Proceedings of the Twenty-Sixth International Conference on
Automated Planning and Scheduling (ICAPS 2016)

Indefinite-Horizon Reachability in Goal-DEC-POMDPs*

Krishnendu Chatterjee
IST Austria
Klosterneuburg, Austria
kchatterjee@ist.ac.at

Martin Chmelík
IST Austria
Klosterneuburg, Austria
mchmelik@ist.ac.at

Abstract

DEC-POMDPs extend POMDPs to a multi-agent setting, where several agents operate in an uncertain environment independently to achieve a joint objective. DEC-POMDPs have been studied with finite-horizon and infinite-horizon discounted-sum objectives, and there exist solvers both for exact and approximate solutions. In this work we consider Goal-DEC-POMDPs, where given a set of target states, the objective is to ensure that the target set is reached with minimal cost. We consider the indefinite-horizon (infinite-horizon with either discounted-sum, or undiscounted-sum, where absorbing goal states have zero-cost) problem. We present a new and novel method to solve the problem that extends methods for finite-horizon DEC-POMDPs and the RTDP-Bel approach for POMDPs. We present experimental results on several examples, and show that our approach presents promising results.

1 Introduction

POMDPs and DEC-POMDPs. Partially-observable Markov decision processes (POMDPs) are standard models for problems related to probabilistic planning, where an agent tries to optimize an objective in an uncertain environment (Howard 1960; Papadimitriou and Tsitsiklis 1987; Kaelbling *et al.* 1998). There are a wide range of applications of POMDPs ranging from reinforcement learning (Kaelbling *et al.* 1996), to software verification (Cerný *et al.* 2011) to robot motion planning (Kaelbling *et al.* 1998). However, in many scenarios there is not a single agent, but a set of agents whose joint goal is to optimize an objective in an uncertain environment. Each agent has a different view of the state space of the system, and must choose a *local* policy based on her own view, such that the *joint* policy optimizes the objective function. For example, a classic scenario is when two robots are necessary to achieve a task (like moving a large box) and each of them has a specific view of the environment (Seuken and Zilberstein 2007; Bernstein *et al.* 2005). *Decentralized POMDPs* (DEC-POMDPs) provide the appropriate model for such scenarios, which are an extension of POMDPs to a multi-agent setting, where each agent has her personal view of the entire system (Seuken and Zilberstein 2007; Bernstein *et al.* 2005; Amato and Zilberstein 2009; Nair *et al.* 2003).

Indefinite-horizon objectives. In this work we consider DEC-POMDPs with indefinite-horizon objectives. We consider DEC-POMDPs with a set of target (or goal) states, and a cost function that assigns a cost to each transition. Given a discount factor $0 < \gamma \leq 1$, the cost of a path is the discounted-sum of the costs of the transitions in the path. If $\gamma = 1$, then we refer to the objective as the undiscounted-sum objective, and if $\gamma < 1$, then we refer to the objective as the discounted-sum objective. The costs are accumulated until the target is reached (i.e., once a target state is reached, from then on the costs are zero). The objective function we consider is to ensure that the target set is reached with probability 1 and the accumulated cost according to the discounted-sum (or undiscounted-sum) is minimized. The objectives we consider are the classical optimization criteria in the setting of POMDPs and DEC-POMDPs. Note that while a finite-horizon objective requires to optimize the cost for a given finite number of steps, and an infinite-horizon objective requires to optimize over the whole length of the paths, an indefinite-horizon objective requires to optimize until the target set is reached.

