Artificial Intelligence 175 (2011) 1757–1789

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Decentralized MDPs with sparse interactions

Francisco S. Melo [a,*], Manuela Veloso [b]

[a] GAIPS – INESC-ID, TagusPark, Edifício IST, 2780-990 Porto Salvo, Portugal
[b] Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA

ARTICLE INFO

Article history:
Received 26 April 2010
Received in revised form 29 April 2011
Accepted 7 May 2011
Available online 10 May 2011

Keywords:
Multiagent coordination
Sparse interaction
Decentralized Markov decision processes

ABSTRACT

Creating coordinated multiagent policies in environments with uncertainty is a challenging problem, which can be greatly simplified if the coordination needs are known to be limited to specific parts of the state space. In this work, we explore how such local interactions can simplify coordination in multiagent systems. We focus on problems in which the interaction between the agents is sparse and contribute a new decision-theoretic model for decentralized sparse-interaction multiagent systems, Dec-SIMDPs, that explicitly distinguishes the situations in which the agents in the team must coordinate from those in which they can act independently. We relate our new model to other existing models such as MMDPs and Dec-MDPs. We then propose a solution method that takes advantage of the particular structure of Dec-SIMDPs and provide theoretical error bounds on the quality of the obtained solution. Finally, we show a reinforcement learning algorithm in which independent agents learn both individual policies and when and how to coordinate. We illustrate the application of the algorithms throughout the paper in several multiagent navigation scenarios.

1. Introduction

Decision-theoretic models, such as Dec-MDPs and Dec-POMDPs, provide a rich framework to tackle decentralized decision-making problems. However, using these models to create coordinated multiagent policies in environments with uncertainty is a challenging problem, even more so if the decision-makers must tackle issues of partial observability. As such, solving finite-horizon Dec-POMDPs is a NEXP-complete problem and thus computationally too demanding to solve except for the simplest scenarios.

Recent years have witnessed a profusion of work on Dec-POMDP-related models that aim at capturing some of the fundamental features of this class of problems, such as partial observability, without incurring the associated computational cost. In this paper, we contribute to this area of research, and introduce a new model for cooperative multiagent decision-making in the presence of *partial observability*. Our model is motivated by the observation that, in many real-world scenarios, the tasks of the different agents in a multiagent system are not coupled at every decision step but only in relatively infrequent situations. We refer to such problems as having *sparse interactions*.

Multi-robot systems provide our primary motivation, as the interaction among different robots is naturally limited by each robot's physical boundaries, such as workspace or communication range, and limited perception capabilities. Therefore, when programming a multi-robot system to perform some task, one natural approach is to subdivide this task into smaller tasks that each robot can then execute autonomously or as part of a smaller group. As an example, consider the scenario in Fig. 1. In this scenario, three robots must navigate to their goal locations, marked with dashed lines. While Robot 3 can

* Corresponding author.
E-mail address: fmelo@inesc-id.pt (F.S. Melo).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2011.05.001

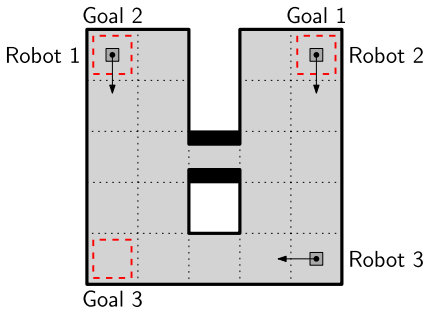


Fig. 1. Example of a simple navigation task.

navigate to its goal, disregarding the remaining robots, Robots 1 and 2 need to coordinate so as not to cross the narrow doorway simultaneously. However, this coordination needs only occur around the doorway.

Other examples include problems of sequential resource allocation, in which groups of agents must interact only to the extent that they need to share some common resource. In this context, several methods have been proposed that leverage sparse interactions by decomposing the global problem into several smaller local problems that can be solved more efficiently, and then combining the obtained solutions [1,2]. Such approaches, however, are not particularly concerned with partial observability issues. Additional examples include problems of task allocation, where the different agents in a multiagent system are assigned to different subtasks, and the interactions between the agents when performing such subtasks is localized to small regions of the joint state space [3]. Such problems include emergency response scenarios, where emergency teams are assigned different subtasks (for example, assisting different victims) and interact only in specific localized situations.

Several approaches have exploited simplified models of interaction in multiagent settings. For example, learning tasks involving multiple agents can be partitioned in a state-wise manner, allowing different agents to independently learn the resulting "smaller tasks" [4]. Similarly, a hierarchical learning algorithm can be used that considers only interactions between the different agents at a higher control level, while allowing the agents to learn lower level tasks independently [5]. Other works use coordination graphs to compactly represent dependences between the actions of different agents, thus capturing the local interaction between them [6,7]. Local interactions have also been exploited to minimize communication during policy execution [8] and in the game-theoretic literature to attain compact game representations. Examples include graphical games [9] and action-graph games [10].

In this article we consider Dec-MDPs with sparse interactions, henceforth Dec-SIMDPs. Dec-SIMDPs leverage the independence between agents to decouple the decision process in significant portions of the joint state space. In those situations in which the agents interact—the *interaction areas*—Dec-SIMDPs rely on communication to bring down the computational complexity of the joint decision process. Dec-SIMDPs balance the independence assumptions with observability: in any given state, the agents are either independent or can share state information (*e.g.*, by communicating).[1] A related model has recently been proposed under the designation of distributed POMDPs with coordination locales [13]. We postpone until Section 6 a more detailed discussion of this and other related models.

The contributions of this article are threefold. We provide a precise formalization of the Dec-SIMDP model and discuss the relation with well-established decision-theoretic models such as Dec-MDPs, MMDPs and MDPs. We then contribute two new algorithms that exhibit significant computational savings when compared to existing algorithms for Dec-SIMDPs, and illustrate their application in several simple navigation tasks. Finally, we investigate the influence of interaction areas in the performance of a multiagent system and contribute a new learning algorithm that allows each agent in one such system to individually learn those interaction areas.

2. Decision-theoretic models for multiagent systems

We now review several standard decision-theoretic models that are relevant for our work [14–16]. We start with single agent models, namely Markov decision processes (MDPs) and their partially observable counterparts (POMDPs) before moving to multiagent models such as multiagent MDPs (MMDPs) and their partially observable counterparts (Dec-POMDPs). We establish the notation we use and review some fundamental concepts of later relevance.

To fully specify the different models in this section, we should explicitly include the initial state x^0 or a distribution thereof. However, in order to avoid cluttering the notation, we omit the explicit reference to this initial state, with the understanding that one such state is implicit.

[1] Both independence assumptions and communication can significantly bring down the computational complexity in Dec-POMDP related models [11,12].

2.1. Markov decision processes

A *Markov decision process* (MDP) describes a sequential decision problem in which a single agent must choose an action at every time step to maximize some reward-based optimization criterion. We use MDPs in the Dec-SIMDP model proposed in Section 3 to describe an agent in those situations where its actions are independent of other agents—*i.e.*, in those situations where the agent can be modeled individually.

Formally, an MDP is a tuple $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$, where $\mathcal{X}$ represents the finite state space, $\mathcal{A}$ represents the finite action space, $\mathsf{P}(x, a, y)$ represents the transition probability from state x to state y when action a is taken, and $r(x, a)$ represents the expected reward for taking action a in state x. The scalar γ is a discount factor.

A Markov *policy* is a mapping $\pi : \mathcal{X} \times \mathcal{A} \to [0, 1]$ such that, for all $x \in \mathcal{X}$,

$$\sum_{a \in \mathcal{A}} \pi(x, a) = 1.$$

Solving an MDP consists of determining a policy π so as to maximize, for all $x \in \mathcal{X}$,

$$V^{\pi}(x) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r\big(X(t), A(t)\big) \;\middle|\; X(0) = x\right],$$

where $X(t)$ denotes the state at time step t, $A(t)$ denotes the action taken at that time instant such that

$$\mathbb{P}\big[A(t) = a \;\big|\; H(t) = h\big] = \mathbb{P}\big[A(t) = a \;\big|\; X(t) = x\big] = \pi(x, a),$$

where $H(t) = \{X(0), A(0), \ldots, X(t-1), A(t-1), X(t)\}$ is the random variable corresponding to the history of the MDP up to time t, and h denotes a particular realization of $H(t)$ such that $X(t) = x$. We write $A(t) \sim \pi$ to denote the above dependence of $A(t)$ on the policy π. We define the Q-function associated with a policy π as

$$Q^{\pi}(x, a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r\big(X(t),\ A(t)\big) \;\middle|\; X(0) = x, A(0) = a\right],$$

where, again, $A(t) \sim \pi$ for all $t > 0$.

For any finite MDP, there is at least one *optimal policy* π^*, such that

$$V^{\pi^*}(x) \geqslant V^{\pi}(x)$$

for every policy π and every state $x \in \mathcal{X}$. The value function corresponding to π^* is denoted by V^* (short for V^{π^*}) and verifies the Bellman optimality equation,

$$V^*(x) = \max_{a \in \mathcal{A}}\left[r(x, a) + \gamma \sum_{y \in \mathcal{X}} \mathsf{P}(x, a, y) V^*(y)\right]. \tag{1}$$

The associated Q-function in turn verifies

$$Q^*(x, a) = r(x, a) + \gamma \sum_{y \in \mathcal{X}} \mathsf{P}(x, a, y) \max_{u \in \mathcal{A}} Q^*(y, u). \tag{2}$$

The optimal policy can be recovered directly from Q^* by assigning an action $a \in \mathcal{A}$ a positive probability of being selected in state $x \in \mathcal{X}$ only if $a \in \arg\max_{u \in \mathcal{A}} Q^*(x, u)$. As such, the solution for any given MDP can be obtained by computing the corresponding optimal Q-function, Q^*.

Given a function q defined over $\mathcal{X} \times \mathcal{A}$, the *Bellman operator* $\mathbf{H}$ is defined as

$$(\mathbf{H}q)(x, a) = r(x, a) + \gamma \sum_{y \in \mathcal{X}} \mathsf{P}(x, a, y) \max_{u \in \mathcal{A}} q(y, u). \tag{3}$$

The function Q^* in (2) is the fixed-point of $\mathbf{H}$ and thus can be computed, *e.g.*, by iteratively applying $\mathbf{H}$ to some initial estimate $Q^{(0)}$, a dynamic programming (DP) method known as *value iteration*.

2.1.1. Q-learning

It is possible to use the operator $\mathbf{H}$ in (3) and use a standard fixed-point iteration to compute Q^*. If P and/or r are unknown, Q^* can be estimated using the *Q-learning algorithm* [17]. Q-learning allows Q^* to be estimated from transitions consisting of state–action–reward–next state experienced in the environment, and is defined by the update rule

$$Q_{k+1}(x, a) = (1 - \alpha_k) Q_k(x, a) + \alpha_k\Big[r(x, a) + \gamma \max_{u \in \mathcal{A}} Q_k(y, u)\Big], \tag{4}$$

where $Q_k(x,a)$ is the kth estimate of $Q^*(x,a)$, y is a sample from the distribution $\mathsf{P}(x,a,\cdot)$ and $\{\alpha_k\}$ is a step-size sequence. The sample y and the value of $r(x,a)$ can be obtained from a generative model or from the actual system, not requiring the knowledge of either P or r. Under suitable conditions, the estimates Q_k converge to Q^* with probability 1 [18]. We henceforth write $\texttt{QLUpdate}(Q\,;x,a,r,y,Q')$ to compactly denote the general Q-learning update operation in (4), *i.e.*,

$$Q_{\text{updt}}(x,a) = \texttt{QLUpdate}\big(Q\,;x,a,r,y,Q'\big) \triangleq (1-\alpha_k)\,Q\,(x,a) + \alpha_k\Big[r + \gamma \max_{u\in\mathcal{A}} Q'(y,u)\Big],$$

where Q and Q' are general Q-functions.

2.2. Partially observable Markov decision processes

Partially observable MDPs describe problems essentially similar to MDPs, in which an agent must choose an action at every time step to maximize a reward-based criterion. However, at each time step t, the agent in a POMDP cannot directly observe the state of the process, $X(t)$, but accesses only an indirect observation thereof, $Z(t)$. POMDPs will be useful when planning for Dec-SIMDPs in Section 4. Taking advantage of the particular structure of Dec-SIMDPs, we reduce the global planning problem to the problem of planning in a collection of individual single-agent POMDPs.

Formally, a POMDP is a tuple $(\mathcal{X},\mathcal{A},\mathcal{Z},\mathsf{P},\mathsf{O},r,\gamma)$, where $\mathcal{X}$ is the finite state space, $\mathcal{A}$ is the finite action space and $\mathcal{Z}$ is the finite observation space. As before, $\mathsf{P}(x,a,y)$ represents the transition probability from state x to state y when action a is taken, and now $\mathsf{O}(x,a,z)$ represents the probability of observing $z\in\mathcal{Z}$ given that the state is x and action a was taken, *i.e.*,

$$\mathsf{O}(x,a,z) = \mathbb{P}\big[Z(t+1)=z \;\big|\; X(t+1)=x,\ A(t)=a\big].$$

Finally, $r(x,a)$ again represents the expected reward for taking action a in state x and γ is a discount factor.

A non-Markov policy is a mapping $\pi:\mathcal{H}\to[0,1]$ such that, for all finite histories $h\in\mathcal{H}$,

$$\sum_{a\in\mathcal{A}} \pi(h,a) = 1,$$

where $h=\{a(0),z(1),a(1),\ldots,a(t-1),z(t)\}$ is a *finite history*, *i.e.*, a finite sequence of action–observation pairs. As before, the history observed up to time t is a random variable, denoted as $H(t)$, and we denote by $\mathcal{H}$ the set of all possible finite histories for the POMDP. Solving a POMDP consists of determining a policy π so as to maximize, for all $|\mathcal{X}|$-dimensional probability vectors $\mathbf{b}$,

$$V^{\pi}(\mathbf{b}) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^t r\big(X(t),A(t)\big) \;\big|\; X(0)\sim\mathbf{b}\right],$$

where $X(0)\sim\mathbf{b}$ denotes the fact that $X(0)$ is distributed according to $\mathbf{b}$. Similarly, we define the Q-function associated with a policy π as

$$Q^{\pi}(\mathbf{b},a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^t r\big(X(t),A(t)\big) \;\big|\; X(0)\sim\mathbf{b},\ A(0)=a\right].$$

We refer to the probability vector $\mathbf{b}$ in the expressions above as the *initial belief state* or just the *initial belief*. It translates the "belief" that the agent has at time $t=0$ regarding its state at that time. From the initial belief and given the history up to time t, $H(t)$, we can construct a sequence $\{\mathbf{b}(t)\}$ of probability vectors recursively as

$$\mathbf{b}_y(t+1) = \mathsf{Bel}\big(b(t),A(t),Z(t+1)\big) \triangleq \eta\sum_x \mathbf{b}_x(t)\mathsf{P}\big(x,A(t),y\big)\mathsf{O}\big(y,A(t),Z(t+1)\big),$$

where Bel is the belief update operator, $\mathbf{b}_x(t)$ denotes the x component of $\mathbf{b}(t)$ and η is a normalization factor. We refer to the vector $\mathbf{b}(t)$ as the *belief* at time t. It corresponds to a distribution over the unknown state at time t, such that

$$\mathbf{b}_x(t) = \mathbb{P}\big[X(t)=x \;\big|\; H(t)\big].$$

Given the POMDP model parameters P and O, every finite history $h\in\mathcal{H}$ can be mapped to a belief $\mathbf{b}$. Moreover, for any given policy, two histories leading to the same belief will yield the same future total expected discounted reward. As such, beliefs can be used as compact representations of histories, and we can define policies in terms of beliefs instead of histories [19]. In fact, it is possible to reinterpret a POMDP as an infinite MDP in which the state space corresponds to the set of all possible beliefs. Therefore, for any finite POMDP, there is at least one *optimal policy* π^*, such that

$$V^{\pi^*}(\mathbf{b}) \geqslant V^{\pi}(\mathbf{b})$$

for any π and every initial belief $\mathbf{b}$. The corresponding value function is denoted by V^* and also verifies the Bellman optimality equation,

$$V^*(\mathbf{b}) = \max_{a\in\mathcal{A}} \sum_x \mathbf{b}_x \left[r(x,a) + \gamma \sum_{z,y} \mathsf{P}(x,a,y)\mathsf{O}(y,a,z) V^*\big(\mathsf{Bel}(\mathbf{b},a,z)\big)\right],$$

where x, y take values in $\mathcal{X}$, z takes values in $\mathcal{Z}$ and $\mathsf{Bel}(\mathbf{b},a,z)$ corresponds to the updated belief after taking action a and observing z. The associated Q-function in turn verifies

$$Q^*(\mathbf{b},a) = \sum_x \mathbf{b}_x \left[r(x,a) + \gamma \sum_{z,y} \mathsf{P}(x,a,y)\mathsf{O}(y,a,z) \max_{u\in\mathcal{A}} Q^*\big(\mathsf{Bel}(\mathbf{b},a,z),u\big)\right].$$

In spite of their representative power, POMDPs have been shown to be undecidable in the worst case for infinite horizon settings such as those considered in this paper [20]. As such, exact solutions can be computed only in very specific instances, and most approaches in the literature resort to approximate or heuristic methods [21,22].

We now describe an MDP-based heuristic solution method for POMDPs that we use later in the paper. This method is known as Q_{MDP} as it makes use of the optimal Q-function for the underlying MDP as an estimate for the optimal Q-function for the POMDP [23]. Since the optimal solution for the underlying MDP can be efficiently computed, this method is very simple and fast to implement and attains good performance in many practical situations [21,23].

Let $\mathcal{M} = (\mathcal{X},\mathcal{A},\mathcal{Z},\mathsf{P},\mathsf{O},r,\gamma)$ be a POMDP with finite state, action and observation spaces. Associated with this POMDP, there is an underlying MDP $\bar{\mathcal{M}} = (\mathcal{X},\mathcal{A},\mathsf{P},r,\gamma)$. Let $\bar{Q}^*$ be the optimal Q-function for $\bar{\mathcal{M}}$. Q_{MDP} uses an estimate for the optimal Q-function for the POMDP given by

$$\hat{Q}(\mathbf{b},a) = \sum_x \mathbf{b}_x \bar{Q}^*(x,a).$$

When using Q_{MDP}, the agent acts under the implicit assumption that state uncertainty only affects the immediate decision, *i.e.*, after one decision the agent will act on the underlying MDP. Since such assumption seldom holds, Q_{MDP} sometimes exhibits poor performance and several works have proposed further improvements on Q_{MDP} to address this issue [21,24], but we do not pursue such discussion here.

2.3. Multiagent MDPs

Multiagent Markov decision processes (MMDPs) generalize MDPs to multiagent cooperative scenarios. MMDPs describe sequential decision tasks in which multiple agents must each choose an individual action at every time-step that jointly maximize some common reward-based optimization criterion. Formally, an MMDP is a tuple $\mathcal{M} = (N,\mathcal{X},(\mathcal{A}_k),\mathsf{P},r,\gamma)$, where N is the number of agents, $\mathcal{X}$ represents the finite state space, $\mathcal{A}_k$ is the finite *individual* action space of agent k. As in MDPs, $\mathsf{P}(x,a,y)$ represents the transition probability from state x to state y when the *joint action* $a = (a_1,\ldots,a_N)$ is taken; $r(x,a)$ represents the expected reward received by *all* agents for taking the joint action a in state x. In an MMDP all agents receive the same reward, which implies that MMDPs represent *fully cooperative* multiagent tasks.

