat{a}_\tau$ be extended variants of SOCs s_τ and private decision rules a_τ, respectively. In other words, for SOC s_τ and private decision rule a_τ, we have $\hat{s}_\tau \colon (x_\tau, o_\tau, u_\tau) \mapsto s_\tau(x_\tau, o_\tau)$ and $\hat{a}_\tau \colon (x_\tau, o_\tau, u_\tau) \mapsto \Pr\{u_\tau | a_\tau, o_\tau^{\rho(\tau)}\}$. We denote $\hat{S}$ and $\hat{A}$ the extended spaces of SOCs and private decision rules, respectively. We further let $\hat{S} \odot \hat{A}$ be the space of all Hadamard products $\hat{s}_\tau \odot \hat{a}_\tau$ between extended SOC $\hat{s}_\tau$ and private decision rule $\hat{a}_\tau$, that is sequential state-action occupancy measure, where $[\hat{s}_\tau \odot \hat{a}_\tau](x_\tau, o_\tau, u_\tau) = \hat{s}_\tau(x_\tau, o_\tau, u_\tau) \cdot \hat{a}_\tau(x_\tau, o_\tau, u_\tau)$.

Theorem 3. *[Proof in Appendix D.5] For any soMDP M', optimal state-value functions $v^*_{M',\lambda,0:\ell'}$ are piecewise linear and convex* w.r.t. *SOCs. Furthermore, optimal action-value functions $q^*_{M',\lambda,0:\ell'}$ are also piecewise linear and convex* w.r.t. *sequential state-action occupancy measures.*

Theorem 3 proved the optimal state-value function for M' is piecewise linear and convex. We now introduce a practical representation of piecewise linear and convex optimal state- and action-value functions $v^*_{M',\lambda,0:\ell'}$ and $q^*_{M',\lambda,0:\ell'}$, respectively. We show they can be represented as the upper envelope of a finite collection of linear functions. Such a representation, suitable only for lower-bound state- and action-value functions, is referred to as the max-plane representation (Smith 2007). In addition, we describe operations for initializing, updating and selecting greedy decisions.

Corollary 1. *[Proof in Appendix F.1] For any soMDP M', let $v^*_{M',\lambda,0:\ell'}$ be the optimal state-value functions* w.r.t. *SOCs. Then, for every $\tau \in \mathbb{N}_{\leq \ell'-1}$, there exists a finite collection of linear functions* w.r.t. *SOCs, $V_\tau \doteq \{\alpha_\tau^{(\kappa)} \colon \kappa \in \mathbb{N}_{\leq k}\}$ such that:*

$$v^*_{M',\lambda,\tau} \colon s_\tau \mapsto \max_{\alpha_\tau^{(\kappa)} \in V_\tau} \langle s_\tau, \alpha_\tau^{(\kappa)} \rangle$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product.

Next, it will prove useful to define a constant private decision rule $a_\tau^{u_\tau}(\cdot) = u_\tau$, which prescribes u_τ for any private history, and the action-value linear function $\beta_\tau^{(\kappa)}$ that state-value linear function $\alpha_{\tau+1}^{(\kappa)}$ induces, *i.e.,* for any (x_τ, o_τ, u_τ),

$$\beta_\tau^{(\kappa)}(x_\tau, o_\tau, u_\tau) = \mathbb{E}\{\lambda(\tau) r_{x_\tau, u_\tau}(\tau) + \alpha_{\tau+1}^{(\kappa)}(x_{\tau+1}, o_{\tau+1})\}.$$

We are now ready to describe how to greedily select a private decision rule from a given state-value function.

Theorem 4. *[Proof in Appendix F.2] For any soMDP M', let $V_{\tau+1}$ be a finite collection of linear functions* w.r.t. *SOCs providing the max-plane representation of state-value function $v_{M',\lambda,\tau+1}$. Then, it follows that greedy private decision rule a^*_τ at SOC s_τ is:*

$$(a_\tau^{s_\tau}, \beta_\tau^{s_\tau}) \in \arg\max_{(a_\tau^{(\kappa)}, \beta_\tau^{(\kappa)}) \colon \kappa \in \mathbb{N}_{\leq k}} \langle \hat{s}_\tau \odot \hat{a}_\tau^{(\kappa)}, \beta_\tau^{(\kappa)} \rangle,$$

$$a_\tau^{(\kappa)}(o_\tau^{\rho(\tau)}) \in \arg\max_{u_\tau \in U(\tau)} \langle \hat{s}_\tau \odot \hat{a}_\tau^{u_\tau}, \beta_\tau^{(\kappa)} \rangle.$$