Restriction. Most problems related to optimizations in POMDPs are undecidable, e.g., infinite-horizon undiscounted planning (Paz 1971; Madani *et al.* 2003). There is a number of POMDP restrictions that imply decidability results, e.g., approximations of infinite-horizon discounted planning, finite-horizon undiscounted planning (Goldsmith and Mundhenk 1998). To develop a practical approach to solve POMDPs, restricted class of POMDPs are considered, where from every state there is a path to a target state, and such POMDPs are referred to as Goal-POMDPs (Bonet and Geffner 2009). Several examples of POMDPs that arise in practice can be modeled as Goal-POMDPs (Bonet and Geffner 2009; Kolobov *et al.* 2011). Hence in this work we consider DEC-POMDPs with the restriction that from every state there is a path to a target state, and refer to them as Goal-DEC-POMDPs. Note, that our restriction is weaker than the requirement that any strategy reaches the goal states with probability 1.

*The research was partly supported by Austrian Science Fund (FWF) Grant No P23499-N23, FWF NFN Grant No S11407-N23 (RiSE), ERC Start grant (279307: Graph Games), and Microsoft faculty fellows award.

Previous results. One of the most efficient and practical approach to solve Goal-POMDPs is the RTDP-Bel approach that extends the classical RTDP approach (Barto *et al.* 1995). There are several methods, exact and approximate, to solve finite-horizon and infinite-horizon DEC-POMDPs (Seuken and Zilberstein 2007; Amato *et al.* 2007) (see Related work for detailed discussion).

Our contributions. In this work we present an approach to solve Goal-DEC-POMDPs with indefinite-horizon objectives. Our approach combines and extends two classical algorithmic approaches, namely, the RTDP-Bel approach for POMDPs, and the principle of policy-iteration algorithm for MDPs (which has been considered for DEC-POMDPs as well (Banerjee *et al.* 2012)). Our approach is as follows: we consider an agent, and fix the policies of all other agents, and treat the problem as a POMDP and obtain an improved policy for the agent using a RTDP-Bel approach to solve the POMDP. We repeat this process for all agents. In principle, we fix policies for the agents, and use the RTDP-Bel approach for the policy improvements. The two challenges for this approach are: (i) how to fix policies in DEC-POMDPs and interpret the problem as POMDP; and (ii) then translate the policy from the POMDP to the DEC-POMDP. For the first issue, we compute a probability distribution over states that mimics the belief-state for the agent, even though in the multi-agent setting there is no simple notion of belief-state (Hansen *et al.* 2004). For the second issue, we merge belief-states in the POMDP that corresponds to probability distribution over states in the DEC-POMDP, and thus interpret the policy from the POMDP in the DEC-POMDP. In addition to our approach, we present several heuristics to make the approach practical. Our heuristics mainly achieve state-space reduction for the POMDP construction, and are based on (i) merging belief-states that are close in terms of probability distribution; and (ii) even if the DEC-POMDP reaches a point where no policy is returned by the RTDP-Bel approach, instead of playing a naive policy, we play based on policies that correspond to similar beliefs in the POMDP. We have implemented our approach along with the heuristics, and present experimental results on several DEC-POMDP examples from the literature that demonstrates the effectiveness of our approach. Moreover, we present additional large examples and demonstrate the scalability of our approach.

1.1 Related work

POMDPs. There are several works for discounted POMDPs (Kurniawati *et al.* 2008; Smith and Simmons 2004; Pineau *et al.* 2003), as well as for Goal-MDPs and Goal-POMDPs (Bonet and Geffner 2009; Kolobov *et al.* 2011; Chatterjee *et al.* 2015). In this work we extend the Goal-POMDP solution to Goal-DEC-POMDPs.