A joint action a is a tuple $a = (a_1,\ldots,a_N)$ and we denote by $\mathcal{A} = \times_{k=1}^N \mathcal{A}_k$ the set of all possible joint actions—the joint action space. For $k = 1,\ldots,N$, we write

$$\mathcal{A}_{-k} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_{k-1} \times \mathcal{A}_{k+1} \times \cdots \times \mathcal{A}_N$$

to denote the set of joint actions of all agents other than agent k. We write a_{-k} to denote a general element of $\mathcal{A}_{-k}$ and refer to any such action as a *reduced joint action* or simply a reduced action. We write $a = (a_{-k},a_k)$ to denote the fact that the joint action a is composed by the reduced action a_{-k} and the individual action a_k for agent k.

We also assume that the state space $\mathcal{X}$ can be factored among the N agents as $\mathcal{X} = \mathcal{X}_0 \times \mathcal{X}_1 \times \cdots \times \mathcal{X}_N$, where $\mathcal{X}_0$ denotes an agent-independent component of the state. As such, each element $x \in \mathcal{X}$ is a tuple $x = (x_0,\ldots,x_N)$, with $x_k \in \mathcal{X}_k$, $k = 0,\ldots,N$. For any $x \in \mathcal{X}$, we refer to the pair (x_0,x_k) as the local state of agent k, and generally denote it as $\bar{x}_k$.

An *individual* Markov policy for agent k is a mapping $\pi_k : \mathcal{X}\times\mathcal{A}_k \to [0,1]$ such that, for all $x \in \mathcal{X}$,

$$\sum_{a_k\in\mathcal{A}_k} \pi_k(x,a_k) = 1.$$

Similarly, a *joint* policy is a mapping $\pi : \mathcal{X}\times\mathcal{A} \to [0,1]$ that we take as the combination of N individual policies, *i.e.*,

$$\pi(x,a) = \prod_{k=1}^N \pi_k(x,a_k),$$

where $a = (a_1,\ldots,a_N)$; π_{-k} denotes a *reduced policy* and $\pi = (\pi_{-k},\pi_k)$ denotes the fact that the joint policy π is composed by the reduced policy π_{-k} and the individual policy π_k for agent k.

In an MMDP, the purpose of *all* agents is to determine a joint policy π so as to maximize, for all $x \in \mathcal{X}$,

$$V^{\pi}(x) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r\big(X(t), A(t)\big) \;\big|\; X(0) = x\right],$$

where $X(t)$ denotes the state at time t and $A(t)$ denotes the joint action taken at time t. The Q-function associated with a joint policy π is defined from V^{π} as its single-agent counterpart.

For the purposes of planning, *i.e.*, computing the optimal policy, an MMDP is indistinguishable from an ordinary MDP. In fact, when considering a centralized controller, an MMDP reduces to an MDP. It is only at execution time that an MMDP differs from an MDP, since the process of decision making is not centralized. However, when we move to partially observable settings, decentralized execution raises severe difficulties even during planning.

2.4. Decentralized POMDPs

Decentralized POMDPs (Dec-POMDPs) are partially observable generalizations of MMDPs. As in MMDPs, the agents in a Dec-POMDP must each choose an individual action at every time step that jointly maximize some common reward-based optimization criterion. Unlike MMDPs, however, the agents can only access the global state of the process by means of local indirect observations. Formally, a Dec-POMDP can be represented a tuple

$$\mathcal{M} = \big(N, (\mathcal{X}_k), (\mathcal{A}_k), (\mathcal{Z}_k), \mathsf{P}, (\mathsf{O}_k), r, \gamma\big),$$

where N is the number of agents, $\mathcal{X} = \times_{k=0}^{N} \mathcal{X}_k$ is the joint state space, $\mathcal{A} = \times_{k=1}^{N} \mathcal{A}_k$ is the set of joint actions, each $\mathcal{Z}_k$ represents the set of possible local observations for agent k, $\mathsf{P}(x, a, y)$ represents the transition probabilities from joint state x to joint state y when the joint action a is taken, each $\mathsf{O}_k(x, a, z_k)$ represents the probability of agent k making the local observation z_k when the joint state is x and the last action taken was a, and $r(x, a)$ represents the expected reward received by all agents for taking the joint action a in joint state x. The scalar γ is a discount factor.

In this partially observable multiagent setting, an individual non-Markov policy for agent k is a mapping $\pi_k : \mathcal{H}_k \to [0, 1]$ such that, for all $h_k \in \mathcal{H}_k$,

$$\sum_{a_k \in \mathcal{A}_k} \pi(h_k, a_k) = 1,$$

where $h_k = \{a_k(0), z_k(1), \ldots, a_k(t-1), z_k(t)\}$ is an *individual history* for agent k, *i.e.*, a sequence of individual action–observation pairs. $\mathcal{H}_k$ is the set of all possible finite histories for agent k and we denote by $H_k(t)$ the random variable that represents the history of agent k at time t.

Like in POMDPs, in a Dec-POMDP each agent has only partial perception of the global state. Therefore, from the agent's perspective, its local observations are *non-Markovian*—the current local observation and action are not sufficient to uniquely determine its next observation. In the general multiagent setting, however, there is no compact representation of histories that plays the role of beliefs in POMDPs, implying that the passage from MMDP to its partially observable counterpart is fundamentally different from the same passage in single-agent scenarios.[2] However, if communication between the agents is instantaneous, free and error-free, then a Dec-POMDP reduces to a "large" POMDP, and partial observability is no longer an issue.

Solving a Dec-POMDP consists in determining a joint policy π that maximizes the total sum of discounted rewards. In order to write this in terms of a function, we consider a distinguished initial state, $x^0 \in \mathcal{X}$, that is assumed common knowledge among all agents. We thus want to maximize

$$V^{\pi} = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r\big(X(t), A(t)\big) \;\big|\; X(0) = x^0\right].$$

Dec-POMDPs constitute one of the most representative models in the decision-theoretic literature [26]. In the remainder of this section we discuss several specializations of this model that are relevant for this work.

2.4.1. Decentralized MDPs

A *decentralized MDP* (Dec-MDP) is a particular case of a Dec-POMDP in which, for every joint observation $z \in \mathcal{Z}$, there is a state $x \in \mathcal{X}$ such that

$$\mathbb{P}\big[X(t) = x \;\big|\; Z(t) = z\big] = 1.$$

[2] This fact can also be observed by considering the worst-case computational complexity of each of the different models. In finite horizon settings, POMDPs are PSPACE-complete, versus the P-completeness of fully observable MDPs [25]. In finite-horizon multiagent settings, however, Dec-MDPs are NEXP-complete [16] even in the 2-agent case, versus the P-completeness of MMDPs.

Intuitively, the agents in a Dec-MDP have *joint full observability*: if all agents share their observations, they can recover the state of the Dec-MDP unambiguously. This is in contrast with MMDPs where each agent, individually, has full observability. In fact, MMDPs can be seen as a particular subclass of Dec-MDPs in which the individual observations of each agent allow it to recover the state of the Dec-MDP/MMDP unambiguously. Formally, for every individual observation $z_k \in \mathcal{Z}_k$, there is a state $x \in \mathcal{X}$ such that

$$\mathbb{P}\big[X(t) = x \;\big|\; Z_k(t) = z_k\big] = 1. \tag{5}$$

Throughout this work, we focus on Dec-MDPs and specializations thereof. Moreover, as with MMDPs, we assume that the state can be factored and the agents have *local full observability*, meaning that each agent can infer from its local observations the corresponding local state unambiguously. Formally local full observability can be translated into the following condition: for every local observation $z_k \in \mathcal{Z}_k$ there is a local state $\bar{x}_k \in \mathcal{X}_0 \times \mathcal{X}_k$ such that

$$\mathbb{P}\big[\bar{X}_k(t) = \bar{x}_k \;\big|\; Z_k(t) = z_k\big] = 1.$$

Although more general Dec-MDP models are possible [16], we adhere to this simplified version, as this is sufficient for our purposes. For future reference, we define the set

$$\mathcal{X}_{-k} = \mathcal{X}_0 \times \cdots \times \mathcal{X}_{k-1} \times \mathcal{X}_{k+1} \times \cdots \times \mathcal{X}_N$$

corresponding to the local state of all agents other than agent k, and denote by x_{-k} a general element of $\mathcal{X}_{-k}$. As with actions, we write $x = (x_{-k}, x_k)$ to denote the fact that the kth component of x takes the value x_k.

2.4.2. Transition, observation and reward independence

Transition-independent Dec-MDPs constitute a particular subclass of Dec-MDPs in which, for all $(x, a) \in \mathcal{X} \times \mathcal{A}$,

$$\mathbb{P}\big[X_0(t+1) = y_0 \;\big|\; X(t) = x,\ A(t) = a\big] = \mathbb{P}\big[X_0(t+1) = y_0 \;\big|\; X_0(t) = x_0\big], \tag{6a}$$

$$\mathbb{P}\big[X_k(t+1) = y_k \;\big|\; X(t) = x,\ A(t) = a\big] = \mathbb{P}\big[X_k(t+1) = y_k \;\big|\; \bar{X}_k(t) = \bar{x}_k,\ A_k(t) = a_k\big]. \tag{6b}$$

The transition probabilities can thus be factorized as

$$\mathsf{P}(x, a, y) = \mathsf{P}_0(x_0, y_0) \prod_{k=1}^{N} \mathsf{P}_k(\bar{x}_k, a_k, y_k), \tag{7}$$

where

$$\mathsf{P}_0(x_0, y_0) = \mathbb{P}\big[X_0(t+1) = y_0 \;\big|\; X_0(t) = x_0\big],$$
$$\mathsf{P}_k(\bar{x}_k, a_k, y_k) = \mathbb{P}\big[X_k(t+1) = y_k \;\big|\; \bar{X}_k(t) = \bar{x}_k,\ A_k(t) = a_k\big].$$

This particular class of Dec-MDPs was introduced in [27] and seeks to exploit a particular form of independence to bring down the computational complexity required to solve such models. In this class of problems, the local state of each agent constitutes a sufficient statistic for its history, and the optimal policy for each agent can thus be computed in terms of this individual state [12]. This particular class of Dec-MDPs has been shown to be NP-complete in finite-horizon settings, versus the NEXP-completeness of general finite-horizon Dec-MDPs [12].

Similarly, *reward-independent Dec-MDPs* correspond to a subclass of Dec-MDPs in which, for all x, a,

$$r(x, a) = f\big(r_k(\bar{x}_k, a_k),\ k = 1, \ldots, N\big), \tag{8}$$

i.e., the global reward function r can be obtained from local reward functions $r_k, k = 1, \ldots, N$. To ensure consistency of the decision process, we also require that

$$f\big(r_{-k}(x_{-k}, a_{-k}), r_k(\bar{x}_k, a_k)\big) \geqslant f\big(r_{-k}(x_{-k}, a_{-k}), r_k(\bar{x}_k, u_k)\big)$$

if and only if

$$r_k(\bar{x}_k, a_k) \geqslant r_k(\bar{x}_k, u_k).$$

One typical example is

$$r(x, a) = \sum_{k=1}^{N} r_k(\bar{x}_k, a_k), \tag{9}$$

where $x = (x_0, \ldots, x_N)$ and $a = (a_1, \ldots, a_N)$. Interestingly, it was recently shown [28,11] that reward independent Dec-MDPs retain NEXP-complete complexity. However, when associated with transition independence, reward independence implies that a Dec-MDP can be decomposed into N independent MDPs, each of which can be solved separately. The complexity of this class of problems thus reduces to that of standard MDPs (P-complete).

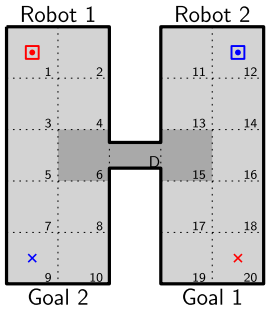


Fig. 2. Scenario in Example 1. Two robots must coordinate around a narrow doorway.

3. Decentralized sparse-interaction MDPs (Dec-SIMDPs)

We now depart from the transition-independent Dec-MDP and introduce a new model for multiagent decision problems that is, at the same time, more general and more specific than transition independent Dec-MDPs. Our goal is to exploit sparse interactions among the different agents in a Dec-MDP and so we are interested in Dec-MDPs in which there is some level of both transition and reward dependency, but this dependency is limited to specific regions of the state space.

To motivate our proposed model, we start with a simple example that is used throughout the section to illustrate the several concepts introduced.

Example 1. Consider the scenario depicted in Fig. 2, corresponding to an environment consisting of two rooms of 10 cells each connected by a narrow doorway D. In this scenario, two mobile robots acting as a team, Robot 1 and Robot 2, must each navigate from their depicted initial positions to the corresponding goal positions, marked in Fig. 2 with the labels Goal 1 and Goal 2, respectively. Each robot has available 4 actions, N, S, E, and W that move to the adjacent cell in the corresponding direction with probability of 0.8, and fail with probability 0.2. In case of failure, the robot remains in the same cell. Whenever a robot reaches the corresponding goal position, the team is granted a reward of $+1$. Conversely, if both robots stand simultaneously in the narrow doorway, corresponding to the cell marked as "D", the robots bump into each other and the team is granted a reward of -20. Also, in this situation, the robots get in the way of one another, and the probability of an individual action failing is 0.4. Each robot is able to perceive unambiguously its own position in the environment. Also, when both robots simultaneously stand in the shaded area in Fig. 2, they are able to see each other and can thus perceive their joint state unambiguously.

This problem can be modeled as a Dec-POMDP

$$\mathcal{M}_H = \big(2, (\mathcal{X}_k), (\mathcal{A}_k), (\mathcal{Z}_k), \mathsf{P}, (\mathsf{O}_k), r, \gamma\big),$$

where

- $\mathcal{X}_k = \{1, \ldots, 20, \mathsf{D}\}$, for $k = 1, 2$;
- $\mathcal{A}_k = \{N, S, E, W\}$, $k = 1, 2$;
- $\mathcal{Z}_k = \mathcal{X}_k \cup \mathcal{X}^I$, $k = 1, 2$, where $\mathcal{X}^I = \{6, 15, \mathsf{D}\} \times \{6, 15, \mathsf{D}\}$;
- the transition and observation functions can be derived from the description above;
- the reward function r is given by

$$r(x, a) = \begin{cases} 2 & \text{if } x = (20, 9), \\ 1 & \text{if } x_1 = 20 \text{ or } x_2 = 9, \\ -20 & \text{if } x = (\mathsf{D}, \mathsf{D}), \\ 0 & \text{otherwise.} \end{cases}$$

The reward function of this example is purposefully designed to include a region in the state space of negative reward where the two robots must coordinate. Outside this interaction area (say, the shaded region in Fig. 2), each robot can pursue its own goal, disregarding the existence of other robot.

The Dec-SIMDP model that we introduce explicitly addresses situations in which the interaction between the agents in a Dec-MDP is localized, much as in Example 1. In order to define our model that leverages such local interactions, we must first formalize concepts such as agent dependence and agent interaction. These definitions refine the notions of transition and reward independence described in Section 2.4.

Let $K = \{k_1, \ldots, k_m\}$ be a subset of agents in a Dec-MDP. We denote by

$$\mathcal{X}_K = \mathcal{X}_0 \times \mathcal{X}_{k_1} \times \cdots \times \mathcal{X}_{k_m}$$

the joint state space of all agents in K. Extending the notation introduced in Section 2, we write $\mathcal{X}_{-K}$ to denote the joint state space of the agents not in K. We write x_K to denote a general element of $\mathcal{X}_K$ and x_{-K} to denote a general element of $\mathcal{X}_{-K}$. We write $x = (x_{-K}, x_K)$ to distinguish the components of x corresponding to agents in K and those corresponding to agents not in K.

We assume that the reward function in our Dec-MDP to be decomposed as

$$r(x,a) = \sum_{k=1}^{N} r_k(\bar{x}_k, a_k) + \sum_{i=1}^{M_R} r_i^I(x_{K_i}, a_{K_i}), \tag{10}$$

where each r_k corresponds to an individual component of the reward function that depends only on agent k and there are M_R sets, K_i, $i = 1, \ldots, M_R$, and M_R reward components, r_i^I (the interaction components), each depending on all the agents in K_i and only on these.

Similarly, we assume that the transition probabilities for our Dec-MDP can be decomposed as

$$\mathsf{P}(x,a,y) = \mathsf{P}_0(x_0,y_0) \prod_{k=1}^{N} \big(1 - \mathbb{I}_{K_i}(k)\mathbb{I}_{\mathcal{X}_{K_i}^I}(x_{K_i})\big)\mathsf{P}_k(\bar{x}_k, a_k, y_k) + \mathsf{P}_0(x_0,y_0) \prod_{i=1}^{M_P} \mathsf{P}_{K_i}(x_{K_i}, a_{K_i}, y_{K_i})\mathbb{I}_{\mathcal{X}_{K_i}^I}(x_{K_i}), \tag{11}$$

where each $\mathcal{X}_{K_i}^I$ includes those states in $\mathcal{X}_{K_i}$ where the transition probabilities for each of the agents $k \in K_i$ cannot be factorized as in (7). We write $\mathbb{I}_U(x)$ to denote the indicator function for set U.

Expressions (10) and (11) explicitly distinguish the components of the reward and transition functions that can be factorized from those that cannot. The decompositions in (10) and (11) imply no loss of generality, since any reward r can be trivially written in that form by setting $M_R = 1$, $r_k \equiv 0$, $K_1 = \{1, \ldots, N\}$, and $r_1^I = r$, and the same occurs with the transition probabilities, by setting $M_P = 1$, $K_1 = \{1, \ldots, N\}$ and $\mathsf{P}_1^I = \mathsf{P}$. However, our interest is not in the general case, but when the supports of $\sum_{i=1}^{M_R} r_i^I$ and $\mathsf{P}_0 \prod_{i=1}^{M_P} \mathsf{P}_{K_i}\mathbb{I}_{\mathcal{X}_{K_i}^I}$ —*i.e.*, the subset of $\mathcal{X} \times \mathcal{A}$ in which these quantities are non-zero—are small when compared with $\mathcal{X} \times \mathcal{A}$.

Definition 3.1 *(Agent independence).* Given a Dec-MDP $\mathcal{M}$, an agent k_0 is *independent* of agent k_1 in a state $x \in \mathcal{X}$ if the following conditions both hold at state x:

- It is possible to decompose the global reward function $r(x,a)$ as in (10) in such a way that no agent set K_i contains both k_0 and k_1.
- It is possible to decompose the transition probabilities $\mathsf{P}(x,a,y)$ as in (11) in such a way that no agent set K_i contains both k_0 and k_1.

When any of the above conditions does not hold, agent k_0 is said to *depend* on agent k_1 in x. Similarly, agent k_0 is independent of a set of agents $K = \{k_1, \ldots, k_m\}$ at state x if the above conditions hold for all $k \in K$ in state x, and dependent otherwise.