Also, $\alpha_\tau^{s_\tau} \colon (x_\tau, o_\tau) \mapsto \beta_\tau^{s_\tau}(x_\tau, o_\tau, a_\tau^{s_\tau}(o_\tau^{\rho(\tau)}))$ is the greedy linear function induced by $\beta_\tau^{s_\tau}$. The update of finite collection V_τ is as follows: $V_\tau \leftarrow V_\tau \cup \{\alpha_\tau^{s_\tau}\}$.

Theorem 4 greedily selects a decision rule $a_\tau^{s_\tau}$ according to a lower-bound value function at a given SOC s_τ and a linear action-value function $\beta_\tau^{s_\tau}$ to update the lower-bound value function $v_{M',\lambda,\tau}$. It then computes a linear state-value function $\alpha_\tau^{s_\tau}$ over SOCs to be added in the collection of linear functions V_τ. These update rules exhibit a complexity that is polynomial only *w.r.t.* the sizes of SOC s_τ and set V_τ, *cf.* Lemma 5 in the appendix. Similar rules have been previously introduced for the simultaneous-move setting. Unfortunately, these rules relied on mixed-integer linear programs in the best case. In the worst case, they enumerate all joint decision rules, thus creating a double exponential time complexity, *cf.* Lemma 5 in the appendix.

5 The *o*SARSA Algorithm

This section extends a reinforcement learning algorithm for ϵ-optimally solving simultaneous-move Dec-POMDPs, known as *o*SARSA (Dibangoye and Buffet 2018), to sequential centralized training for decentralized execution.

Algorithm 1: *o*SARSA(ϵ).

```
Init a_{0:ℓ'-1} ← blind policy, α.(·) ← 0, g. ← −∞.
foreach episode do
    Init SOC s_0 ← b_0 and step τ_0 ← 0.
    for τ = 0 to ℓ' − 1 do
        Select ε-greedily a_τ^{s_τ} w.r.t. α_{τ+1}.
        Compute SOC s_{τ+1} ← T(s_τ, a_τ^{s_τ}).
        if Accept(a_τ^{s_τ}, g_{τ+1}, α_{τ+1}(s_{τ+1})) then
            a_{0:ℓ'-1} ← ⟨a_{0:τ-1}, a_τ^{s_τ}, a_{τ+1:ℓ'-1}⟩.
            g_{τ+1} ← α_{τ+1}(s_{τ+1}).
            τ_0 ← τ.
    for τ = τ_0 to 0 do
        Update α_τ backward.
```

*o*SARSA generalizes SARSA from MDPs to *o*MDPs. It iteratively improves a (sequential) policy and its corresponding linear action-value functions, one per time step. The improved (sequential) policies are constructed by generating trajectories of OCs, one per episode, guided by an ϵ-greedy exploration strategy. A mixed-integer linear program (MILP) is used to select greedy joint decision rules to avoid the enumeration of double-exponential joint decision rules at every step. Once a trajectory resumes, it updates the linear action-value functions in the reversed order in which OCs were visited in it. We chose *o*SARSA among all point-based algorithms because it leverages the piecewise linearity and convexity properties of optimal value functions in Dec-POMDPs, yet maintains only a single linear value function per time step. Besides, it is guaranteed to find ϵ-optimal policy asymptotically. Note that algorithms that do not restrict to the max-plane representations for value functions, *e.g.,* the feature-based value iteration (FB-HSVI) algorithm (Dibangoye et al. 2016), may not fully exploit our findings.

To leverage the sequential centralized training for the decentralized execution paradigm, we apply the *o*SARSA algorithm in M' corresponding to $\mathtt{M}$. Hence, Algorithm 1 proceeds by generating trajectories of SOCs instead of OCs, one trajectory per episode, guided by sequential ϵ-greedy exploration steps. The sequential greedy selection rule requires only a polynomial time complexity, *cf.* Theorem 4. We use a portfolio of heuristic policies to guide the selection of the next sequential action to enhance exploration. This portfolio includes the random policy, the underlying MDP policy, and the blind policy (Hauskrecht 2000). We introduce an acceptance rule based on Simulated Annealing (SA) (Rutenbar 1989) to avoid the algorithm being stuck in a local optimum. Therein, a modification of the current sequential policy is kept not only if the performance improves but also in the opposite case, with a probability depending on the loss and on a temperature coefficient decreasing with time. While the point-based greedy selection rules in the sequential centralized training for decentralized execution paradigm are faster than those in its simultaneous counterpart, the length of tra-

jectories increases by order of n, *i.e.,* the number of agents. In most domains, however, the complexity of a single point-based greedy selection affects oSARSA far more strongly than the length of trajectories. The complexity of point-based greedy selection is a far better predictor of the ability to solve a simultaneous-move Dec-POMDP using oSARSA than the length of trajectories.

6 Experiments

This section presents the results of our experiments, which were carried out to juxtapose the sequential planning approach with its simultaneous counterpart employed in many leading-edge global multi-agent planning and reinforcement learning algorithms, encompassing the utilization of oSARSA, *cf.* Algorithm 1, as a standard algorithmic scheme. Our empirical analysis involves two variants of the oSARSA algorithm, namely oSARSA$^{\texttt{sim}}$ and oSARSA$^{\texttt{seq}}$, each employing a distinct reformulation of the original simultaneous-move Dec-POMDP $\texttt{M}$ and point-based backup method. oSARSA$^{\texttt{sim}}$ relies on oMDP $\texttt{M}'$ *w.r.t.* $\texttt{M}$ and utilizes mixed-integer linear programs (MILPs) for implicit enumeration of joint decision rules. We used ILOG CPLEX Optimization Studio to solve the MILPs. oSARSA$^{\texttt{seq}}$, instead, relies on soMDP M' *w.r.t.* $\texttt{M}$ and utilizes point-based sequential backup operator introduced in Theorem 4. We used the same heuristics (random, blind, and MDP policies) for oSARSA$^{\texttt{sim}}$ and oSARSA$^{\texttt{seq}}$. For reference, we also reported, when available, the performance of state-of-the-art ϵ-optimal solver for two-agent simultaneous-move Dec-POMDP $\texttt{M}$, *i.e.,* FB-HSVI. Unfortunately, most global methods are not geared to scale up with the number of agents. To present a comprehensive view, we have also compared our results against local policy- and value-based reinforcement learning methods, *i.e.,* advantage actor-critic (A2C) (Konda and Tsitsiklis 1999) and independent Q-learning (IQL) (Tan 1993). All experiments were run on an Ubuntu machine with 16 GB of available RAM and an 1.8 GHz processor, using only one core and a time limit of 1 hour. The source code is available at https://git.lwp.rug.nl/ml-rug/osarsa-aaai-25.

We have comprehensively assessed various algorithms using several two-agent benchmarks from academic literature, available at masplan.org. These benchmarks include mabc, recycling, grid3x3, boxpushing, mars, and tiger. To enable the comparison of multiple agents, we have also used the multi-agent variants of these benchmarks, *cf.* (Peralez et al. 2024). Please refer to the appendix for a detailed description of these benchmarks. Our empirical study aimed to assess the superiority of the sequential planning approach through the drop in the complexity of the point-based backup operators. Each experience in reinforcement learning algorithms, *i.e.,* A2C, IQL and oSARSA, was repeated with three different seeds. The values reported in Tables 1 and 2, are the best solution obtained among the three trials for two- and many-agent domains, respectively.

Table 1 shows that oSARSA$^{\texttt{seq}}$ outperforms oSARSA$^{\texttt{sim}}$ on all tested two-agent domains, sometimes by a significant margin. On boxpushing at planning horizon $\ell = 100$, for instance, oSARSA$^{\texttt{seq}}$ achieves value 2366.21 against 1895.16