An agent k_0 depends on another agent k_1 in a particular state if either the reward function or the transition probabilities or both cannot be decomposed so as to decouple the influence of each agent's individual state and action. However, the dependence relation is not symmetrical: even if agent k_0 depends on agent k_1, the reverse need not hold. In fact, it is possible that the two agents are reward independent and that the transition probabilities from state x_{k_0} depend on the individual state/action of agent k_1 without the reverse holding.

We illustrate the notion of agent independence using Example 1.

Example 1 *(Cont.).* 1 In the Dec-MDP $\mathcal{M}_H$, as previously introduced, it is possible to write the reward function r as

$$r(x,a) = r_1(x_1,a_1) + r_2(x_2,a_2) + r^I(x,a),$$

where $r_1(x_1,a_1) = \mathbb{I}_{20}(x_1)$, $r_2(x_2,a_2) = \mathbb{I}_9(x_2)$ and $r^I(x,a) = \mathbb{I}_{(\mathsf{D},\mathsf{D})}(x)$. We write $\mathbb{I}_x(\cdot)$ to denote the indicator function for the set $\{x\}$. In state (D,D), Robot 1 and Robot 2 both depend on the other, as seen in the transition probabilities. In the remaining joint states in $\mathcal{X}$, Robot 1 and Robot 2 are independent.

Definition 3.2 *(Agent interaction).* In a Dec-MDP $\mathcal{M}$, a set of agents K *interact* at state $x \in \mathcal{X}$ if the following conditions all hold

- For all $k_0 \in K$, if agent k_0 depends on some agent k_1 in state x, then $k_1 \in K$.
- For all $k_1 \in K$, if there is an agent k_0 that depends on agent k_1 in state x, then $k_0 \in K$.
- There is no strict subset $K' \subset K$ such that the above conditions hold for K'.

If the agents in a set K interact in a state x, then we refer to x_K as an *interaction state* for the agents in K.

The concept of interaction introduced above captures a local dependence between a set of agents in a Dec-MDP. If x_K is an interaction state for the agents in K, this does not mean that each agent k in K depends on all other agents in that state x. Instead, it means that there is at least one agent in K that either depends on k or k depends on it. Furthermore, the definition of agent interaction is transitive: if agents k_0 and k_1 interact at some state x and agents k_1 and k_2 interact at that same state x, then agents k_0 and k_2 interact at x.

We again resort to Example 1 to illustrate the concept of agent interaction.

Example 1 *(Cont.).* Robot 1 and Robot 2 interact at state (D,D). In fact, as seen above, not only does Robot 1 depend on Robot 2, but the converse also holds. Also, there is no strict subset of $\{1,2\}$ such that the conditions of Definition 3.2 hold.

Definition 3.3 *(Interaction area).* Consider a general N-agent Dec-MDP $\mathcal{M} = (N, (\mathcal{X}_k), (\mathcal{A}_k), (\mathcal{Z}_k), \mathsf{P}, (\mathsf{O}_k), r, \gamma)$. We define an *interaction area* $\mathcal{X}^I$ as verifying:

(i) $\mathcal{X}^I \subset \mathcal{X}_K$, for some set of agents K;
(ii) there is at least one state $x^* \in \mathcal{X}^I$ such that x^* is an interaction state for the agents in K;
(iii) for any $x \in \mathcal{X}^I$, $a_K \in \mathcal{A}_K$ and $y \notin \mathcal{X}^I$,

$$\mathbb{P}\big[X_K(t+1) = y \;\big|\; X_K(t) = x,\ A_K(t) = a_K\big] = \mathsf{P}_0(x_0, y_0) \prod_{k\in K} \mathsf{P}(x_k, a_k, y_k); \tag{12}$$

(iv) the set $\mathcal{X}^I$ is connected.[3]

An agent k is *involved in an interaction* at time t if there is at least one interaction area $\mathcal{X}^I$ involving a set of agents K, such that $k \in K$ and $X_K(t) \in \mathcal{X}^I$. Otherwise, we say agent k is involved in no interaction.

Condition (i) states that each interaction area involves a subset K of the agents in a Dec-MDP. Condition (ii) ensures that in every interaction area there is at least one interaction state involving all agents in K, to minimize the number of agents involved in each interaction. Condition (iii) defines interaction areas as regions of the state space "around" an interaction state. Finally, condition (iv) states that interaction areas are sets of adjacent states in terms of the transition function. Unlike interaction states, interaction areas are not disjoint: an agent can be simultaneously involved in an interaction at two different interaction areas.

Example 1 *(Cont.).* The joint state (D,D) is an *interaction state*, corresponding to the situation in which both Robot 1 and Robot 2 are in the narrow doorway. An interaction area should include the state (D,D) and at least those states immediately adjacent to it. In fact, in order for the two robots to avoid bumping into each other in the narrow doorway, they must coordinate before actually reaching the doorway, and hence the inclusion of the neighboring states in the interaction area associated with this interaction state. In this particular scenario, this corresponds to the shaded area in Fig. 2. For illustration purposes, we consider $\mathcal{X}^I = \{6, 15, \mathsf{D}\} \times \{6, 15, \mathsf{D}\}$, although other (larger) sets are also possible.

As required in (i), $\mathcal{X}^I \subset \mathcal{X}_1 \times \mathcal{X}_2$. Also, as required in (ii), $(\mathsf{D},\mathsf{D}) \in \mathcal{X}^I$. Condition (iii) requires that the transition probabilities from states in $\mathcal{X}^I$ to states not in $\mathcal{X}^I$ can be factored as in (7). In our scenario, such factorization is not possible only in state (D,D), but it is also not possible to transition from this state to a state outside $\mathcal{X}^I$. This means that condition (iii) also holds. Finally, $\mathcal{X}^I$ is connected and condition (iv) also holds.

We exploit the fact that an agent not involved in any interaction should be unaffected by partial joint state observability, in the sense that its optimal action is the same independently of the state of the other agents. The purpose of defining the interaction areas in a Dec-MDP is precisely to single out those situations in which the actions of one agent depend on other agents. As such, when in an interaction area, the agent should use state information from the other agents in the interaction area to choose its actions. Therefore, we consider scenarios in which all agents in a particular interaction area $\mathcal{X}^I \subset \mathcal{X}_K$ at time t have full access to the state $X_K(t)$. We henceforth refer to such a Dec-MDP as having *observable interactions*.

Definition 3.4 *(Observable interactions).* A Dec-MDP has *observable interactions*, if for any interaction area $\mathcal{X}^I$ involving a set K of agents, for each $k \in K$ there is a set of local observations $\mathcal{Z}_k^I \subset \mathcal{Z}_k$ such that for every $x \in \mathcal{X}^I$

$$\mathbb{P}\big[Z_k(t) \in \mathcal{Z}_k^I \;\big|\; X_K(t) \in \mathcal{X}^I\big] = 1$$

and, for every $z_k \in \mathcal{Z}_k^I$ there is a local state $x_K \in \mathcal{X}^I$, such that

$$\mathbb{P}\big[X_K(t) = x_K \;\big|\; Z_k(t) = z_k\big] = 1.$$

[3] A set $U \subset \mathcal{X}$ is *connected* if, for any pair of states $x, y \in U$, there is a sequence of actions that, with positive probability, yields a state-trajectory $\{x(0), \ldots, x(T)\}$ such that $x(t) \in U$, $t = 0, \ldots, T$, and either $x(0) = x$ and $x(T) = y$ or vice versa.

Our focus on Dec-MDPs with observable interactions translates a property often observed in real-world scenarios: when involved in an interaction, agents are often able to observe or communicate information that is relevant for coordination. Interaction areas encapsulate the need for information sharing in a general multiagent decision problem.

Informally, we say that a Dec-MDP $\mathcal{M} = (N, (\mathcal{X}_k), (\mathcal{A}_k), (\mathcal{Z}_k), \mathsf{P}, (\mathsf{O}_k), r, \gamma)$ has *sparse interactions* if all agents are independent except in a set of M interaction areas, $\{\mathcal{X}_1^I, \ldots, \mathcal{X}_M^I\}$, with $\mathcal{X}_i^I \subset \mathcal{X}_{K_i}$ for some set of agents K_i, and such that $|\mathcal{X}_i^I| \ll |\mathcal{X}_{K_i}|$. We refer to a Dec-MDP with sparse, observable interactions as a *decentralized sparse-interaction MDP* (Dec-SIMDP).

Definition 3.5 *(Dec-SIMDP).* Let $\mathcal{M} = (N, (\mathcal{X}_k), (\mathcal{A}_k), (\mathcal{Z}_k), \mathsf{P}, (\mathsf{O}_k), r, \gamma)$ be a Dec-MDP with sparse observable interactions, encapsulated in a set of M interaction areas, $\{\mathcal{X}_1^I, \ldots, \mathcal{X}_M^I\}$, as described above. We represent such a *decentralized sparse-interaction MDP* (Dec-SIMDP) as a tuple

$$\Gamma = \big(\{\mathcal{M}_k,\ k = 1, \ldots, N\}, \big\{\big(\mathcal{X}_i^I, \mathcal{M}_i^I\big),\ i = 1, \ldots, M\big\}\big),$$

where

- each $\mathcal{M}_k$ is an MDP $\mathcal{M}_k = (\mathcal{X}_0 \times \mathcal{X}_k, \mathcal{A}_k, \mathsf{P}_k, r_k, \gamma)$ that individually models agent k in the absence of other agents, where r_k is the component of the joint reward function associated with agent k in the decomposition in (9);
- each $\mathcal{M}_i^I$ is an MMDP that captures a *local interaction* between K_i agents in the states in $\mathcal{X}_i^I$ and is given by $\mathcal{M}_i^I = (K_i, \mathcal{X}_{K_i}, (\mathcal{A}_k), \mathsf{P}_i^I, r_i^I, \gamma)$, with $\mathcal{X}_i^I \subset \mathcal{X}_{K_i}$.

Each MMDP $\mathcal{M}_i^I$ describes the interaction between a subset K_i of the N agents, and the corresponding state space $\mathcal{X}_{K_i}$ is a superset of an interaction area as defined above.

A Dec-SIMDP is an alternative way of representing a Dec-MDP with observable interactions. For all agents outside interaction areas, the joint transition probabilities and reward function for a Dec-SIMDP can be factorized as in (7) and (9), and it is possible to model these agents using individual MDPs. In the states of each interaction area in a Dec-SIMDP, and only in these, the agents involved in the interaction are able to communicate freely. In these areas, the agents can thus use communication to overcome local state perception and can be modeled using a local MMDP, observing each other's state and deciding jointly on their action. Outside these areas, the agents have only a local perception of the state and, therefore, choose the actions independently of the other agents. A simple, albeit inaccurate, way of thinking of a Dec-SIMDP is as a Dec-MDP in which each agent has access, at each time step, to all state-information required to predict its next local state and reward. In fact, a Dec-SIMDP may include states—namely in interaction areas—in which agents are able to perceive joint state information, not strictly required to predict their next local state and reward. Hence the inaccuracy of the description.

For illustration purposes, we revisit Example 1, using a Dec-SIMDP model to describe the corresponding navigation problem.

Example 1 *(Cont.).* The problem depicted in Fig. 2 can be represented as a Dec-SIMDP

$$\Gamma = \big(\{\mathcal{M}_1, \mathcal{M}_2\}, \big(\mathcal{X}^I, \mathcal{M}^I\big)\big),$$

where

- M_1 is an MDP $\mathcal{M}_1 = (\mathcal{X}_1, \mathcal{A}_1, \mathsf{P}_1, r_1, \gamma)$, describing the task of Robot 1 (navigating to Goal 1) in the absence of Robot 2. $\mathcal{X}_1$ and $\mathcal{A}_1$ are the same as in the definition of the original Dec-MDP (see page 1764), P_1 and r_1 arise from the decomposition of r and P described in (10) and (11);
- M_2 is an MDP $\mathcal{M}_2 = (\mathcal{X}_2, \mathcal{A}_2, \mathsf{P}_2, r_2, \gamma)$, describing the task of Robot 2 (navigating to Goal 2) in the absence of Robot 1;
- $\mathcal{X}^I$ is the interaction area already defined;
- $\mathcal{M}^I$ is an MMDP $\mathcal{M}^I = (2, \mathcal{X}, \mathcal{A}, \mathsf{P}, r^I, \gamma)$, that captures the interaction between Robot 1 and Robot 2 in $\mathcal{X}^I$.

In this example, there is a unique MMDP $\mathcal{M}^I$ since there is a single interaction area $\mathcal{X}^I$. Also, since there are only two robots/agents in the environment, the MMDP $\mathcal{M}^I$ is a fully observable version of the original Dec-MDP $\mathcal{M}$ with a simplified reward function. In more general scenarios, there will be an MMDP corresponding to each interaction area that includes only the agents involved in that interaction.

Finally, we explore the relation between the Dec-SIMDP model and the MDP and MMDP models. As expected, in the absence of any interaction areas, the Dec-SIMDP reduces to a set of independent MDPs that can be solved separately. A Dec-SIMDP with no interaction areas thus captures the situation in which the agents are completely independent. In those situations in which all agents interact in all states, as assumed in the general Dec-MDP model, the whole state space is an interaction area and, as such, our assumption of observable interactions renders our model equivalent to an MMDP. Nevertheless, the appeal of the Dec-SIMDP model is that many practical situations do not fall in either of the two extreme

cases *i.e.*, independent MDPs vs. fully observable MMDP. It is in these situations that the Dec-SIMDP model may bring an advantage over more general but potentially intractable models.

We first assume interaction areas to be known in advance (*i.e.*, they are provided as part of the model specification) and present a planning algorithm that leverages the particular structure of Dec-SIMDPs in Section 4. In Section 5, we then discuss how these areas can be determined and an algorithm that allows each agent to learn these interaction areas.

4. Planning in Dec-SIMDPs

We address the problem of planning in Dec-SIMDPs, *i.e.*, estimating the optimal policy for each agent in a Dec-SIMDP when the model is fully specified, including the interaction areas.

We start by introducing a general heuristic approach that relies on the solution for an associated POMDP, leading to two general algorithms, MPSI and LAPSI. We then introduce the concept of *generalized α-vectors* for Dec-SIMDPs and describe instances of both MPSI and LAPSI that use generalized α-vectors, discussing the main properties of these methods. The complete proofs of all results in this section are in Appendix A.

4.1. Heuristic planning in Dec-SIMDPs

We start by considering a Dec-SIMDP in which all except one of the agents have full state observability. We refer to this agent as agent k and further suppose that the remaining agents (those with full state observability) follow some fixed known policy, π_{-k}. Agent k can be modeled as a POMDP and the other agents can be collectively regarded as part of the environment. In this particular situation, any POMDP solution method can be used to compute the policy for agent k.

Our heuristic departs from this simplified setting and computes a policy for each agent k as if all other agents had full observability and followed some fixed known policy π_{-k}. This hypothesized policy π_{-k} allows each agent k to approximately track the other agents and choose its actions accordingly. The closer π_{-k} is to the actual policy of the other agents, the better agent k is able to track them and select its individual actions. In fact, even if the ability of an agent to track the other agents necessarily depends on the stochasticity of the environment,[4] in our scenarios the determinant factor in the agents' ability to maintain a reliable (albeit flat) belief on the state of the other agents depends on how accurate the transition model for the other agents is, which critically depends on how close π_{-k} is to their actual policy.

Algorithm 1 summarize this general algorithm:

Algorithm 1 Heuristic planning algorithm for Dec-SIMDPs

Require: Dec-SIMDP model Γ;
1: **for all** agents $k = 1, \ldots, N$ **do**
2: Build hypothetical policy $\hat{\pi}_{-k}$;
3: Using Γ and $\hat{\pi}_{-k}$, build POMDP model for agent k;
4: Use preferred POMDP solution technique to compute π_k;
5: **end for**

The idea behind Algorithm 1 can be used in general Dec-POMDPs. However, as the hypothesized policy π_{-k} seldom corresponds to the actual policy followed by the other agents, this method does not allow each agent k to properly track the other agents and decide accordingly, even if the environment is close to deterministic. Hence this approach is inadequate for general Dec-POMDPs, where we expect it to lead to poor results. The particular structure of Dec-SIMDPs, however, renders this approach more appealing because (i) outside of interaction areas, the individual policy of agent k ideally exhibits little dependence on the state/policy of the other agents. As such, poor tracking in these areas has little impact on the policy of agent k; and (ii) inside interaction areas, local full observability allows agent k to perfectly track the other agents involved in the interaction and choose its actions accordingly.

We present two algorithms, Myopic Planning for Sparse Interactions (MPSI) and Look-Ahead Planning for Sparse Interactions (LAPSI), that are particular instances of Algorithm 1 but consider different hypothetical policies for the other agents. In MPSI, agent k considers each of the other agents as completely self-centered and oblivious to the interactions. Agent k thus acts as if each agent j, $j \neq k$, acts according to a policy π_j—the optimal policy for the corresponding MDP $\mathcal{M}_j$ in the Dec-SIMDP. In environments with almost no interaction, the MPSI heuristic provides a good approximation to the policy of the other agents outside the interaction areas.

In LAPSI, agent k considers that all other agents jointly adopt the optimal policy for the underlying MMDP.[5] LAPSI is the counterpart to MPSI, as it provides a good approximation to the policy of the other agents in scenarios where the interactions are not so sparse.

[4] For example, in POMDPs with very stochastic transitions (meaning that transitions out of one state may lead to many states), the beliefs used for decision-making are often flat, where many states have non-negligible probability.

[5] The MMDP associated with the Dec-SIMDP is the MMDP obtained by endowing all agents with global full-state observability.

Using the corresponding hypothesized policies for the remaining agents, MPSI and LAPSI can now leverage any POMDP solution methods to obtain a policy for each agent k. We now introduce the concept of *generalized α-vectors*, that we use to construct particular instances of both MPSI and LAPSI.

4.2. Generalized α-vectors for Dec-SIMDPs

MPSI and LAPSI are described in terms of general POMDP solvers and can be used to compute an individual policy for each agent in a Dec-SIMDP. We now propose particular instances of both MPSI and LAPSI that exploit the structure of the Dec-SIMDP model. We depart from the single-agent POMDP model constructed using the proposed approach (see Algorithm 1) and introduce *generalized α-vectors*, a simplification of the α-vectors used in POMDP solvers [29]. Using the generalized α-vectors, we derive a variation of the Q_{MDP} heuristic [23] and provide bounds for the performance of this method when applied to Dec-SIMDPs. We focus on 2-agent scenarios, to avoid unnecessarily complicating the presentation, remarking that the development presented extends easily to more than two agents only at the cost of more cumbersome expressions.

Using the POMDP model obtained by adopting the introduced heuristic, agent k computes an individual policy π_k that maps *beliefs* to actions. However, agent k has full local state observability, implying that the kth component of the state is unambiguously determined. Furthermore, given our assumption of observable interactions (Definition 3.4), at each time step only those state components corresponding to agents not interacting with agent k are unobservable. By definition, these state-components do not depend on the state-action of agent k and depend only on π_{-k}.