ℓ	oSARSA$^{\texttt{seq}}$	oSARSA$^{\texttt{sim}}$	A2C	IQL	HSVI
		tiger			
10	**15.18**	**15.18**	−0.78	−1.30	**15.18**
20	**30.37**	**30.37**	−25.08	−14.61	28.75
40	**67.09**	**67.09**	−62.23	−50.72	**67.09**
100	**170.91**	169.30	−184.71	−165.42	**170.90**
		recycling			
10	**31.86**	**31.86**	31.48	31.48	**31.86**
20	**62.63**	**62.63**	62.25	**62.63**	n.a.
40	**124.17**	**124.17**	123.79	123.79	n.a.
100	**308.79**	**308.79**	308.40	308.72	**308.78**
		gridsmall			
10	**6.03**	**6.03**	4.99	5.31	**6.03**
20	**13.96**	13.90	11.16	11.49	13.93
40	**30.93**	29.89	22.56	23.54	28.55
100	**78.37**	78.31	56.92	47.79	75.92
		grid3x3			
10	**4.68**	**4.68**	**4.68**	**4.68**	**4.68**
20	**14.37**	**14.37**	13.37	**14.36**	**14.35**
40	**34.35**	**34.35**	**32.34**	**34.35**	**34.33**
100	**94.35**	OOT	92.34	94.32	94.24
		boxpushing			
10	**224.26**	219.19	54.69	223.48	223.74
20	**470.43**	441.98	123.59	254.41	458.10
40	**941.07**	918.62	236.79	283.79	636.28
100	**2366.21**	1895.16	599.97	560.16	n.a.
		mars			
10	**26.31**	24.47	17.90	17.63	**26.31**
20	**52.32**	52.20	34.43	35.27	52.13
40	**104.07**	103.25	67.74	65.94	103.52
100	**255.18**	OOT	152.52	124.73	249.92
		mabc			
10	**9.29**	**9.29**	9.20	9.20	**9.29**
20	**18.31**	**18.31**	18.10	18.20	n.a.
40	**36.46**	**36.46**	36.10	36.20	n.a.
100	**90.76**	**90.76**	90.20	90.20	**90.76**

Table 1: For each two-agent domain and planning horizon ℓ, we report the best value per algorithm. OOT means time limit of 1 hour has been exceeded and 'n.a.' is not available.

for oSARSA$^{\texttt{sim}}$. It improves the best known values provided by FB-HVI in all tested domains, except for recycling and mabc, where it achieves equivalent values. Moreover, the margin between oSARSA$^{\texttt{seq}}$ and FB-HSVI increases as the planning horizon increases. In boxpushing, the margin between oSARSA$^{\texttt{seq}}$ and FB-HSVI increases from 0.52 at planning horizon $\ell = 10$ to 304.79 at planning horizon $\ell = 40$. It is worth noticing that oSARSA$^{\texttt{sim}}$ achieves competitive performances against FB-HSVI in most tested two-agent domains, except mars and grid3x3, where it runs out of time. A2C and IQL are competitive on two weakly-coupled domains, *e.g.,* recycling and grid3x3, but get stuck in local optima in the other domains. Table 2 shows that oSARSA$^{\texttt{seq}}$

n	ℓ	oSARSA$^{\texttt{seq}}$	oSARSA$^{\texttt{sim}}$	A2C	IQL
			tiger		
3	10	**11.29**	OOT	−19.26	−9.82
4	10	**6.80**	OOT	−110.08	−18.48
5	2	**-4.00**	**-4.00**	−30.00	**-4.00**
5	4	**3.84**	OOT	−50.00	−7.01
5	6	**-0.16**	OOT	−93.83	−11.56
5	8	**-5.43**	OOT	−128.98	−14.97
5	10	**2.41**	OOT	−126.62	−131.91
			recycling		
3	10	**85.23**	**85.23**	45.83	47.51
4	10	**108.92**	105.96	57.70	55.49
5	10	**133.84**	**133.84**	74.50	72.88
6	10	**159.00**	OOT	86.37	79.83
7	2	**45.50**	OOT	18.20	20.80
7	4	**80.50**	OOT	57.11	40.84
7	6	**115.50**	OOT	64.49	59.06
7	8	**150.50**	OOT	72.97	71.84
7	10	**185.50**	OOT	85.07	91.07
			gridsmall		
3	10	**5.62**	OOT	0.34	0.57
4	2	**0.13**	**0.13**	0.02	**0.13**
4	4	**0.78**	OOT	0.22	0.55
4	6	**1.75**	OOT	0.39	0.77
4	8	**2.85**	OOT	0.65	1.11
4	10	**4.09**	OOT	1.50	1.84

Table 2: For each many-agent domain and planning horizon ℓ, we report the best value per algorithm. OOT means time limit of 1 hour has been exceeded.

outperforms oSARSA$^{\texttt{sim}}$, A2C and IQL on all tested many-agent domains over different planning horizons. For medium to large teams and planning horizons, oSARSA$^{\texttt{sim}}$ runs out of time. Even though A2C and IQL scale up with larger teams, they achieve poor performances against oSARSA$^{\texttt{seq}}$.

The empirical results of oSARSA$^{\texttt{seq}}$ are due to two interdependent reasons: (1) the piecewise linearity and convexity property of the value function over (sequential occupancy-) states and (2) the polynomial-time update rule. The uniform continuity property (1) generalizes values from seen states to unseen ones. Doing so guides algorithms toward the most promising parts of the search space while discarding others. Although uniform continuity holds for both oSARSA$^{\texttt{sim}}$ and oSARSA$^{\texttt{seq}}$ (albeit in different state spaces), its efficiency differs. In particular, oSARSA$^{\texttt{seq}}$ can discard parts of the search space at each decision step, but oSARSA$^{\texttt{sim}}$ can only do it after n decision steps. In other words, even if the search space is the same, oSARSA$^{\texttt{seq}}$ will often explore fewer states than oSARSA$^{\texttt{sim}}$. The exponential complexity drop enables oSARSA$^{\texttt{seq}}$ to perform exponentially faster episodes than oSARSA$^{\texttt{sim}}$, thus visiting many states that will, in return, help discard larger parts of the search space. However, each update in a single state in oSARSA$^{\texttt{sim}}$ is double exponential in the worst case. So, as the number of agents and actions per agent increases, performing even a single update (not to mention an episode) in oSARSA$^{\texttt{sim}}$ is prohibitive. This insight hinders its ability to exploit the uniform continuity property in larger instances.

7 Discussion

This paper highlights the need for a paradigm shift in ϵ-optimally solving Dec-POMDPs, transitioning from simultaneous- to sequential-move centralized training for decentralized execution. This shift addresses the silent coordination dilemma and the decision entanglement problems, which have hindered the scalability of ϵ-optimal methods. Specifically, it leads to an exponential drop in the complexity of backups, although it comes at the cost of longer planning horizons. It facilitates the application of efficient single-agent methods while preserving their theoretical guarantees, as evidenced by the superior performance of the oSARSA$^{\texttt{seq}}$ algorithm compared to all other tested baselines across domains involving two or more agents. Interestingly, this novel paradigm also addresses the multi-agent credit assignment problem, breaking down the optimal value functions among agents. This insight is important because this problem has recently attracted much attention in the multi-agent reinforcement learning community (Foerster et al. 2018; Rashid et al. 2018; Wang et al. 2023; Mahajan et al. 2019; Son et al. 2019; Wang et al. 2021; Li et al. 2021; Chen et al. 2024). Yet, with the exception of Kuba et al. (2022), all contributions suggest approximate solutions. Kuba et al. (2022) introduces a clever way of factorizing the joint value function leveraging advantage functions but limits its applicability to trust-region learning algorithms. The sequential-move centralized for decentralized control paradigm applies to all value- and policy-based algorithms.

Studies conducted by Cazenave (2010) and subsequent research by Bertsekas (2020) demonstrated that one can effectively solve both multi-agent path planning and multi-agent cooperative reinforcement learning problems, one agent at a time. However, Bertsekas (2020) limited their approach to sequential policies, thereby hindering the ability to derive optimal decentralized policies in general cases. Our research advances this field by applying a broader framework of Dec-POMDP while offering an optimal dynamic programming theory. This paradigm was applied successfully in subclasses of simultaneous-move Dec-POMDPs, namely two and many-player hierarchical information-sharing Dec-POMDPs (Xie, Dibangoye, and Buffet 2020; Peralez et al. 2024). Similarly, Koops et al. (2023) employs a sequential-move centralized training paradigm for decentralized control but confines the analysis to a tree-search algorithm, specifically the multi-agent A* (MAA*) (Szer and Charpillet 2006; Oliehoek, Spaan, and Vlassis 2008; Oliehoek et al. 2013). Additionally, Kovařík et al. (2022) demonstrated that simultaneous-move games could be treated as sequential-move problems through extensive-form games, although they did not provide a practical computational theory or algorithms. We posit that transitioning from simultaneous- to sequential-move centralized training for decentralized execution establishes a foundation for enhancing exact and approximate planning and reinforcement learning algorithms applicable to cooperative, competitive, and ultimately mixed-motive partially observable stochastic games.