Recovering the POMDP model for agent k (Section 2),

$$Q^*(\mathbf{b}, a) = \sum_x \mathbf{b}_x \left[r(x,a) + \gamma \sum_{z,y} \mathsf{P}(x,a,y)\mathsf{O}(y,a,z) \max_u Q^*\big(\mathsf{Bel}(\mathbf{b},a,z), u\big) \right].$$

Since agent k has full local observability, the belief $\mathbf{b}$ concerns only the state component of the other agent,

$$Q^*(\bar{x}_k, \mathbf{b}_{-k}, a) = \sum_{x_{-k}} \mathbf{b}_{x_{-k}} \left[r(x,a) + \gamma \sum_{z,y} \mathsf{P}(x,a,y)\mathsf{O}(y,a,z) \max_u Q^*\big(\bar{y}_k, \mathsf{Bel}(\mathbf{b},a,z), u\big) \right].$$

The other agent is assumed to follow a fixed policy that depends only on the current state, so we can eliminate the explicit dependence on its action,

$$Q^*(\bar{x}_k, \mathbf{b}_{-k}, a_k) = \sum_{x_{-k}} \mathbf{b}_{x_{-k}} \left[r_{\pi_{-k}}(x,a_k) + \gamma \sum_{z,y} \mathsf{P}_{\pi_{-k}}(x,a_k,y)\mathsf{O}(y,a,z) \max_{u_k} Q^*\big(\bar{y}_k, \mathsf{Bel}(\mathbf{b},a_k,z), u_k\big) \right],$$

where

$$r_{\pi_{-k}}(x,a_k) = \sum_{a_{-k}} \pi_{-k}(x,a_{-k}) r\big(x,(a_{-k},a_k)\big),$$

$$\mathsf{P}_{\pi_{-k}}(x,a_k,y) = \sum_{a_{-k}} \pi_{-k}(x,a_{-k}) \mathsf{P}\big(x,(a_{-k},a_k),y\big).$$

Every time step t that agent k is in an interaction area, implying that so is the other agent, agent k can unambiguously perceive their joint state and hence $Z_k(t) = X(t)$. In all remaining time steps, $Z_k(t) = \bar{X}_k(t)$, meaning that the agent observes only its local state. Denoting by $\mathcal{X}_I$ the set of all joint states in any interaction area, *i.e.*, $\mathcal{X}_I = \bigcup_{i=1}^M \mathcal{X}_i^I$,

$$\begin{aligned} Q^*(\bar{x}_k, \mathbf{b}_{-k}, a_k) = \sum_{x_{-k}} \mathbf{b}_{x_{-k}} \Bigg[& r_{\pi_{-k}}(x,a_k) + \gamma \sum_{y \in \mathcal{X}_I} \mathsf{P}_{\pi_{-k}}(x,a_k,y) \max_{u_k} Q^*(\bar{y}_k, y_{-k}, u_k) \\ & + \gamma \sum_{y \notin \mathcal{X}_I} \mathsf{P}_{\pi_{-k}}(x,a_k,y) \max_{u_k} Q^*\big(\bar{y}_k, \mathsf{Bel}(\mathbf{b},a_k,\bar{y}_k), u_k\big) \Bigg]. \end{aligned} \tag{13}$$

We now focus explicitly on the updated belief $\mathsf{Bel}(\cdot)$ and the optimal Q-function for the states in the interaction areas. The general belief-update expression is (Section 2.2)

$$\mathsf{Bel}_y\big(\mathbf{b}, a_k, Z_k(t+1)\big) = \eta \sum_x \mathbf{b}_x(t) \mathsf{P}\big(x, A(t), y\big) \mathsf{O}\big(y, A(t), Z_k(t+1)\big).$$

In the current setting, the belief concerns only the distribution over states of the other agent. As such, if $X_k(t) = x_k$, then

$$\mathsf{Bel}_{y_{-k}}\big(\mathbf{b}, a_k, Z_k(t+1)\big) = \eta \sum_{x_{-k}} \mathbf{b}_{x_{-k}}(t) \mathsf{P}_{\pi_{-k}}\big(x, A_k(t), y_{-k}\big) \mathsf{O}\big(y_{-k}, A_k(t), Z_k(t+1)\big),$$

where $x = (x_{-k}, \bar{x}_k)$,

$$\mathsf{P}_{\pi_{-k}}(x, a_k, y_{-k}) = \sum_{a_{-k}} \pi_{-k}(x, a_{-k}) \mathsf{P}_{-k}\big(x_{-k}, (a_{-k}, a), y_{-k}\big)$$

and, as in Section 2.4, P_{-k} denotes the transition probabilities corresponding to all except the kth components of the state. If the agents are not in an interaction area, the transitions of the other agent do not depend on the actions of agent k and hence

$$\mathsf{Bel}_{y_{-k}}\big(\mathbf{b}, a_k, Z_k(t+1)\big) = \eta \sum_{x_{-k}} \mathbf{b}_{x_{-k}}(t) \mathsf{P}_{\pi_{-k}}(x, y_{-k}) \mathsf{O}\big(y_{-k}, A_k(t), Z_k(t+1)\big).$$

If $X(t+1)$ is in an interaction area, then agent k can observe the state of the other agent and, as such,

$$\mathsf{Bel}_{y_{-k}}(\mathbf{b}, a_k, y) = \mathbb{I}_{X_{-k}(t+1)}(y_{-k}),$$

where $\mathbb{I}_x$ denotes the indicator function for the singleton set $\{x\}$. For the general situation in which $X(t+1) \notin \mathcal{X}_I$,

$$\mathsf{Bel}_{y_{-k}}(\mathbf{b}, a_k, \bar{y}_k) = \eta \sum_{x_{-k}} \mathbf{b}_{x_{-k}}(t) \mathsf{P}_{\pi_{-k}}\big(x, A_k(t), y_{-k}\big) \mathbb{I}_{\mathcal{X}_I^c}(y), \tag{14}$$

where, for a general set $U \subset \mathcal{X}$, $\mathbb{I}_U$ is the indicator function for the set U and U^c denotes the complement of U in $\mathcal{X}$.

We now consider the expression for $Q^*(\bar{x}_k, \mathbf{b}_{-k}, a_k)$ when the agents are in an interaction area. In this case, $\mathbf{b}_{-k} = \mathbf{e}_{x_{-k}}$ for some x_{-k}, where $\mathbf{e}_x$ is the probability vector where each component y is given by $\mathbb{I}_x(y)$. This implies that

$$\begin{aligned} Q^*(\bar{x}_k, x_{-k}, a_k) = r_{\pi_{-k}}(x, a_k) &+ \gamma \sum_{y \in \mathcal{X}_I} \mathsf{P}_{\pi_{-k}}(x, a_k, y) \max_{u_k} Q^*(\bar{y}_k, y_{-k}, u_k) \\ &+ \gamma \sum_{y \notin \mathcal{X}_I} \mathsf{P}_{\pi_{-k}}(x, a_k, y) \max_{u_k} Q^*\big(\bar{y}_k, \mathsf{Bel}(b, a_k, \bar{y}_k), u_k\big). \end{aligned} \tag{15}$$

Noting the similarity between the right-hand side of (15) and the term in square brackets in the right-hand side of (13), we define a *generalized α-vector* for agent k, $\boldsymbol{\alpha}_k$, recursively as follows:

$$\begin{aligned} \boldsymbol{\alpha}_k(x, a_k) = r_{\pi_{-k}}(x, a_k) &+ \gamma \sum_{y} \mathsf{P}_{\pi_{-k}}(x, a_k, y) \max_{u_k} \boldsymbol{\alpha}_k(y, u_k) \mathbb{I}_{\mathcal{X}_I}(y) \\ &+ \gamma \sum_{y_k} \mathsf{P}(\bar{x}_k, a_k, \bar{y}_k) \max_{u_k} \sum_{y_{-k}} \mathsf{P}_{\pi_{-k}}(x_{-k}, y_{-k}) \boldsymbol{\alpha}_k(y, u_k) \mathbb{I}_{\mathcal{X}_I^c}(y). \end{aligned} \tag{16}$$

Theorem 4.1. *Given a two-agent Dec-SIMDP Γ, the generalized α-vectors associated with agent k when the other agent follows a* fixed and known policy *π_{-k} are well defined, i.e., they always exist and are unique.*

Proof. (sketch) The complete proof of the theorem is in Appendix A. We introduce a dynamic-programming operator $\mathbf{T}_k$ such that

$$\boldsymbol{\alpha}_k = \mathbf{T}_k \boldsymbol{\alpha}_k$$

and show this operator to be a contraction in the sup-norm. The statement in the theorem follows from Banach's fixed-point theorem. □

The operator $\mathbf{T}_k$ is used to iteratively compute the generalized α-vectors, in a way very similar to the value-iteration algorithm used to compute the optimal Q-function for an MDP. The next result establishes that the generalized α-vectors associated with a Dec-SIMDP can be computed efficiently.

Theorem 4.2. *The generalized α-vectors for a* 2*-agent Dec-SIMDP Γ verifying the conditions of Theorem* 4.1 *can be computed in polynomial time.*

Proof. (sketch) The result follows from noting that the generalized α-vectors can be computed by solving an associated MDP. □

Given that an MDP is a particular case of a Dec-SIMDP in which there is a single agent, for such a Dec-SIMDP the generalized α-vectors correspond exactly to the optimal Q-values. It then follows from Theorem 4.2 that computing the generalized α-vectors for a Dec-SIMDP is P-complete.

All results extend to scenarios with more than two agents. For example, the definition in (16) takes the more general form

$$\boldsymbol{\alpha}_k(x,a_k) = r_{\pi_{-k}}(x,a_k) + \gamma \sum_{y_O} \mathsf{P}_{\pi_{-k}}(x_O, a_k, y_O) \max_{u_k} \sum_{y_{-O}} \mathsf{P}_{\pi_{-k}}(x_{-O}, y_{-O}) \boldsymbol{\alpha}_k(y,u_k),$$

where $y = (y_O, y_{-O})$, for $y \in \mathcal{X}$. The components y_O correspond to the observable components of y—those that belong to agents involved in an interaction with agent k—and y_{-O} correspond to the remaining components.

4.3. Generalized α-vectors in LAPSI and MPSI

We now use the generalized α-vectors to compute estimates $\hat{Q}(\mathbf{b}, a_k)$ of the optimal Q-function for agent k,

$$\hat{Q}(\bar{x}_k, \mathbf{b}_{-k}, a_k) = \sum_{x_{-k}} \mathbf{b}_{x_{-k}} \boldsymbol{\alpha}_k(x, a_k). \tag{17}$$

This approximation shares several features with the Q_{MDP} heuristic (Section 2.2), since we compute the generalized α-vectors for a Dec-SIMDP by solving an associated MDP. One would thus expect our methods to suffer in states of great uncertainty, much like Q_{MDP} does. In these states, action selection may be conservative but our assumed sparse interactions will hopefully minimize the effects of this situation. Our experimental results indicate that this may indeed be the case.

We now derive error bounds for the approximation in (17) that depend only on the *dispersion* of the maximum values of the generalized α-vectors outside the interaction areas. This result can be extended to general POMDPs, providing error bounds for the Q_{MDP} heuristic that depend only on the optimal Q-function for the underlying MDP.

Given a generalized α-vector $\boldsymbol{\alpha}_k(x,a_k)$, let x_O denote the *observable* components of x—those corresponding to agents interacting with k at $\bar{x}_k$—and x_{-O} the remaining components. We define the *dispersion* of a set of α-vectors $\boldsymbol{\alpha}_k(x,a_k)$, with $x \in \mathcal{X}$ and $a_k \in \mathcal{A}_k$, as

$$\boldsymbol{\Delta}_k = \max_{x_O} \left| \max_{u_k} \sum_{x_{-O}} \boldsymbol{\alpha}_k\big((x_O, x_{-O}), u_k\big) - \sum_{x_{-O}} \max_{u_k} \boldsymbol{\alpha}_k\big((x_O, x_{-O}), u_k\big) \right|. \tag{18}$$

The dispersion of a set of α-vectors measures how the maximum value of $\boldsymbol{\alpha}_k$ taken over all actions differs from the corresponding average in the non-observable components of the state. The dispersion quantifies how the lack of knowledge of agent k on the state of the other agents can impact the action choice of agent k in terms of value.

Theorem 4.3. *Let $\mathcal{M} = (\{\mathcal{M}_k,\ k = 1,\ldots,N\}, \{(\mathcal{X}_i^I, \mathcal{M}_i^I),\ i = 1,\ldots,M\})$ be a Dec-SIMDP and let π_k denote the policy for agent k obtained from the approximation* (17) *when the policy π_{-k} for the other agents is fixed and known. Then,*

$$\left\| V^* - V^{\pi_k} \right\|_\infty \leqslant \frac{2\gamma^2}{(1-\gamma)^2} \boldsymbol{\Delta}_k, \tag{19}$$

where $\boldsymbol{\Delta}_k$ represents the dispersion of the α-vectors associated with π_{-k}.

Proof. See Appendix A. □

Theorem 4.3 translates well-known bounds for approximate policies to our particular setting. As expected, the bound in (19) is proportional to the total dispersion of the generalized α-vectors. Also, the bound in (19) is zero if either

- The whole state space is an interaction area, *i.e.*, $\mathcal{X}_I = \mathcal{X}$. In this case, we recover the MMDP version of the problem (Section 3).
- There are no interaction states, *i.e.*, $\mathcal{X}_I = \emptyset$. In this case, the generalized α-vectors for agent k do not depend on the other agents, implying that $\boldsymbol{\Delta}_k = 0$.

Finally, the above bounds assume that the policy hypothesized for the other agents corresponds to their actual policy, which is not the case in either MPSI and LAPSI. In particular, defining the errors in $r_{\pi_{-k}}$ and $\mathsf{P}_{\pi_{-k}}$, arising from the imperfect knowledge of π_{-k}, as

$$\varepsilon_r = \|r_{\pi_{-k}} - r_{\hat{\pi}_{-k}}\|_\infty, \qquad \varepsilon_{\mathsf{P}} = \|\mathsf{P}_{\pi_{-k}} - \mathsf{P}_{\hat{\pi}_{-k}}\|_\infty,$$

we finally get

$$\left\| V^* - V^{\pi_k} \right\|_\infty \leqslant \frac{2\gamma^2}{(1-\gamma)^2} \boldsymbol{\Delta}_k + \frac{\varepsilon_r + \gamma\varepsilon_{\mathsf{P}}}{1-\gamma}, \tag{20}$$

where the first term corresponds to the approximation error of the POMDP policy, and the second term corresponds to the approximation error of the policy of the other agents.

The error bounds in (20) are generally loose—as known similar bounds for MDPs are also loose. However, in our view, the most interesting aspect of the proposed bounds arises from the definition and identification of the *dispersion of generalized α-vectors*, on which those bounds critically depend.

Using the generalized α-vectors in LAPSI and MPSI is now straightforward. Both methods use the estimate in (17) to choose the action for agent k. The difference between the two methods lies on the policy π_{-k} hypothesized for the other agents, needed to both track the belief $\mathbf{b}_{-k}$ and to compute the α-vectors. In MPSI, π_{-k} is taken as the reduced policy obtained from the individual policies π_j, $j \neq k$, the optimal policy for the corresponding MDP $\mathcal{M}_j$ in the Dec-SIMDP. In LAPSI, π_{-k} is obtained from the optimal policy for the underlying MMDP by ignoring component k.

For illustration purposes, we apply both of these methods to several problems of different dimension, using (17) as our estimate for the POMDP optimal Q-function. As discussed above, the estimate in (17) corresponds to a variation of the Q_{MDP} heuristic. Our results indicate that, even using such a sub-optimal POMDP solver, LAPSI is able to attain a performance close to optimal in all test scenarios while incurring a computational cost similar to that of solving the underlying MMDP. MPSI, while computationally more efficient, seems to lead to agents that excessively avoid interactions. We also compare our algorithms with a previous algorithm for Dec-SIMDPs, the IDMG algorithm [30]. Our results indicate that LAPSI is able to attain similar performance to that of IDMG while providing significant computational savings. We also show that the IDMG algorithm is, by design, unable to consider future interactions when planning outside the interaction areas, which leads to poor performance. Such limitation is not present in LAPSI.

4.4. Results

To gain a better understanding of the applicability and general properties of our methods, we compared the performance of both MPSI and LAPSI to those of individual agents that plan disregarding other agents in the environment and of the optimal fully observable MMDP policy. We also tested the performance of the IDMG algorithm, previously introduced as an heuristic planning algorithm for Dec-SIMDPs [30]. It was empirically shown that this algorithm attains near-optimal performance on several scenarios, and we use the results obtained with this approach as a benchmark against which we compare the performance of LAPSI and MPSI. We briefly describe the IDMG algorithm and outline the main differences between this approach and those presented in this article.

4.4.1. The IDMG algorithm

Let $\Gamma = (\{\mathcal{M}_k,\ k = 1, \ldots, N\}, \{(\mathcal{X}_i^I, \mathcal{M}_i^I),\ i = 1, \ldots, M\})$ denote an N agent Dec-SIMDP. Let Q_k^* denote the optimal Q-function associated with each of the MDPs $\mathcal{M}_k$ in Γ. Similarly, let Q_i^I denote the joint optimal Q-function associated with the MMDP $\mathcal{M}_i^I$ in Γ. IDMG proposes to use these two sets of functions, $\{Q_k^*,\ k = 1, \ldots, N\}$ and $\{Q_i^I,\ i = 1, \ldots, M\}$, as follows:

- At every time step t for which an agent k is involved in no interaction, it follows the greedy policy with respect to Q_k^*, *i.e.*,

$$\pi_k(x_k) \in \arg\max_{a_k \in \mathcal{A}_k} Q_k^*(x_k, a_k).$$

- Otherwise, let O denote the set of all agents interacting with k at time step t. For each $k' \in O$, and all $x \in \mathcal{X}$, let

$$\bar{Q}_{k'}(x, a) = Q_{k'}^*(\bar{x}_{k'}, a_{k'}) + \sum_{i=1}^{M} Q_i^I(x_{K_i}, a_{K_i}) \mathbb{I}_{\mathcal{X}_{K_i}}(x_{K_i}).$$

  For each $x \in \mathcal{X}$, $\bar{Q}_{k'}(x, \cdot)$ combines $Q_{k'}^*(x, \cdot)$ with the joint Q-functions associated with all interactions of agent k' in state x.

  Since these functions are different for each agent $k' \in O$, at each time-step t IDMG adopts a game-theoretic approach and constructs a *matrix game*

$$\Gamma_{X(t)} = \big(|O|, (\mathcal{A}_{k'}), (q_{k'})\big),$$

  involving all agents in O. Each payoff function $q_{k'}$ is given by

$$q_{k'}(a_O) = r_{k'}(\bar{x}_{k'}, a_{k'}) + \sum_{i=1}^{M} r_i^I(x_{K_i}, a_{K_i}) \mathbb{I}_{\mathcal{X}_{K_i}}(x_{K_i}) + \gamma \sum_{y_O \in \mathcal{X}_O} \mathsf{P}(x_O, a_O, y_O) \text{Nash}_{k'}\big(\bar{Q}_{k'}(y_O, \cdot),\ k' \in O\big),$$

  where $X(t) = x$ and $\text{Nash}_{k'}$ is a max-like operator denoting the Nash value for agent k' with respect to the game defined by the matrices $\bar{Q}_{k'}$ in state y_O. Finally, at every time step t for which agent k is involved in some interaction, it chooses its action according to the Nash equilibrium for the game $\Gamma_{X(t)}$.

Table 1
Distinction between different methods used for comparison in terms of planning and prescribed policies.