Acknowledgements

This work was supported by ANR project Planning and Learning to Act in Systems of Multiple Agents under Grant ANR-19-CE23-0018, and ANR project Data and Prior, Machine Learning and Control under Grant ANR-19-CE23-0006, and ANR project Multi-Agent Trust Decision Process for the Internet of Things under Grant ANR-21-CE23-0016, all funded by French Agency ANR. J.C. was supported by SEFRI in the context of the HORIZON Europe Hyper-AI project (grant agreement No. 101135982).

References

Bellman, R. E. 1957. *Dynamic Programming*. Dover Publications, Inc.

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.

Bertsekas, D. 2020. Multiagent value iteration algorithms in dynamic programming and reinforcement learning. *Results in Control and Optimization*, 1.

Bono, G.; Dibangoye, J. S.; Matignon, L.; Pereyron, F.; and Simonin, O. 2018. Cooperative Multi-agent Policy Gradient. In *Proceedings of the Twenty-Eight European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, ECML/PKDD'18, 459–476.

Cazenave, T. 2010. Partial Move A*. In *Proceedings of the Twenty-Second International Conference on Tools with Artificial Intelligence*, ICTAI'10, 25–31.

Chen, S.; Zhang, Z.; Yang, Y.; and Du, Y. 2024. STAS: Spatial-Temporal Return Decomposition for Multi-agent Reinforcement Learning. In *Proceedings of the Thirty-Eigth AAAI Conference on Artificial Intelligence*, volume 16 of *AAAI'24*, 17337–17345.

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.

Dibangoye, J. S.; and Buffet, O. 2018. Learning to Act in Decentralized Partially Observable MDPs. In *Proceedings of the Thirty-Fifth International Conference on Machine Learning*, ICML'18, 1233–1242.

Foerster, J. N.; Farquhar, G.; Afouras, T.; Nardelli, N.; and Whiteson, S. 2018. Counterfactual Multi-Agent Policy Gradients. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligenc*, 2974–2982.

Hansen, E. A.; Bernstein, D. S.; and Zilberstein, S. 2004. Dynamic Programming for Partially Observable Stochastic Games. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, AAAI'04, 709–715.

Hauskrecht, M. 2000. Value-Function Approximations for Partially Observable Markov Decision Processes. *Journal of Artificial Intelligence Research*, 13: 33–94.

Konda, V. R.; and Tsitsiklis, J. N. 1999. Actor-Critic Algorithms. In *Advances in Neural Information Processing Systems 12*, NIPS'99, 1008–1014.

Koops, W.; Jansen, N.; Junges, S.; and Simão, T. D. 2023. Recursive small-step multi-agent A* for Dec-POMDPs. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, IJCAI'23, 5402–5410.

Kovařík, V.; Schmid, M.; Burch, N.; Bowling, M. H.; and Lisý, V. 2022. Rethinking Formal Models of Partially Observable Multiagent Decision Making. *Artificial Intelligence*, 303: 103645.

Kuba, J. G.; Chen, R.; Wen, M.; Wen, Y.; Sun, F.; Wang, J.; and Yang, Y. 2022. Trust Region Policy Optimisation in Multi-Agent Reinforcement Learning. In *Proceedings of the Tenth International Conference on Learning Representations*, ICLR'22.

Li, J.; Kuang, K.; Wang, B.; Liu, F.; Chen, L.; Wu, F.; and Xiao, J. 2021. Shapley Counterfactual Credits for Multi-Agent Reinforcement Learning. In *Proceedings of the Twenty-Seventh Conference on Knowledge Discovery & Data Mining*, KDD'21, 934–942.

Mahajan, A.; Rashid, T.; Samvelyan, M.; and Whiteson, S. 2019. MAVEN: Multi-Agent Variational Exploration. In *Advances in Neural Information Processing Systems 32*, NeurIPS'19, 7613–7624.