Policy	Planning	In int. areas	Outside int. areas
IDMG	Equilibrium	Equilibrium	MDP
MPSI	MDP (hyp.)	POMDP	POMDP
LAPSI	MMDP (hyp.)	POMDP	POMDP
Indiv.	–	MDP	MDP
Opt.	MMDP	MMDP	MMDP

There are two important distinctions between the IDMG algorithm and LAPSI/MPSI. Outside interaction areas, IDMG is oblivious to the existence of other agents in the environment, *i.e.*, it never considers the effect that future interactions may have in its performance. This, as will soon be apparent, leads to situations in which IDMG may perform arbitrarily poor.

A second significant difference concerns the use of a game-theoretic solution to resolve the interactions that requires the computation of multiple equilibria. Computing equilibria is known to be a computationally demanding problem, which lead to a very significant computational advantage to LAPSI/MPSI, as will also become apparent from the results.

Table 1 summarizes the main distinctions between the different methods used for comparison. For each method, the second column indicates which policy is hypothesized for the other agents during planning. The third column corresponds to the policy prescribed by that method in interaction areas, and the last column corresponds to the policy prescribed outside interaction areas. During planning, IDMG assumes the other agents adopt an equilibrium solution in interaction areas and prescribes this same policy in those areas. Outside interaction areas, IDMG follows an individual MDP policy that disregards the other agents. During planning, LAPSI and MPSI assume the other agents to follow the joint MMDP and individual MDP policies, respectively. Both methods then construct a POMDP for planning, whose policy is then followed inside and outside interaction areas. Individual agents do not consider other agents during planning and always follow an individual MDP policy. The optimal agents assume that the other agents follow the joint MMDP policy during planning and then adopt that same policy.

4.4.2. Experimental setup

Fig. 3 depicts the different scenarios used to test our algorithm. The reason for using navigation scenarios is that the Dec-SIMDP model appears particularly appealing for modeling multi-robot problems. Furthermore, in this class of problems, the results can easily be visualized and interpreted. In each of the test scenarios, each robot in a set of two/four robots must reach one specific state. In the smaller environments (Fig. 3a through 3d), the goal state is marked with a boxed number, corresponding to the number of the robot. The cells with a simple number correspond to the initial states for the robots. In the larger environments (Fig. 3e through 3j), the goal for each robot is marked with a cross and the robots each depart from the other robot's goal state, in an attempt to increase the possibility of interaction.

Each robot has 4 possible actions that move the robot in the corresponding direction with probability 0.8 and fail with probability 0.2, leaving the state of the robot unchanged. The shaded regions correspond to interaction areas, inside of which the darker cells correspond to interaction states. When two or more robots stand in one of these cells simultaneously, they both get a penalty of -20. Upon reaching the corresponding goal, each agent receives a reward of $+1$. In all experiments we used $\gamma = 0.95$.

Table 2 summarizes the dimension of the joint state space for the corresponding Dec-MDP. For comparison purposes, Table 2 also includes the number of interaction states and the number of states in the individual MDPs that give an idea of the number of states in the corresponding Dec-SIMDP model.

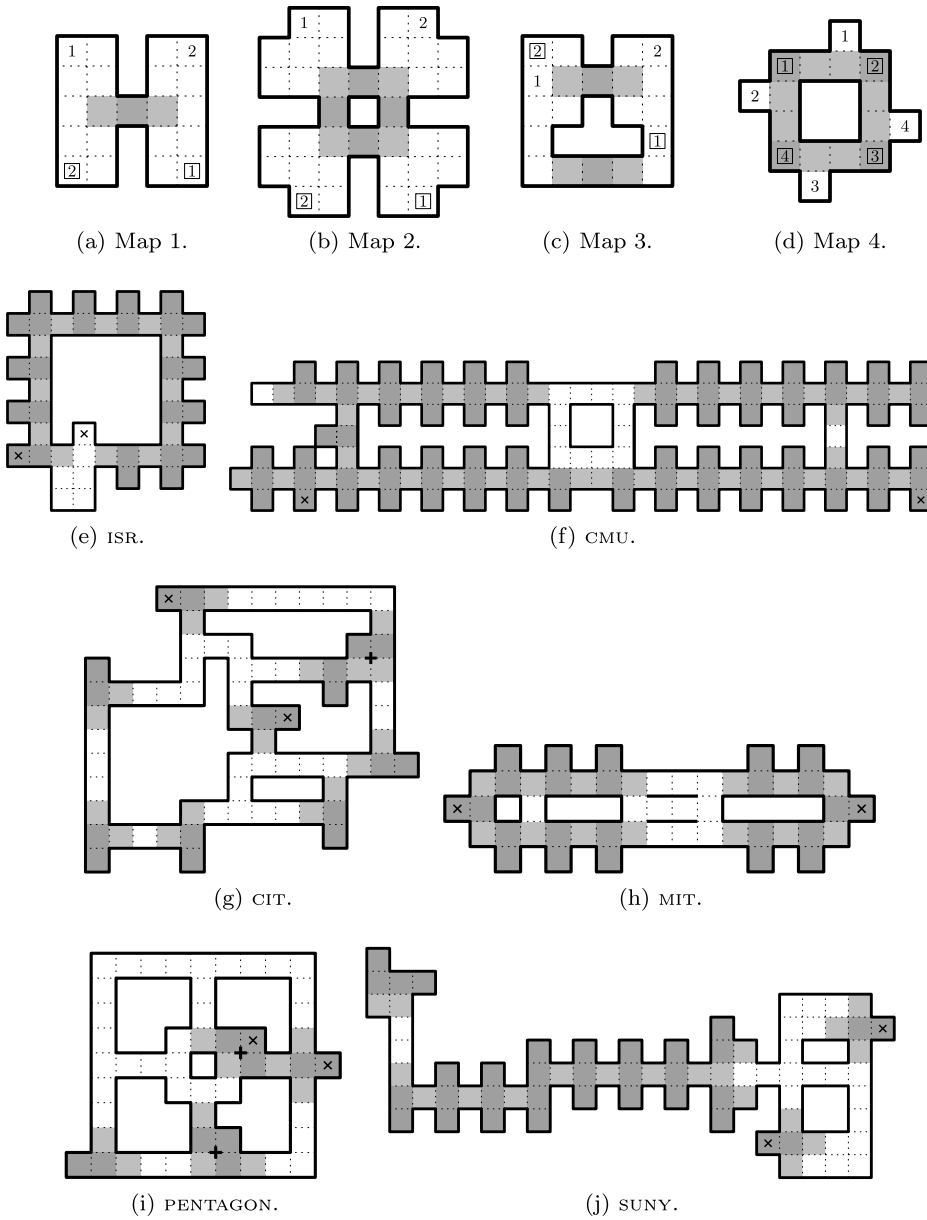
4.4.3. Description and discussion of results

For each of the different scenarios in Fig. 3, we run the algorithms in Table 1 and test the obtained policies for 1000 independent Monte Carlo trials. Each trial lasts for 250 time steps.

Except for the individual agents, all methods are able to completely avoid miscoordinations in all tested scenarios. Table 3 summarizes the average number of miscoordinations for the individual agents. The results in Table 3 provide a hint on the coordination needs for the different environments. For example, environments such as CIT or SUNY require no coordination at all, while Map 4 (Fig. 3d) requires significant coordination. Tables 4 and 5 present comparative results in terms of total discounted reward and steps-to-goal. Performance results for the IDMG algorithm in Map 4 (Fig. 3d) are not available, since the complexity of this particular instance prevented the method from providing solutions in practical computational time.

The LAPSI algorithm performs very close to the optimal MMDP policy in all environments, in spite of the significant difference in terms of state information available to both methods. Also, in most scenarios, LAPSI and IDMG perform similarly, both in terms of total discounted reward and in terms of steps-to-goal. The only exceptions are Map 2, where LAPSI outperforms IDMG, and ISR, where IDMG outperforms LAPSI. Interestingly, however, the difference in terms of time-to-goal in the ISR environment is not significant. Our results agree with previous ones that showed that IDMG attains close-to-optimal performance in most scenarios considered here [30].

Interestingly, in the ISR scenario alone, the individual agents are actually able to outperform the Dec-SIMDP planning methods. Observing the results in terms of steps-to-goal, we conclude that in this particular scenario the Dec-SIMDP planning methods take longer to reach the goal, an indication that they may be attempting to avoid miscoordinations at a cost of



(a) Map 1. (b) Map 2. (c) Map 3. (d) Map 4.

(e) ISR. (f) CMU.

(g) CIT. (h) MIT.

(i) PENTAGON. (j) SUNY.

Fig. 3. Environments used in the experiments.

Table 2
Dimension of the different test scenarios. The number of joint states in a scenario involving N agents in an environment described by a map with M cells is given by M^N.

Environment	# Joint states	# States in inter. areas	# MDP states
Map 1	441	9	21 (×2)
Map 2	1296	36	36 (×2)
Map 3	400	18	20 (×2)
Map 4	65,536	1188	16 (×4)
CIT	4900	153	70 (×2)
CMU	17,689	714	133 (×2)
ISR	1849	203	43 (×2)
MIT	2401	192	49 (×2)
PENTAGON	2704	113	52 (×2)
SUNY	5476	287	74 (×2)

Table 3
Number of miscoordinations for the individual agents in each of the test scenarios. The results are averaged over 1000 independent Monte Carlo trials. For each environment, bold entries correspond to optimal values (differences from optimal are not statistically significant).

Environment	Indiv.
Map 1	0.409
Map 2	0.239
Map 3	0.280
Map 4	1.943
CIT	**0.000**
CMU	0.249
ISR	**0.004**
MIT	**0.010**
PENTAGON	0.120
SUNY	**0.000**

Table 4
Total discounted reward for each of the four different algorithms in each of the test scenarios. The results are averaged over 1000 independent Monte Carlo trials. For each environment, bold entries correspond to optimal values (differences are not statistically significant). Italic entries correspond to values whose differences are not statistically significant.

Environment	IDMG	MPSI	LAPSI	Indiv.	Opt.
Map 1	**12.035**	11.130	**11.992**	5.919	**12.059**
Map 2	10.672	10.159	10.947	7.423	**11.108**
Map 3	*13.722*	*13.249*	*13.701*	9.378	**13.837**
Map 4	–	15.384	15.564	−4.741	**16.447**
CIT	**11.178**	**11.105**	**11.126**	**11.120**	**11.128**
CMU	**2.839**	2.688	**2.824**	0.982	**2.840**
ISR	14.168	*13.937*	*13.997*	14.301	**14.407**
MIT	**6.663**	**6.641**	**6.648**	**6.628**	**6.705**
PENTAGON	**16.031**	15.162	**15.976**	14.167	**16.016**
SUNY	**11.161**	**11.130**	**11.139**	**11.144**	**11.149**

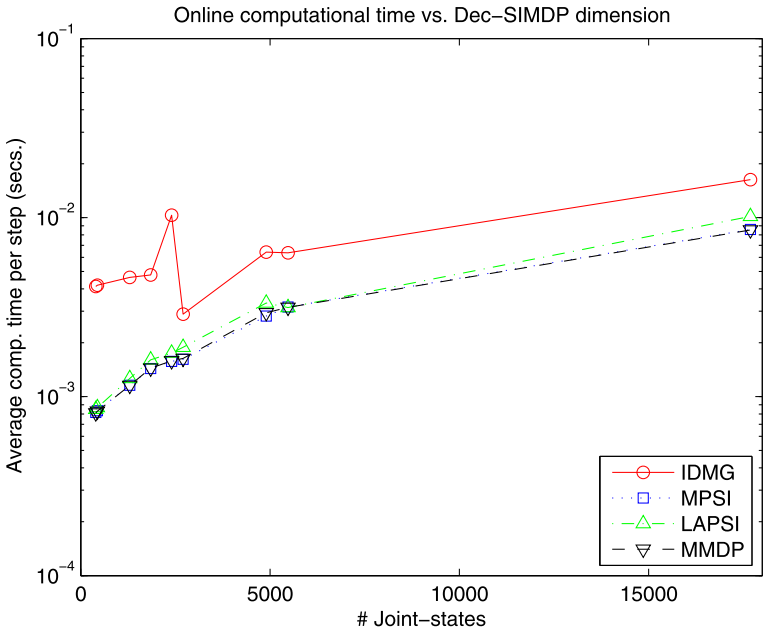
Table 5
Steps-to-goal for each of the four different algorithms in each of the test scenarios. The results are averaged over 1000 independent Monte Carlo trials. For each environment, bold entries correspond to optimal values (differences are not statistically significant). Italic entries correspond to values whose differences are not statistically significant.

Environment	IDMG	MPSI	LAPSI	Indiv.	Opt.
Map 1	*11.021*	12.752	*11.091*	**9.964**	*10.977*
Map 2	13.368	14.433	12.828	**12.494**	**12.546**
Map 3	*8.450*	9.282	*8.477*	**7.529**	8.267
Map 4	–	*6.088*	*6.071*	6.292	**5.001**
CIT	**12.422**	**12.552**	**12.514**	**12.526**	**12.512**
CMU	*39.338*	49.341	*39.444*	**38.880**	*39.340*
ISR	*7.993*	8.986	*8.012*	**7.500**	**7.440**
MIT	**22.578**	24.507	**22.618**	**22.329**	**22.444**
PENTAGON	*5.348*	19.684	*5.416*	**5.008**	*5.365*
SUNY	**12.448**	**12.500**	**12.487**	**12.477**	**12.469**

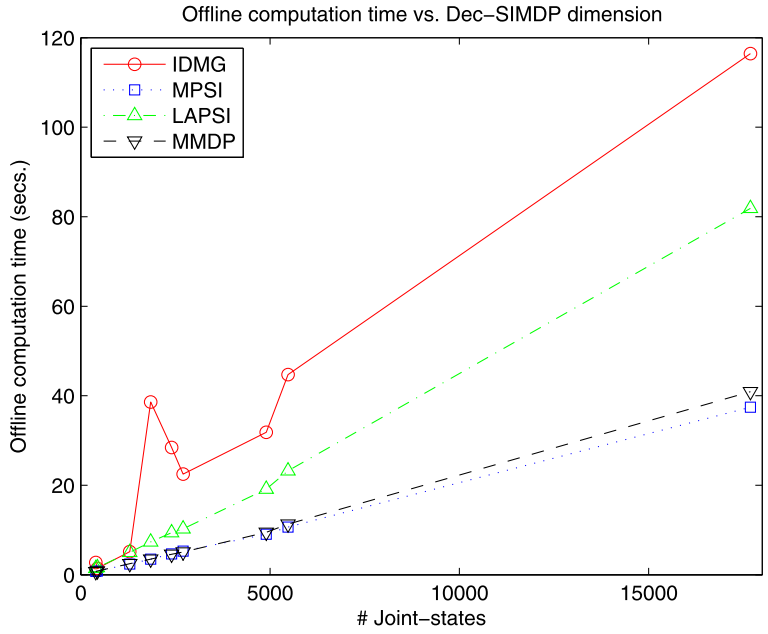
reaching the goal later, this slightly impacting their performance. However, in most other scenarios, the performance of the individual agents is significantly worse than that of all other methods. The difference is more noticeable in those scenarios where coordination is more critical.

Another interesting observation is that MPSI typically performs worse than the other methods. Since an agent in MPSI considers the other agents to disregard the consequences of miscoordinations (each is focused only on its individual goal), it is expected that the agent following MPSI is more "cautious" and hence the observed longer time to the goal. To see why this is so, consider for example the environment in Map 1, in which the two agents cannot simultaneously cross the narrow passage. If the agents find each other in opposite sides of the passage, one of them must make way to the other. In MPSI, each agent k assumes that the other agent will not deviate and, as such, agent k will cautiously deviate. This means that, in this case, both agents will deviate. Both agents will eventually move across the narrow passage, but the cautious decision-process causes delays and impacts the total reward received, as seen in Table 5. In LAPSI the agents are implicitly following coordinated policies, so the delays observed in MPSI do not occur.

In our results, the difference in performance between both LAPSI and IDMG and the optimal MMDP policy occurs both in terms of total discounted reward and in terms of steps-to-goal. Given the discount factor γ, the latter in part explains



(a) Online



(b) Offline

Fig. 4. Computation time for the different algorithms as a function of the problem dimension, corresponding to the number of joint states reported in the second column of Table 2.

the former: if the agents take longer to reach their goal, the corresponding reward is further discounted. The above results thus indicate that both our algorithms and the IDMG algorithm require more time to reach the goal configuration than that needed by MMDP solution, and this time is spent in avoiding the penalties. Also, the choice of interaction areas greatly influences the ability of the algorithms to avoid penalties without incurring any delays in reaching the goal.[6]

Since IDMG requires the computation of several equilibria both in the off-line planning phase and in the on-line running phase, the computational complexity of the IDMG algorithm may quickly become prohibitive, in scenarios with large action

[6] This was already reported in [30] concerning the IDMG algorithm.

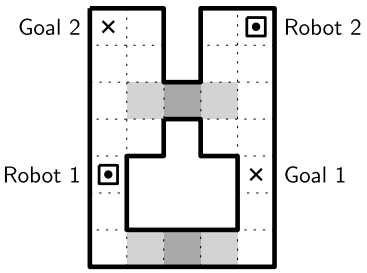


Fig. 5. Example scenario where avoiding the interaction may be beneficial.

Table 6
Estimated error bounds for the different environments.

Environment	MPSI	LAPSI
Map 1	7.248	3.683
Map 2	9.826	5.717
Map 3	13.019	6.436
CIT	38.153	20.792
CMU	98.059	44.586
ISR	78.229	44.406
MIT	54.701	32.704
PENTAGON	21.586	21.369
SUNY	54.856	28.723

spaces and/or with many interaction areas. We compare the computational effort of our methods with that of IDMG, both in terms of the average off-line computation time and the on-line computation time. Fig. 4 summarizes these results. Both MPSI and LAPSI are significantly more computationally efficient than the IDMG algorithm according to either of the two performance metrics. It is also interesting to note how the average computation times evolve with the dimension of the problem: while the computation time for both MPSI and LAPSI grows linearly with the dimension of the problem, the computation time of IDMG also depends on the number of interaction areas and hence the irregular growth pattern observed in Fig. 4.

Finally, the IDMG method is, by construction, unable to consider future interactions when planning for the action in a non-interaction area. In this sense, the IDMG algorithm is "myopic" to such interactions and only handles these as it reaches an interaction area, which can have a negative impact on the performance of the method.

Consider the scenario depicted in Fig. 5. Once again, the two robots must reach the marked states while avoiding simultaneously crossing the narrow pathways. We model this problem using a Dec-SIMDP: the two sets of shaded cells represent two interaction areas in which the robots only get a non-zero penalty by standing simultaneously in the darker state. In this environment, and ignoring the interaction, Robot 1 can reach its goal by using either of the narrow pathways, since both trajectories have the same length. However, Robot 2 should use the upper pathway, since it is significantly faster than using the lower pathway.

By using the IDMG algorithm, Robot 2 goes to the upper pathway while Robot 1 chooses randomly between the two, for example, the upper pathway. In this case, according to the IDMG algorithm, both robots reach the interaction area simultaneously and Robot 1 must move out of the way for Robot 2 to go on. Therefore, Robot 1 takes 10 steps to reach Goal 1 while Robot 2 takes 8 steps to reach Goal 2, for an average of 9 steps to reach the goal. If, instead, Robot 1 takes the lower pathway, the two robots reach their goal states in 8 steps each. The IDMG algorithm chooses between these two possibilities randomly—or, at least, has no way to differentiate between the two. Therefore, IDMG agents take an average of 8.5 steps to reach their goal. We ran 1000 independent trials using the IDMG algorithm in this scenario and, indeed, obtained an average of 8.485 steps-to-goal, with a standard deviation of 0.5.

For comparison purposes, we also ran 1000 independent trials using the LAPSI algorithm in this same scenario. Out of 1000 trials, Robot 1 always picked the lower pathway and the group took an average of 8 steps-to-goal with a variance of 0. This difference could be made arbitrarily large by increasing the "narrow doorway" to an arbitrary number of states, thus causing an arbitrarily large delay.

We conclude this section by providing Table 6 with the value of the bounds from Theorem 4.3 for some of the test environments. Although these bounds are generally loose, they still provide insights on some properties of our methods. The error bounds for MPSI are, in general, larger than those for LAPSI, since the policy estimate for MPSI is, in most scenarios, more crude than that of LAPSI. Moreover, although the bounds exhibit some dependency on the dimension of the problem—larger scenarios tend to have larger bounds,—this is not always the case. For example, while Map 3 is smaller than Map 2, the corresponding bounds are actually larger. Also, ISR is much smaller than CMU but the corresponding bounds are approximately similar.

5. Learning interaction areas

In the previous sections we introduced the Dec-SIMDP model and proposed the MPSI and the LAPSI planning algorithms, both able to leverage the sparse interactions in Dec-SIMDPs to overcome the computational burden associated with more general decision-theoretic models. The Dec-SIMDP model relies on the concept of interaction areas that capture local dependences between groups of agents. These interaction areas include non-interaction states that nevertheless improve the coordination performance of the agent group. We now introduce a learning algorithm that learns those interaction areas.

We start by considering the simple case of a 2-agent transition-independent MMDP given by $\mathcal{M} = (2, \mathcal{X}, (\mathcal{A}_k), \mathsf{P}, r, \gamma)$ where, as before, r is decomposable as

$$r(x, a) = r_1(x_1, a_1) + r_2(x_2, a_2) + r^I(x, a),$$

where r_k corresponds to the individual component of the reward function associated with agent k and r^I is the joint component of the reward function. If $r^I \equiv 0$, each agent can use standard Q-learning to learn its optimal individual policy π_k^*, and the policy $\pi^* = (\pi_1^*, \pi_2^*)$ is optimal for $\mathcal{M}$. However, if $r^I(x, a) \neq 0$ in some state x, the policy learned by each agent using Q-learning and disregarding the existence of the other agent is generally suboptimal [31,32]. The optimal policy in that case must take into account whatever the other agent is doing.

When moving to a Dec-MDP setting, the situation in which $r^I \equiv 0$ poses no additional difficulties, as the problem can still be decomposed into independent single-agent MDPs. If the interaction between the two agents is sparse, we expect each agent to coordinate only around those states x for which $r^I(x, a) \neq 0$—the interaction areas. This coordination may require each agent to have access to state and action information about the other agent. Therefore, interaction areas should comprise the states in which information on the joint state brings improvements in performance over local-state information.

In general, the performance obtained using joint state information in all time steps is never worse than that obtained using this information only at certain situations. However, since we are interested in determining the interaction areas, we are only interested in states in which this information brings an improvement in performance. To this purpose, we add an artificial penalty every time the agent uses joint state information, to ensure that the latter is only used when actually useful.

We augment the individual action space of each agent with one pseudo-action, the COORDINATE action, that incurs the aforementioned penalty and consists of two steps:

1. An *active perception step*, in which the agent tries to determine the local state information of the other agent;
2. A *coordinating step*, in which the agent makes use of the local state information from the other agent, if available, to choose one of its primitive actions.

The active perception step of the COORDINATE action may not succeed; whether or not it actually succeeds is environment dependent. Considering once again a multi-robot navigation scenario, the active perception step can consist for example of the use of an onboard camera to localize the other robot. In this case, the robot is able to localize the other robot only when the latter is in its field-of-view. Another possibility consists of the use of explicit wireless communication, in which one robot requires the other robot to share its location.

Going back to our algorithm, each agent k uses standard Q-learning to estimate $Q_k^*(\bar{x}_k, a_k)$ for all local states $\bar{x}_k$ and all $a_k \in \mathcal{A}_k \cup \{\text{COORDINATE}\}$. The role of the coordinating step is to use the local state information from the other agent, provided by the active perception step, to guide the actual choice of the actions in $\mathcal{A}_k$. To this purpose, each agent k keeps an estimate of a *second Q-function*, the interaction Q-function Q_k^I, defined in terms of the immediate reward of agent k and the value of agent k's policy. Since this policy is defined in terms of Q_k^*, the values of Q_k^I at different state-action pairs are independent, *i.e.*,

$$Q_k^I(x, a) = r_k(x, a_k) + \gamma \sum_{\bar{y}_k \in \mathcal{X}_k} \mathsf{P}_k^a(\bar{x}_k, \bar{y}_k) \max_{b_k \in \mathcal{A}_k} Q_k^*(\bar{y}_k, b_k).$$

The above relation can be used in a Q-learning-like update to estimate the value Q_k^I associated with each individual action $a_k \in \mathcal{A}_k$ in each joint state (x_1, x_2).

Algorithm 2 summarizes our algorithm; π_ℓ is the learning policy *i.e.*, a policy that ensures sufficient exploration, such as an ε-greedy policy, and by $\pi_g(Q, \cdot)$ the greedy policy with respect to function Q. The flag *ActivePercept* is true if the active perception step is successful and the general instruction `QLUpdate`$(Q; x, a, r, y, Q')$ denotes the general Q-learning update defined in Section 2.1.

The update of Q_k^I uses the estimates of Q_k^*, as the individual action at the next step depends on the values in Q_k^* and not on Q_k^I. Hence, the values in Q_k^I only determine the one step behavior of the COORDINATE action, and therefore there is not a direct dependency among entries in Q_k^I corresponding to different states/actions (Fig. 6). Such independence is particularly useful as it implies that, unlike algorithms that learn directly on the joint state-action space, our matrix Q_k^I is sparse—*i.e.*, the number of non-zero elements is small—and thus requires a similar sample complexity as that necessary to learn Q_k^*.

Algorithm 2 Learning algorithm for agent k

Require: Learning policy π_ℓ
Initialize Q_k^* and Q_k^I;
Set $t = 0$;
while (FOREVER) **do**
  Choose $A_k(t)$ using π_ℓ;
  if $A_k(t) =$ COORDINATE **then**
    if *ActivePercept* = TRUE **then**
      $\hat{A}_k(t) = \pi_g(Q_k^I, X(t))$;
    else
      $\hat{A}_k(t) = \pi_g(Q_k^*, \bar{X}_k(t))$;
    end if
    Sample $R_k(t)$ and $\bar{X}_k(t+1)$;
    if *ActivePercept* = TRUE **then**
      QLUpdate$(Q_k^I, X(t), \hat{A}_k(t), R_k(t), \bar{X}_k(t+1), Q_k^*)$;
    end if
  else
    Sample $R_k(t)$ and $\bar{X}_k(t+1)$;
  end if
  QLUpdate$(Q_k^*, \bar{X}_k(t), A_k(t), R_k(t), \bar{X}_k(t+1), Q_k^*)$;
  $t = t + 1$;
end while

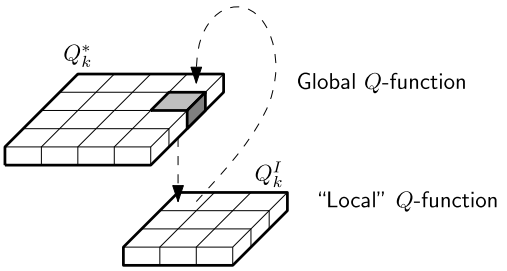


Fig. 6. Illustration of the dependence between the global Q-function, Q_k^*, and the local Q-function, Q_k^I.

Also, the COORDINATE action does not use any joint action information, only joint state information. This fact and the aforementioned independence of the components of Q_k^I associated with different $\bar{x}_k$ makes the learning of Q_k^I similar to a generalized fictitious play process [33]. However, in scenarios where the action information from other agents is necessary for coordination, our algorithm may exhibit poor performance.

So far we have described how to apply our learning algorithm to 2-agent MMDPs. In MMDPs with $N > 2$ agents, and since the local state information from the other agents arises from an active perception step, each agent can only perceive the local state information concerning one other agent. Therefore, in problems for which coordination requires state information concerning more than two agents, the performance of our algorithm is expected to decrease. Another implication is that we can apply Algorithm 2 without modification, by disregarding the identity of the other agent and merely considering that the local information obtained by active perception concerns some other agent. However, if we consider more general active perception processes—*i.e.*, the agent is actually able to perceive the state of all other agents—the algorithm can be trivially modified to address coordination of any number of agents, with an obvious tradeoff in terms of the memory requirements of the algorithm.

5.1. Results

We apply our learning algorithm to the test scenarios in Fig. 3, where we include an individual penalty of -0.1 in all scenarios every time an agent uses the COORDINATE action. The excessive use of such COORDINATE action impacts the sparsity of Q_k^I, which is undesirable for large problems.

We run our learning algorithm in each of the test environments. Table 7 summarizes the number of learning steps allowed in each environments. During learning, exploration is ensured by combining a greedy policy with optimistic initial values [34]. We use an initial learning rate of $\alpha = 0.1$. Every 2×10^4 time steps, this rate is updated as $\alpha \leftarrow 0.1 \times \alpha$. For comparison purposes, we also run:

- One instance of single-agent Q-learning for each robot, in the absence of other robots in the environment. The robots thus learn how to reach their goal but not how to avoid miscoordination, since there are no other robots in the environment during the learning stage and hence the learner never actually experiences the miscoordination penalty.
- One instance of single-agent Q-learning for each robot, in the presence of other robots in the environment. The robots thus learn how to best reach their goal while avoiding interaction *using only individual state information*. Due to the

Table 7
Number of time-steps used for learning in each of the test environments.

Environment	Learning steps
Map 1	10^4
Map 2	10^5
Map 3	10^4
CIT	5×10^5
CMU	2×10^5
ISR	5×10^5
MIT	2×10^5
PENTAGON	3×10^5
SUNY	2×10^5

Table 8
Distinction between different learning approaches used for comparison. The second and third columns indicate whether learning and test are conducted in the presence or absence of other agents. The last column indicates state observability.

Approach	Learning	Test	State observability
Indiv.	Single	Multiple	Local
Non-coop.	Multiple	Multiple	Local
Coop.	Multiple	Multiple	Local + Active perc.
Opt.	Multiple	Multiple	Global

Table 9
Total discounted reward for each of the four different algorithms in each of the test scenarios. The results are averaged over 1000 independent Monte Carlo trials. For each environment, bold entries correspond to optimal values (differences are not statistically significant).

Environment	Indiv.	Non-coop.	Coop.	Opt.
Map 1	2.826	3.046	6.304	**12.059**
Map 2	1.607	4.487	9.957	**11.108**
Map 3	5.470	7.002	7.171	**13.837**
Map 4	−3.180	0.000	4.229	**16.447**
CIT	9.790	**11.002**	**11.112**	**11.128**
CMU	−1.319	0.910	1.630	**2.840**
ISR	10.695	**14.296**	**14.258**	**14.407**
MIT	6.554	**6.666**	**6.502**	**6.705**
PENTAGON	14.676	12.918	**16.554**	16.016
SUNY	**11.167**	**11.104**	**11.135**	**11.149**

general lack of knowledge about the global state, the agents in this second comparison set correspond to *independent learners* in the sense of [31].

Table 8 summarizes this information. The second and third columns show whether learning and testing are conducted in the presence or absence of other agents (corresponding to the labels "Multiple" and "Single", respectively). The last column indicates whether the agents have access to only local state information, global state information or, in the case of our method, local state information complemented by state information from the active perception step of the COORDINATE action.

We evaluate each of the learned policies in the different environments, performing 1000 independent Monte Carlo trials. Each trial lasts for 250-time-steps, during which each learned policy (Indiv., Non-coop., and Coop.) is evaluated in the original problem, *i.e.*, with two/four robots moving simultaneously in the environment. For comparison, we also present the results obtained with the optimal fully observable MMDP policy from Section 4. However, since the framework considered here is different from that considered in Section 4, the results from the optimal MMDP policy cannot be used for direct comparison, but only as a reference to better assess the performance of the learning methods. Table 9 presents the results in terms of total discounted reward.

The results in Table 9 show that our algorithm generally outperforms the two other learning methods, matching the optimal MMDP policy in some of the test scenarios. However, the behavior of the different algorithms varies in the different scenarios. For example, individual agents never experience miscoordination penalties during learning. Therefore, they act as if such penalty does not exist, which impacts negatively the total discounted reward they receive as seen, for example, in Map 2. The non-cooperative agents experience miscoordination penalties during learning and are better able to avoid them, although their lack of joint state information prevents them from attaining optimal performance. In particular, in the Map 4 environment, they adopt cautious policies that prefer avoiding penalties to attaining the agents' goals. Such cautious policies explain the total discounted reward of 0.00 and the steps to goal.

Table 10
Steps-to-goal for each of the four different algorithms in each of the test scenarios. The results are averaged over 1000 independent Monte Carlo trials. For each environment, bold entries correspond to optimal values (differences are not statistically significant). Numbers in parentheses indicate that not all agents reached the goal.

Environment	Indiv.	Non-coop.	Coop.	Opt.
Map 1	**10.862**	**10.841**	(10.821)	10.977
Map 2	*12.900*	*12.800*	17.500	**12.546**
Map 3	*8.303*	**8.154**	(8.467)	*8.267*
Map 4	**8.244**	$>$ 250.00	(4.999)	**5.001**
CIT	*13.542*	*13.500*	*13.520*	**12.512**
CMU	40.100	39.800	(39.815)	**39.340**
ISR	*8.216*	*8.211*	*8.226*	**7.440**
MIT	**24.100**	**24.600**	**25.300**	**22.444**
PENTAGON	**5.300**	**5.500**	**4.900**	**5.365**
SUNY	**12.441**	**12.558**	**12.492**	**12.539**

Table 11
Number of crashes for the four different algorithms in each of the test scenarios. The results are averaged over 1000 independent Monte Carlo trials. For each environment, bold entries correspond to optimal values (differences are not statistically significant).

Environment	Indiv.	Non-coop.	Coop.	Opt.
Map 1	0.408	0.397	**0.000**	**0.000**
Map 2	0.500	0.300	**0.000**	**0.000**
Map 3	0.303	0.261	**0.000**	**0.000**
Map 4	1.911	**0.000**	**0.000**	**0.000**
CIT	**0.000**	**0.000**	**0.000**	**0.000**
CMU	0.500	0.100	**0.000**	**0.000**
ISR	0.012	0.005	**0.006**	**0.000**
MIT	**0.000**	**0.000**	**0.000**	**0.000**
PENTAGON	0.100	0.200	**0.000**	**0.000**
SUNY	**0.000**	**0.000**	**0.000**	**0.000**

Tables 10 and 11 show the number of steps-to-goal and the number of miscoordinations per trial for each of the test environments. Our algorithm exhibits nearly no miscoordinations in all environments, which means that the robots are able to coordinate in their choice of path. However, in some scenarios, this is attained at the cost of having only one of the agents reach the goal, corresponding to the values in parenthesis in Table 10. This indicates that, in these scenarios, coordination would require incurring in an excessive penalty arising from using the COORDINATE action, and the agents opt by sacrificing reaching one of the goals. This is an interesting aspect of our learning approach: the agents trade-off the cost of the COORDINATE action by the usefulness of the information provided. This is in contrast with the individual agents: they always reach their goal in minimum time, but at a cost of severe miscoordination penalties that they never experience during learning, and hence their poor performance in terms of total discounted reward.

Our results also confirm that interaction in the larger environments is much sparser than that in the smaller environments. In some of the test scenarios (*e.g.*, CIT), the environment and the initial/goal positions for both robots are such that explicit coordination actions are not really necessary to avoid miscoordinations. Therefore, both the individualistic Q-learners and the non-cooperative Q-learners should be able to attain a good performance. In general, as seen in Tables 9 through 11, there are some environments in which all methods attain similar performance, indicating that no coordination actions are necessary, since both the Individual and Non-coordinated approaches are able to attain optimal performance. The fact that our algorithm attains a similar performance means that it learns that the COORDINATE action is not necessary, otherwise the total discounted reward would be inferior to that of the other methods.

We also analyze how the increasing penalties for miscoordination affect the performance of the robots in all scenarios. We run all three learning algorithms for different values of the miscoordination penalty and evaluated the obtained policies. As expected, as the penalty for miscoordination increases, the total discounted reward of those methods that are unable to avoid miscoordinations (Table 11) decreases accordingly, accentuating the differences already observed in Table 10. The reliability of the performance of the individualistic and non-coordinated policies—translated in the variance of the observed performance—is also severely affected as the miscoordination penalty increases.

Finally, the non-cooperative Q-learning algorithm is able to surpass the individualistic Q-learning in several scenarios. This is particularly evident by observing the performance as the miscoordination penalty increases, and can be interpreted as the non-cooperative agents having to act with increasing caution due to the penalties experienced during learning.

In conclusion, our proposed method is able learn to coordinate only when coordination is necessary to attain good performance. In several scenarios, our learning agents resort to coordination actions in those states where joint state information leads to improved performance, and these states corresponds precisely to the interaction areas defined in Section 3. In other scenarios, our agents are also able to learn to trade-off the benefits of coordination actions against the costs of such actions.

6. Related work

A wide range of models have been proposed to formalize decision-making problems in multiagent systems, including stochastic games [35], multiagent MDPs [36], Dec-POMDPs [37,16], and I-POMDPs [38]. These models differ in scope, assumptions, and computational complexity [26]. For example, multiagent MDPs are can be solved in polynomial time, while finite-horizon Dec-MDP and Dec-POMDP models are NEXP-complete, even for the benign 2-agent scenarios, and complexity results are even worse in non-cooperative settings.

Efforts to handle the worst-case complexity of decentralized multiagent models led to approximate methods that trade-off optimality and computability (*e.g.*, [39,40]), and reduced models that trade off representability and computability. Several such models assume that the interaction among agents can be simplified in different ways. For example, in transition-independent Dec-MDPs, the transition probabilities for each agent depend solely on its own actions and local states. Transition independent Dec-MDPs are NP-complete and can be solved using the Coverage Set algorithm [41,27].

Our Dec-SIMDP model can be viewed as an extension of transition-in dependent Dec-MDP, as it allows the transition probabilities for an agent to dependent on the actions of other agents in interaction areas, where joint state observability or free instantaneous communication is assumed. We expect the worst-case complexity of a Dec-SIMDP to be between that of transition-independent Dec-MDPs and general Dec-MDPs.

Local interactions have also been exploited in other multiagent scenarios. For example, several works propose the use of hierarchical approaches that subdivide an overall task in a hierarchy of subtasks, each restricted to the states and actions relevant to that particular subtask [5,42,4]. The subtasks are conducted individually by each agent and do not require the local state of different agents to be shared. Going up the hierarchy corresponds to moving from low-level "local" tasks to higher-level "global" tasks, in which coordination is necessary and must be accounted for explicitly. Since execution at the highest level corresponds to several low-level time steps, communication needs are minimized.