Oliehoek, F. A.; Spaan, M. T. J.; Amato, C.; and Whiteson, S. 2013. Incremental Clustering and Expansion for Faster Optimal Planning in Dec-POMDPs. *Journal of Artificial Intelligence Research*, 46: 449–509.

Oliehoek, F. A.; Spaan, M. T. J.; and Vlassis, N. A. 2008. Optimal and Approximate Q-value Functions for Decentralized POMDPs. *Journal of Artificial Intelligence Research*, 32: 289–353.

Ooi, J. M.; and Wornell, G. W. 1996. Decentralized control of a multiple access broadcast channel: Performance bounds. In *Proceedings of the Thirty-Fifth IEEE Conference on Decision and Control*, 293–298.

Peralez, J.; Delage, A.; Buffet, O.; and Dibangoye, J. S. 2024. Solving Hierarchical Information-Sharing Dec-POMDPs: An Extensive-Form Game Approach. In *Proceedings of the Forty-First International Conference on Machine Learning*, ICML'24, 40414–40438.

Rabinovich, Z.; Goldman, C. V.; and Rosenschein, J. S. 2003. The Complexity of Multiagent Systems: the Price of Silence. In *Proceedings of the Second International Conference on Autonomous Agents and Multiagent Systems*, AAMAS'03, 1102–1103.

Radner, R. 1962. Team Decision Problems. *The Annals of Mathematical Statistics*, 33(3): 857–881.

Rashid, T.; Samvelyan, M.; de Witt, C. S.; Farquhar, G.; Foerster, J. N.; and Whiteson, S. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the Thirty-Fifth International Conference on Machine Learning*, 4292–4301.

Rutenbar, R. A. 1989. Simulated Annealing Algorithms: An overview. *IEEE Circuits and Devices Magazine*, 5(1): 19–26.

Shoham, Y.; and Leyton-Brown, K. 2008. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press.

Smith, T. 2007. *Probabilistic Planning for Robotic Exploration*. Ph.D. thesis, The Robotics Institute, Carnegie Mellon University.

Sokota, S.; Lockhart, E.; Timbers, F.; Davoodi, E.; D'Orazio, R.; Burch, N.; Schmid, M.; Bowling, M. H.; and Lanctot, M. 2021. Solving Common-Payoff Games with Approximate Policy Iteration. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*, volume 11 of *AAAI'21*, 9695–9703.

Son, K.; Kim, D.; Kang, W. J.; Hostallero, D. E.; and Yi, Y. 2019. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. In *Proceedings of the Thirty-Sixth International Conference on Machine Learning*, ICML'19, 5887–5896.

Szer, D.; and Charpillet, F. 2006. Point-Based Dynamic Programming for DEC-POMDPs. In *Proceedings of the Twenty-First AAAI Conference on Artificial Intelligence*, volume 2 of *AAAI'06*, 1233 – 1238.

Szer, D.; Charpillet, F.; and Zilberstein, S. 2005. MAA*: A Heuristic Search Algorithm for Solving Decentralized POMDPs. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, UAI'05, 576–583.

Tan, M. 1993. Multi-agent reinforcement learning: independent versus cooperative agents. In *Proceedings of the Tenth International Conference on International Conference on Machine Learning*, ICML'93, 330–337.

Wang, J.; Ren, Z.; Liu, T.; Yu, Y.; and Zhang, C. 2021. QPLEX: Duplex Dueling Multi-Agent Q-Learning. In *Proceedings of the Nineth International Conference on Learning Representations*, ICLR'21.

Wang, Z.; Du, Y.; Zhang, Y.; Fang, M.; and Huang, B. 2023. MACCA: Offline Multi-agent Reinforcement Learning with Causal Credit Assignment. *CoRR*, abs/2312.03644.

Xie, Y.; Dibangoye, J. S.; and Buffet, O. 2020. Optimally solving two-agent decentralized pomdps under one-sided information sharing. In *Proceedings of the Thirty-Seventh International Conference on Machine Learning*, ICML'20, 10473–10482.

Yoshikawa, T.; and Kobayashi, H. 1978. Separation of Estimation and Control for Decentralized Stochastic Control Systems. *Automatica*, 14(6): 623–628.