Coordination graphs [6,43] capture local dependences between the different agents in an MMDP, allowing the overall Q-function to be decomposed into local Q-functions that can be optimized individually. Coordination graphs have been used for efficient planning and learning in large multiagent MDPs [44]. Although the problems to which coordination graphs have been applied are significantly different from those we study, the interactions between agents captured by coordination graphs are related to our notion of interaction areas in our Dec-SIMPD model, and it would be interesting to investigate the use of a graphical structure to compactly represent the dependencies among agents in a Dec-SIMDP.

The coordination graph structure has also been learned from experience based on the concept of utile coordination [7]. Although using a fundamentally different approach, this work relates to our learning of interaction areas in that both methods infer where joint state information can improve the performance of the agents. We differ in the fact that our agents explicitly learn how to trade-off the benefits of querying the other agents' states with the environment-imposed limitations of this querying process and the associated cost. Our method captures the impact that communication costs may have both on the decision process and on the process of learning these inter-agent dependencies.

Both hierarchical approaches and coordination graphs discussed above exploit local interactions among the agents and should thus be able to accommodate some level of partial state observability.

In the game-theoretic literature, local dependences between players in large games have also been explored. For example, *graphical games* [9] represent n-player matrix games as undirected graphs, where players correspond to nodes, and the payoff for each node depends only on the actions of its direct neighbors in the graph. *Action-graph* games further generalize the concept of graphical games, exploring sparser dependences between players in a game [10]. Multiple algorithms have been proposed to compute Nash equilibria in this class of games, mostly relying on "continuation methods" [45], where a known solution for a simple game is gradually perturbed toward a solution to the desired game. Continuation methods run in time exponential in the in-degree of the action-graph, and not in its number of nodes. Therefore, games with many context-specific independences yield sparsely connected action-graph games, leading to exponential savings in computational time. In some classes of problems, additional structure of the corresponding action-graph games can lead even to more efficient computation of equilibria [46,47].

The main concept of our Dec-SIMDP model was originally proposed under the designation of *interaction-driven Markov games* [30], with the associated IDMG algorithm. Our contributions in this paper extend the original IDMG formulation in several aspects. We formalize the relation between Dec-MDPs and Dec-SIMDPs, introducing concepts such as interaction areas and interaction states. We presented two general heuristic planning algorithms with corresponding convergence analysis and error bounds that overcome some limitations of the original IDMG algorithm: (i) outside interaction areas, both LAPSI and MPSI reason about future interactions, unlike IDMG which only considers interactions when they actually occur; (ii) in interaction areas, LAPSI and MPSI also offer computational advantages over IDMG, since the latter requires the computation of several equilibria.

Other closely related models to the Dec-SIMDP model are those that explore *event-driven interactions* [48–50], and distributed POMDPs with *coordination locales* (DCPLs) [13]. Particularly in DCPLs, each agent is assumed independent of all other agents except on previously specified coordination locales. As in the IDMG algorithm, the proposed TREMOR algorithm for DCPLs models each agent k using a POMDP model that is solved to yield a policy π_k for that agent. Coordination locales are handled by modifying the POMDP model for each agent taking the policies of the other agents into account. Instead, our approach assumes that interactions are fully observable, which can be used for coordination.

More recently, an information-theoretic measure of inter-agent influence has been proposed under the designation of *influence gap* [28], which indicates how much the actions of one agent determine the actions of the other agents in the optimal policy. Such influence gap then attempts at quantifying the level of dependence between the different agents. As expected, larger influence gaps, corresponding to smaller inter-agent influence, typically translate into less computational complexity. While the influence gap is a measure of *global* inter-agent influence, our interaction areas capture *local* interactions between the agents. Larger or numerous interaction areas typically lead to harder problems. As future work, it would be interesting to relate the number or size of interaction areas in Dec-SIMDPs in terms of the proposed influence gap, using the latter to assess the potential usefulness of the Dec-SIMDP model in particular problems.

7. Conclusion and future work

In this work, we analyzed how local interactions in a multiagent system can be used to simplify the process of decentralized decision making. We introduced Dec-SIMDPs, a new decision-theoretic model for decentralized multiagent systems with sparse interaction that explicitly distinguishes the situations in which the agents in a team must coordinate from those in which they can act independently. We formalized the relation between Dec-MDPs and Dec-SIMDPs, establishing Dec-SIMDPs as particular instances of Dec-MDPs. We presented MPSI and LAPSI, two general heuristic planning approaches that reduce planning in a Dec-SIMDP to a sequence of planning problems in single-agent POMDPs. We analyzed particular instances of MPSI and LAPSI and derived bounds for the quality of the obtained policies. Finally, we presented an RL algorithm that can be used to learn interaction areas in a Dec-SIMDP. We illustrated the application of all algorithms throughout the paper in several multiagent navigation scenarios.

Both instances of MPSI and LAPSI used in the experimental results rely on having each agent track the other agents in the environment using a belief vector that is then used to choose the actions. The difference between the two algorithms lies in the assumed policy for the other agents. MPSI assumes each of the other agents to be completely driven by its individual goals, thus discarding whatever interaction there may be. In the cases where these interactions are negative, MPSI agents then act more cautiously. In contrast, the LAPSI agent assumes that the other agents are team-players, in that they choose their actions for the common goal of the group. Hence, the LAPSI agent adopts a policy that is closer to the actual optimal fully-observable policy. The LAPSI algorithm successfully leverages the particular independences between the different agents to attain efficient and yet near-optimal performance. In MPSI and LAPSI, these modeling strategies are used to abstract the decision process of each agent into a single-agent decision process—namely, a POMDP. Although we illustrated our methods using a Q_{MDP}-like approach, the same principle can be used with any other POMDP solver.

The differences between MPSI and LAPSI may provide additional information in defining the interaction areas. While MPSI relies on the optimal policies for the individual MDPs in the Dec-SIMDP model, LAPSI relies on the joint policy for the underlying MMDP. Since outside interaction areas we expect the actions of the different agents to be approximately independent, the interactions areas should be those in which the estimated policies using the individual MDPs and the joint MMDP disagree. This provides one recipe for choosing the interaction states as those in which individual state-information is not sufficient to determine the best action.

The results of learning algorithm open several interesting questions. One first issue concerns the dependence of the performance of the algorithm on the cost of the COORDINATE action. In fact, we observed that our agents are able to learn a trade-off between the benefits arising from good coordination and the cost of that same coordination. Such trade-off is similar to the problem of exchanging reward by information arising in POMDPs [22] and it would be interesting to analyze both how this trade-off is influenced by the cost of the COORDINATE action and how such trade-off extends to situations where further partial state observability is considered.

Another aspect open to future investigation regards the performance guarantees of the algorithm. As remarked in Section 5, the parallel learning process taking place when the agent executes the COORDINATE action bears significant resemblances to a generalized fictitious play. It would be interesting to analyze how the convergence guarantees of fictitious play can be translated to our particular learning algorithm. The scenarios used in our work, in which our learning algorithm exhibited interesting performance, were focused on localized coordination, based on the underlying assumption of the advantages of our model and learning. As coordination increases to a more global level, the learning algorithm may need to be adapted to attain the desired performance.

In view of the completely independent way by which the agents learn, it remains to investigate if the learning algorithm would be applicable to different reward functions for the different agents, or other scenarios in which the agents are not completely independent in terms of dynamics but exhibit some weak coupling—for example in the states where interaction occurs. It would also be interesting to explore our ideas in more general models such as Dec-POMDPs, alleviating the requirement of full local state observability.

Acknowledgements

The authors would like to acknowledge the thorough comments of the anonymous reviewers that significantly contributed to improve the quality of the presentation, and the helpful discussions with Matthijs Spaan. This research was partially sponsored by the project CMU-PT/SIA/0023/2009 under the Carnegie Mellon Portugal Program and its Information and Communications Technologies Institute, and the Portuguese *Fundação para a Ciência e Tecnologia*. The views and conclusions contained in this document are those of the authors only.

Appendix A. Proofs

A.1. Proof of Theorem 4.1

We show that the generalized α-vectors can be computed using a convergent dynamic-programming-like approach, by iterating over the recurrent expression in (16).

In a Dec-SIMDP verifying the conditions of the theorem, a generalized α-vector $\boldsymbol{\alpha}_k$ is actually a $|\mathcal{X}| \times |\mathcal{A}_k|$ matrix with component (x, a_k) given by $\boldsymbol{\alpha}_k(x, a_k)$. For a general matrix $|\mathcal{X}| \times |\mathcal{A}_k|$ matrix W, let $\mathbf{T}_k$ be the operator defined as

$$(\mathbf{T}_k W)(x, a_k) = r_{\pi_{-k}}(x, a_k) + \gamma \sum_{y \in \mathcal{X}_I} \mathsf{P}_{\pi_{-k}}(x, a_k, y) \max_{u_k} W(y, u_k) + \gamma \max_{u_k} \sum_{y \notin \mathcal{X}_I} \mathsf{P}_{\pi_{-k}}(x, a_k, y) W(y, u_k),$$

where $(\mathbf{T}_k W)(x, a_k)$ denotes the element (x, a_k) of the matrix $\mathbf{T}_k W$. The operator $\mathbf{T}_k$ is closely related with the Bellman operator $\mathbf{H}$ introduced in (3). We establish the assertion of the theorem by showing $\mathbf{T}_k$ to be a contraction in the supremum norm. In fact, we have

$$\begin{aligned} \|\mathbf{T}_k W_1 - \mathbf{T}_k W_2\|_\infty &= \max_{x, a_k} \big| (\mathbf{T}_k W_1)(x, a_k) - (\mathbf{T}_k W_2)(x, a_k) \big| \\ &\leqslant \gamma \max_{x, a_k} \sum_y \mathsf{P}_{\pi_{-k}}(x, a_k, y) \max_{u_k} \big| W_1(y, u_k) - W_2(y, u_k) \big|, \end{aligned}$$

where the last inequality follows from Jensen's inequality, implying that

$$\|\mathbf{T}_k W_1 - \mathbf{T}_k W_2\|_\infty \leqslant \gamma \max_{x, a_k} \big| W_1(x, a_k) - W_2(x, a_k) \big| = \gamma \|W_1 - W_2\|_\infty.$$

We have thus shown that $\mathbf{T}_k$ is a contraction in the supremum norm, which implies that

- $\mathbf{T}_k$ has a unique fixed-point, corresponding to the generalized α-vectors;
- $\mathbf{T}_k$ can be used to compute the generalized α-vectors in a dynamic-programming-like fashion, using the update rule

$$\boldsymbol{\alpha}_k^{(n+1)}(x, a_k) = \big(\mathbf{T}_k \boldsymbol{\alpha}_k^{(n)}\big)(x, a_k),$$

where $\boldsymbol{\alpha}_k^{(n)}$ denotes the nth estimate of $\boldsymbol{\alpha}_k(x, a_k)$. □

A.2. Proof of Theorem 4.2

We show that the problem of computing the generalized α-vectors for a Dec-SIMDP verifying the conditions of the theorem is equivalent in terms of complexity to that of solving an MDP whose dimension depends polynomially on the dimension of the original Dec-SIMDP. In particular, we show that computing the generalized α-vectors for such Dec-SIMDP is equivalent to computing the optimal Q-function for an MDP. Since MDPs are known to be P-complete [25], the desired result follows.

We rewrite (16) as

$$\begin{aligned} \boldsymbol{\alpha}_k(x, a_k) = r_{\pi_{-k}}(x, a_k) &+ \gamma \sum_{y \in \mathcal{X}_I} \mathsf{P}_{\pi_{-k}}(x, a_k, y) \max_{u_k} \boldsymbol{\alpha}_k(y, u_k) \\ &+ \gamma \sum_{y \notin \mathcal{X}_I} \mathsf{P}_{\pi_{-k}}(x, a_k, y) \max_{u_k} \eta_{x a_k} \sum_{z \notin \mathcal{X}_I} \mathsf{P}_{\pi_{-k}}(x, a_k, z) \boldsymbol{\alpha}_k(z, u_k), \end{aligned} \tag{A.1}$$

where

$$\eta_{x a_k} = \frac{1}{\sum_{y \notin \mathcal{X}_I} \mathsf{P}_{\pi_{-k}}(x, a_k, y)}.$$

We now construct an MDP $\hat{\mathcal{M}} = (\hat{\mathcal{X}}, \mathcal{A}_k, \hat{\mathsf{P}}, \hat{r})$, where $\hat{\mathcal{X}} = \mathcal{X} \cup \mathcal{X} \times \mathcal{A}_k$ and

$$\hat{\mathsf{P}}(\hat{x}, a_k, \hat{y}) = \begin{cases} \mathsf{P}_{\pi_{-k}}(\hat{x}, a_k, \hat{y}) & \text{if } \hat{x} \in \mathcal{X} \text{ and } \hat{y} \in \mathcal{X}_I, \\ 1/\eta_{\hat{x} a_k} & \text{if } \hat{x} \in \mathcal{X} \text{ and } \hat{y} = (\hat{x}, a_k), \\ \eta_{z u_k} \sum_{y \notin \mathcal{X}_I} \mathsf{P}_{\pi_{-k}}(z, u_k, y) \mathsf{P}_{\pi_{-k}}(y, a_k, \hat{y}) & \text{if } \hat{x} = (z, u_k) \text{ and } \hat{y} \in \mathcal{X}_I, \\ \eta_{z u_k} \mathsf{P}_{\pi_{-k}}(z, u_k, y)/\eta_{y a_k} & \text{if } \hat{x} = (z, u_k),\ \hat{y} = (y, a_k), \text{ and } y \notin \mathcal{X}_I, \\ 0 & \text{otherwise.} \end{cases}$$

These probabilities are well defined since, for $\hat{x} \in \mathcal{X}$,

$$\sum_{\hat{y}} \hat{\mathsf{P}}(\hat{x}, a_k, \hat{y}) = \sum_{\hat{y}\in\mathcal{X}_I} \mathsf{P}_{\pi_{-k}}(\hat{x}, a_k, \hat{y}) + 1/\eta_{\hat{x}a_k} = \sum_{\hat{y}\in\mathcal{X}_I} \mathsf{P}_{\pi_{-k}}(\hat{x}, a_k, \hat{y}) + \sum_{y\notin\mathcal{X}_I} \mathsf{P}_{\pi_{-k}}(\hat{x}, a_k, y) = 1$$

and, for $\hat{x} = (z, u_k)$

$$\sum_{\hat{y}} \hat{\mathsf{P}}(\hat{x}, a_k, \hat{y}) = \eta_{zu_k} \sum_{y\notin\mathcal{X}_I} \mathsf{P}_{\pi_{-k}}(z, u_k, y)\Bigg(\sum_{\hat{y}\in\mathcal{X}_I} \mathsf{P}_{\pi_{-k}}(y, a_k, \hat{y}) + 1/\eta_{ya_k}\Bigg) = \eta_{zu_k} \sum_{y\notin\mathcal{X}_I} \mathsf{P}_{\pi_{-k}}(z, u_k, y) = 1.$$

The reward function for $\hat{\mathcal{M}}$ is

$$\hat{r}(\hat{x}, a_k) = \begin{cases} r_{\pi_{-k}}(\hat{x}, a_k) & \text{if } \hat{x}\in\mathcal{X}, \\ \eta_{zu_k} \sum_{y\notin\mathcal{X}_I} \mathsf{P}_{\pi_{-k}}(z, u_k, y) r_{\pi_{-k}}(y, a_k) & \text{if } \hat{x} = (z, u_k). \end{cases}$$

The optimal Q-function for this MDP verifies the recursive relation (2), namely:

$$Q^*(\hat{x}, a_k) = \hat{r}(\hat{x}, a_k) + \gamma \sum_{\hat{y}\in\hat{\mathcal{X}}} \hat{\mathsf{P}}(\hat{x}, a_k, \hat{y}) \max_{u_k} Q^*(\hat{y}, u_k).$$

For $\hat{x}\in\mathcal{X}$, and replacing the definitions of $\hat{\mathsf{P}}$ and $\hat{r}$,

$$Q^*(\hat{x}, a_k) = r_{\pi_{-k}}(\hat{x}, a_k) + \gamma \sum_{\hat{y}\in\mathcal{X}_I} \mathsf{P}_{\pi_{-k}}(\hat{x}, a_k, \hat{y}) \max_{u_k} Q^*(\hat{y}, u_k) + \frac{\gamma}{\eta_{\hat{x}a_k}} \max_{u_k} Q^*\big((\hat{x}, a_k), u_k\big). \tag{A.2}$$

Similarly, for $\hat{x} = (z, u_k)$,

$$\begin{aligned} Q^*(\hat{x}, a_k) &= \eta_{zu_k} \sum_{y\notin\mathcal{X}_I} \mathsf{P}_{\pi_{-k}}(z, u_k, y)\Bigg[r_{\pi_{-k}}(y, a_k) + \gamma \sum_{\hat{y}\in\mathcal{X}_I} \mathsf{P}_{\pi_{-k}}(y, a_k, \hat{y}) \max_{u_k} Q^*(\hat{y}, u_k) \\ &\qquad + \frac{\gamma}{\eta_{ya_k}} \max_{u_k} Q^*\big((y, a_k), u_k\big)\Bigg] \\ &= \eta_{zu_k} \sum_{y\notin\mathcal{X}_I} \mathsf{P}_{\pi_{-k}}(z, u_k, y) Q^*(y, a_k). \end{aligned} \tag{A.3}$$

Replacing (A.2) in (A.3) yields

$$Q^*(\hat{x}, a_k) = r_{\pi_{-k}}(\hat{x}, a_k) + \gamma \sum_{\hat{y}\in\mathcal{X}_I} \mathsf{P}_{\pi_{-k}}(\hat{x}, a_k, \hat{y}) \max_{u_k} Q^*(\hat{y}, u_k) + \frac{\gamma}{\eta_{\hat{x}a_k}} \max_{u_k} \eta_{\hat{x}a_k} \sum_{y\notin\mathcal{X}_I} \mathsf{P}_{\pi_{-k}}(\hat{x}, a_k, y) Q^*(y, u_k),$$

which is (A.1). As such, in computing the optimal Q-function for the MDP $\hat{\mathcal{M}}$, we compute the generalized α-vectors for the original Dec-SIMDP as

$$\boldsymbol{\alpha}_k(x, a_k) = Q^*(x, a_k).$$

Since the dimension of the new MDP grows linearly with the dimension of the generalized α-vectors, and, hence, with the dimension of the corresponding Dec-SIMDP, the statement of the theorem follows. □

A.3. Proof of Theorem 4.3

For a general MDP $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$, if $\hat{\pi}$ is the greedy policy with respect to a function $\hat{Q}$, *i.e.*, if

$$\hat{\pi}(x) = \arg\max_{a\in\mathcal{A}} \hat{Q}(x, a)$$

for all $x\in\mathcal{X}$, then

$$\big\| V^{\hat{\pi}} - V^* \big\|_\infty \leqslant \frac{2\gamma}{(1-\gamma)^2} \mathbf{BE}(\hat{Q}), \tag{A.4}$$

where $\mathbf{BE}(\hat{Q})$ is the Bellman error associated with the function $\hat{Q}$,[7]

$$\mathbf{BE}(\hat{Q}) = \sup_{x,a} \Bigg| r(x, a) + \gamma \sum_{y} \mathsf{P}(x, a, y) \max_{u} \hat{Q}(y, u) - \hat{Q}(x, a) \Bigg|.$$

[7] This fact follows from Proposition 6.1 of [18].

In our Dec-SIMDP setting, since we are assuming the policy π_{-k} to be fixed and known, the decision process from agent k's perspective is a standard POMDP. Since a POMDP can be recast as an equivalent belief MDP, it follows that the relation (A.4) also holds for POMDPs. Writing down the Bellman error for a general POMDP $(\mathcal{X}, \mathcal{A}, \mathcal{Z}, \mathsf{P}, \mathsf{O}, r, \gamma)$ thus yields

$$\mathbf{BE}(\hat{Q}) = \sup_{\mathbf{b},a} \left| \sum_x \mathbf{b}_x \left[r(x,a) + \gamma \sum_{z,y} \mathsf{P}(x,a,y)\mathsf{O}(y,a,z) \max_u \hat{Q}\left(\mathbf{b}'_{za}, u\right) \right] - \hat{Q}(\mathbf{b}, a_k) \right|.$$

For simplicity of notation, we consider the *Bellman error at* $(\mathbf{b}, a)$ to be

$$\mathbf{BE}(\hat{Q}, \mathbf{b}, a) = \left| \sum_x \mathbf{b}_x \left[r(x,a) + \gamma \sum_{z,y} \mathsf{P}(x,a,y)\mathsf{O}(y,a,z) \max_u \hat{Q}\left(\mathbf{b}'_{za}, u\right) \right] - \hat{Q}(\mathbf{b}, a) \right|.$$

For the POMDP as perceived by agent k in our Dec-SIMDP setting,

$$\mathbf{BE}(\hat{Q}, \mathbf{b}, a_k) = \left| \sum_{x_{-k}} \mathbf{b}_{x_{-k}} \left[r_{\pi_{-k}}(x, a_k) + \gamma \sum_{z,y} \mathsf{P}_{\pi_{-k}}(x, a_k, y)\mathsf{O}(y,z) \max_u \hat{Q}\left(\mathbf{b}'_{za_k}, u\right) \right] - \hat{Q}(\mathbf{b}, a_k) \right|$$

which, replacing the definitions of $\hat{Q}(\mathbf{b}, a_k)$ and the generalized α-vectors yields, leads to

$$\begin{aligned} \mathbf{BE}(\hat{Q}, \mathbf{b}, a_k) = \gamma \Bigg| & \sum_{x_O, y_O} \mathbf{b}_{x_O} \mathsf{P}_{\pi_{-k}}(x_O, a_k, y_O) \max_{u_k} \sum_{x_{-O}, y_{-O}} \mathbf{b}_{x_{-O}} \mathsf{P}_{\pi_{-k}}(x_{-O}, y_{-O}) \boldsymbol{\alpha}_k(y, u_k) \\ & - \sum_{x, y_O} b_{x_O} \mathbf{b}_{x_{-O}} \mathsf{P}_{\pi_{-k}}(x_O, a_k, y_O) \max_{u_k} \sum_{y_{-O}} \mathsf{P}_{\pi_{-k}}(x_{-O}, y_{-O}) \boldsymbol{\alpha}_k(y, u_k) \Bigg|, \end{aligned}$$

using the notation introduced in Section 4.2. Letting

$$\Lambda_k(x_{-O}, y_O, u_k) = \sum_{y_{-O}} \mathsf{P}_{\pi_{-k}}(x_{-O}, y_{-O}) \boldsymbol{\alpha}_k(y, u_k),$$

then

$$\mathbf{BE}(\hat{Q}, \mathbf{b}, a_k) \leqslant \gamma \max_{y_O} \left| \max_{u_k} \sum_{x_{-O}} \mathbf{b}_{x_{-O}} \Lambda(x_{-O}, y_O, u_k) - \sum_{x_{-O}} \mathbf{b}_{x_{-O}} \max_{u_k} \Lambda(x_{-O}, y_O, u_k) \right|.$$

In order to bound the right-hand side of the expression above, we need two auxiliary results that generalize some of the bounds in [51] to the case of functions defined over $\mathbb{R}^n$ and are of independent interest *per se*.

Lemma 1. *Let $\{x_k, k = 1, \ldots, M\}$ be a set of points in $\mathbb{R}^n$, for some (finite) n, and $\{\beta_k,\ k = 1, \ldots, M\}$ a set of corresponding weights, verifying $0 \leqslant \beta_k \leqslant 1,\ k = 1, \ldots, M$ and $\sum_k \beta_k = 1$. Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function. Then, it holds that*

$$\sum_k \beta_k f(x_k) - f\left(\sum_k \beta_k x_k \right) \leqslant \beta^* \left[\sum_k f(x_k) - M f\left(\sum_k x_k / M \right) \right], \tag{A.5}$$

where

$$\beta^* = \max_k \beta_k.$$

Proof. The proof essentially follows that of Lemma 1 in [51]. Let k^* be such that $\beta^* = \beta_{k^*}$. Eq. (A.5) can be rewritten as

$$\sum_{k \neq k^*} (\beta^* - \beta_k) f(x_k) + f\left(\sum_k \beta_k x_k \right) \geqslant \beta^* M f\left(\sum_k x_k / M \right)$$

or, equivalently,

$$\sum_{k \neq k^*} \frac{\beta^* - \beta_k}{M\beta^*} f(x_k) + \frac{1}{M\beta^*} f\left(\sum_k \beta_k x_k \right) \geqslant f\left(\sum_k x_k / M \right).$$

Since

$$\sum_{k \neq k^*} \frac{\beta^* - \beta_k}{M\beta^*} + \frac{1}{M\beta^*} = 1,$$

then

$$\sum_{k\neq k^*} \frac{\beta^* - \beta_k}{M\beta^*} f(x_k) + \frac{1}{M\beta^*} f\Big(\sum_k \beta_k x_k\Big) \geqslant f\Big(\sum_{k\neq k^*} \frac{\beta^* - \beta_k}{M\beta^*} x_k + \frac{1}{M\beta^*} \sum_k \beta_k x_k\Big) = f\Big(\sum_k x_k/M\Big),$$

where the first inequality follows from Jensen's inequality, implying (A.5). □

Corollary 1. *Let $\{x_i, i = 1, \ldots, N\}$ be a set of points in $\mathbb{R}^n$, for some finite n, and $\{p_i,\ i = 1, \ldots, N\}$ a corresponding sequence of weights verifying $0 \leqslant p_i \leqslant 1$ and $\sum_i p_i = 1$. Let U denote a closed convex set that can be represented as the convex hull of a set of points $\{a_k,\ k = 1, \ldots, M\}$ in $\mathbb{R}^n$. Let $f : U \to \mathbb{R}$ be a convex function. Then, it holds that*

$$\sum_i p_i f(x_i) - f\Big(\sum_i p_i x_i\Big) \leqslant \sum_k f(a_k) - Mf\Big(\sum_k a_k/M\Big). \tag{A.6}$$

Proof. Each x_i can be written as

$$x_i = \sum_k \lambda_{ik} a_k,$$

with $0 \leqslant \lambda_{ik} \leqslant 1$ and $\sum_k \lambda_{ik} = 1, i = 1, \ldots, n$. Then,

$$\begin{aligned}\sum_i p_i f(x_i) - f\Big(\sum_i p_i x_i\Big) &= \sum_i p_i f\Big(\sum_k \lambda_{ik} a_k\Big) - f\Big(\sum_i p_i \sum_k \lambda_{ik} a_k\Big)\\ &\leqslant \sum_k f(a_k) \sum_i p_i \lambda_{ik} - f\Big(\sum_k a_k \sum_i p_i \lambda_{ik}\Big).\end{aligned}$$

Letting $\beta_k = \sum_i p_i \lambda_{ik}$,

$$\sum_i p_i f(x_i) - f\Big(\sum_i p_i x_i\Big) \leqslant \sum_k \beta_k f(a_k) - f\Big(\sum_k \beta_k a_k\Big).$$

Finally, applying Lemma 1,

$$\begin{aligned}\sum_i p_i f(x_i) - f\Big(\sum_i p_i x_i\Big) &\leqslant \sum_k \beta_k f(a_k) - f\Big(\sum_k \beta_k a_k\Big)\\ &\leqslant \beta^* \Big[\sum_k f(a_k) - Mf\Big(\sum_k a_k/M\Big)\Big]\\ &\leqslant \sum_k f(a_k) - Mf\Big(\sum_k a_k/M\Big). \qquad \square\end{aligned}$$

The two bounds differ as (A.5) depends on the function f and the set of points $\{x_k, k = 1, \ldots, M\}$, while (A.6) depends only on the function f and on the set U.

For any given $u_k^* \in \mathcal{A}_k$, $x_{-O}^* \in \mathcal{X}_{-O}$ and $y_O^* \in \mathcal{X}_O$, it holds that $\Lambda_k(x_{-O}^*, y_O^*, u_k^*)$ lies in the convex hull of the set of alpha-vectors $\boldsymbol{\alpha}_k(y, u_k^*)$ where $y_O = y_O^*$. Then, from Corollary 1,

$$\mathbf{BE}(\hat{Q}, \mathbf{b}, a_k) \leqslant \gamma \max_{y_O} \Big| \max_{u_k} \sum_{y_{-O}} \boldsymbol{\alpha}_k\big((y_O, y_{-O}), u_k\big) - \sum_{y_{-O}} \max_{u_k} \boldsymbol{\alpha}_k\big((y_O, y_{-O}), u_k\big)\Big|,$$

finally yielding

$$\big\|V^* - V^{\pi_k}\big\|_\infty \leqslant \frac{2\gamma^2}{(1-\gamma)^2} \max_{y_O} \Big| \max_{u_k} \sum_{y_{-O}} \boldsymbol{\alpha}_k\big((y_O, y_{-O}), u_k\big) - \sum_{y_{-O}} \max_{u_k} \boldsymbol{\alpha}_k\big((y_O, y_{-O}), u_k\big)\Big|. \qquad \square$$

References

[1] N. Meuleau, M. Hauskrecht, K. Kim, L. Peshkin, L. Kaelbling, T. Dean, C. Boutilier, Solving very large weakly coupled Markov decision processes, in: Proc. 15th AAAI Conf. Artificial Intelligence, 1998, pp. 165–172.
[2] S. Singh, D. Cohn, How to dynamically merge Markov decision processes, Advances in Neural Information Processing Systems 10 (1998) 1057–1063.
[3] R. Nair, M. Tambe, Hybrid BDI-POMDP framework for multiagent teaming, Journal of Artificial Intelligence Research 23 (2005) 367–420.
[4] P. Stone, M. Veloso, Team-partitioned, opaque-transition reinforcement learning, in: Proc. RoboCup-98, 1998, pp. 206–212.
[5] M. Ghavamzadeh, S. Mahadevan, R. Makar, Hierarchical multiagent reinforcement learning, Journal of Autonomous Agents and Multiagent Systems 13 (2) (2006) 197–229.
[6] C. Guestrin, D. Koller, R. Parr, Multiagent planning with factored MDPs, Advances in Neural Information Processing Systems 14 (2001) 1523–1530.
[7] J. Kok, P. Hoen, B. Bakker, N. Vlassis, Utile coordination: learning interdependencies among cooperative agents, in: IEEE Symp. Computational Intelligence and Games, 2005, pp. 61–68.
[8] M. Roth, R. Simmons, M. Veloso, Exploiting factored representations for decentralized execution in multiagent teams, in: Proc. 6th Int. Conf. Autonomous Agents and Multiagent Systems, 2007, pp. 469–475.
[9] M. Kearns, M. Littman, S. Singh, Graphical models for game theory, in: Proc. 17th Conf. Uncertainty in Artificial Intelligence, 2001, pp. 253–260.
[10] A. Xin Jiang, K. Leyton-Brown, N. Bhat, Action-graph games, Tech. rep. TR-2008-13, Univ. British Columbia, 2008.
[11] M. Allen, S. Zilberstein, Complexity of decentralized control: special cases, Advances in Neural Information Processing Systems 22 (2009) 19–27.
[12] C. Goldman, S. Zilberstein, Decentralized control of cooperative systems: categorization and complexity analysis, Journal of Artificial Intelligence Research 22 (2004) 143–174.
[13] P. Varakantham, J. Kwak, M. Taylor, J. Marecki, P. Scerri, M. Tambe, Exploiting coordination locales in distributed POMDPs via social model shaping, in: Proc. 19th Int. Conf. Automated Planning and Scheduling, 2009, pp. 313–320.
[14] M. Puterman, Markov Decision Processes, Discrete Stochastic Dynamic Programming, John Wiley & Sons, Inc., 1994.
[15] L. Kaelbling, M. Littman, A. Cassandra, Planning and acting in partially observable stochastic domains, Artificial Intelligence 101 (1998) 99–134.
[16] D. Bernstein, R. Givan, N. Immerman, S. Zilberstein, The complexity of decentralized control of Markov decision processes, Mathematics of Operations Research 27 (4) (2002) 819–840.
[17] C. Watkins, Learning from delayed rewards, Ph.D. thesis, King's College, Cambridge Univ., 1989.
[18] D. Bertsekas, J. Tsitsiklis, Neuro-Dynamic Programming, Athena Scientific, 1996.
[19] R. Smallwood, E. Sondik, The optimal control of partially observable Markov processes over a finite horizon, Operations Research 21 (5) (1973) 1071–1088.
[20] O. Madani, S. Hanks, A. Condon, On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems, in: Proc. 16th AAAI Conf. Artificial Intelligence, 1999, pp. 541–548.
[21] A. Cassandra, Exact and approximate algorithms for partially observable Markov decision processes, Ph.D. thesis, Dept. Computer Sciences, Brown Univ., 1998.
[22] D. Aberdeen, A (revised) survey of approximate methods for solving partially observable Markov decision processes, Tech. rep., National ICT Australia, 2003.
[23] M. Littman, A. Cassandra, L. Kaelbling, Learning policies for partially observable environments: scaling up, in: Proc. 12th Int. Conf. Machine Learning, 1995, pp. 362–370.
[24] F. Melo, M. Ribeiro, Transition entropy in partially observable Markov decision processes, in: Proc. 9th Int. Conf. Intelligent Autonomous Systems, 2006, pp. 282–289.
[25] C. Papadimitriou, J. Tsitsiklis, The complexity of Markov decision processes, Mathematics of Operations Research 12 (3) (1987) 441–450.
[26] S. Seuken, S. Zilberstein, Formal models and algorithms for decentralized decision-making under uncertainty, Journal of Autonomous Agents and Multiagent Systems 17 (2) (2008) 190–250.
[27] R. Becker, S. Zilberstein, V. Lesser, C. Goldman, Solving transition independent decentralized Markov decision processes, Journal of Artificial Intelligence Research 22 (2004) 423–455.
[28] M. Allen, Interactions in decentralized environments, Ph.D. thesis, Univ. Massachusetts, Amherst, 2009.
[29] A. Cassandra, Optimal policies for partially observable Markov decision processes, Tech. rep. CS-94-14, Dept. Computer Sciences, Brown Univ., 1994.
[30] M. Spaan, F. Melo, Interaction-driven Markov games for decentralized multiagent planning under uncertainty, in: Proc. 7th Int. Conf. Autonomous Agents and Multiagent Systems, 2008, pp. 525–532.
[31] C. Claus, C. Boutilier, The dynamics of reinforcement learning in cooperative multiagent systems, in: Proc. 15th AAAI Conf. Artificial Intelligence, 1998, pp. 746–752.
[32] M. Tan, Multi-agent reinforcement learning: independent vs. cooperative agents, in: Readings in Agents, Morgan Kaufman, 1997, pp. 487–494.
[33] D. Leslie, E. Collins, Generalised weakened fictitious play, Games and Economic Behavior 56 (2006) 285–298.
[34] R. Sutton, A. Barto, Reinforcement Learning: An Introduction, MIT Press, 1998.
[35] L. Shapley, Stochastic games, Proceedings of the National Academy of Sciences 39 (1953) 1095–1100.
[36] C. Boutilier, Planning, learning and coordination in multiagent decision processes, in: Theoretical Aspects of Rationality and Knowledge, 1996, pp. 195–210.
[37] D. Pynadath, M. Tambe, The communicative multiagent team decision problem: analyzing teamworktheories and models, Journal of Artificial Intelligence Research 16 (2002) 389–423.
[38] P. Gmytrasiewicz, P. Doshi, A framework for sequential planning in multiagent settings, Journal of Artificial Intelligence Research 24 (2005) 49–79.
[39] R. Emery-Montemerlo, G. Gordon, J. Schneider, S. Thrun, Approximate solutions for partially observable stochastic games with common payoffs, in: Proc. 3rd Int. Conf. Autonomous Agents and Multiagent Systems, 2004, pp. 136–143.
[40] M. Roth, Execution-time communication decisions for coordination of multiagent teams, Ph.D. thesis, Carnegie Mellon University, August 2007.
[41] R. Becker, S. Zilberstein, V. Lesser, C. Goldman, Transition-independent decentralized Markov decision processes, in: Proc. 2nd Int. Conf. Autonomous Agents and Multiagent Systems, 2003, pp. 41–48.
[42] R. Makar, S. Mahadevan, Hierarchical multiagent reinforcement learning, in: Proc. 5th Int. Conf. Autonomous Agents, 2001, pp. 246–253.
[43] C. Guestrin, S. Venkataraman, D. Koller, Context-specific multiagent coordination and planning with factored MDPs, in: Proc. 18th AAAI Conf. Artificial Intelligence, 2002, pp. 253–259.
[44] J. Kok, N. Vlassis, Sparse cooperative Q-learning, in: Proc. 21st Int. Conf. Machine Learning, 2004, pp. 61–68.
[45] N. Bhat, K. Leyton-Brown, Computing Nash equilibria of action-graph games, in: Proc. 20th Conf. Uncertainty in Artificial Intelligence, 2004, pp. 35–42.
[46] A. Xin Jiang, K. Leyton-Brown, A polynomial-time algorithm for action-graph games, in: Proc. 21st AAAI Conf. Artificial Intelligence, 2006, pp. 679–684.
[47] A. Xin Jiang, K. Leyton-Brown, Computing pure Nash equilibria in symmetric action-graph games, in: Proc. 22nd AAAI Conf. Artificial Intelligence, 2007, pp. 79–85.
[48] R. Becker, V. Lesser, S. Zilberstein, Decentralized Markov decision processes with event-driven interactions, in: Proc. 3rd Int. Conf. Autonomous Agents and Multiagent Systems, 2004, pp. 302–309.

[49] R. Becker, V. Lesser, S. Zilberstein, Analyzing myopic approaches for multiagent communication, in: Proc. IEEE Int. Conf. Intelligent Agent Technology, 2005, pp. 550–557.
[50] R. Becker, A. Carlin, V. Lesser, S. Zilberstein, Analyzing myopic approaches for multiagent communications, Computational Intelligence 25 (1) (2009) 31–50.
[51] S. Simic, On a global upper bound for Jensen's inequality, Journal of Mathematical Analysis and Applications 343 (2008) 414–419.