DEC-POMDPs. The problem of DEC-POMDPs with various objectives has been an active research area, see (Seuken and Zilberstein 2008; Oliehoek 2012; Kochenderfer 2015) for surveys. Some important works related to DEC-POMDPs are as follows: (i) exact algorithms for finite-horizon objectives (Hansen *et al.* 2004; Szer *et al.* 2005); (ii) approximate methods for finite-horizon objectives (Nair *et al.* 2003; Seuken and Zilberstein 2007); (iii) approximate methods for infinite-horizon discounted objectives (Amato *et al.* 2007; Bernstein *et al.* 2005; Szer and Charpillet 2005). The undecidability for infinite-horizon objectives was established in (Bernstein *et al.* 2002). A translation of DEC-POMDPs to continuous-state MDPs was established in (Dibangoye *et al.* 2013). The error-bounded approximations for infinite-horizon discounted-sum objectives was considered in (Dibangoye *et al.* 2014). An approach based on fixing policies in turns for finite-horizon objectives was considered in (Banerjee *et al.* 2012). While (Banerjee *et al.* 2012) does not consider indefinite-horizon objectives, we show how to extend the policy iteration along with RTDP-Bel approach to DEC-POMDPs with indefinite-horizon objectives. One of the most closely related work is (Amato and Zilberstein 2009) that also considers reaching goals in DEC-POMDPs, but our approach is very different: (Amato and Zilberstein 2009) considers an approach where policies for all the agents are constructed simultaneously, and in contrast we consider extension of policy iteration (for single agent with other policies fixed) and RTDP-Bel approach to solve the problem.

2 Definitions: Goal-POMDPs, DEC-POMDPs

A probability distribution f on a set X is a function $f : X \to [0, 1]$ such that $\sum_{x \in X} f(x) = 1$, and we denote by $\mathcal{D}(X)$ the set of all probability distributions on X. For $f \in \mathcal{D}(X)$ we denote by $\mathrm{Supp}(f) = \{x \in X \mid f(x) > 0\}$ its support.

DEC-POMDPs. A *Decentralized Partially Observable Markov Decision Process (DEC-POMDP)* is defined as a tuple $\mathsf{D} = (I, S, \{\mathcal{A}_i\}, \delta, \{\mathcal{Z}_i\}, \mathcal{O}, c, \lambda)$, where (i) I is a finite set of agents; (ii) S is a set of states; (iii) $\mathcal{A}_i$ is a finite set of actions for every agent $i \in I$; (iv) $\delta : S \times \prod_{i \in I} \mathcal{A}_i \to \mathcal{D}(S)$ is a probabilistic transition function that given the current state s and the vector of actions $\vec{a} \in \prod_{i \in I} \mathcal{A}_i$ for every agent gives the probability distribution over the successor states; (v) $\mathcal{Z}_i$ is a finite set of observations for every agent $i \in I$; (vi) $\mathcal{O} : S \times \prod_{i \in I} \mathcal{A}_i \to \mathcal{D}(\prod_{i \in I} \mathcal{Z}_i)$ gives the probability of observing $\vec{z} \in \prod_{i \in I} \mathcal{Z}_i$ given the current state and the played actions; (vi) $c : S \times \prod_{i \in I} \mathcal{A}_i \to \mathbb{R}$ is a function that given a state and the played actions assigns a cost; and (vii) $\lambda \in \mathcal{D}(S)$ is the initial state distribution. We will for simplicity use $\overline{\mathcal{A}}$ as a shortcut for $\prod_{i \in I} \mathcal{A}_i$ and similarly $\overline{\mathcal{Z}}$ for $\prod_{i \in I} \mathcal{Z}_i$. Let $\mathbb{D}$ denote the class of all DEC-POMDPs.

Subclasses. We identify two subclasses of $\mathbb{D}$ based on the type of the observation function $\mathcal{O}$.

Observation–independent DEC-POMDPs. A DEC-POMDP D is *observation–independent* (Allen and Zilberstein 2009) if the observation function $\mathcal{O}$ can be written as a set of individual observation function $\mathcal{O}_j$ for $j \in I$, where $\mathcal{O}_j : S \times \prod_{i \in I} \mathcal{A}_i \to \mathcal{D}(\mathcal{Z}_j)$. Let $\mathbb{O}$ denote this subclass.

Action–independent DEC-POMDPs. A DEC-POMDP D is *action–independent* if it is *observation–independent* and moreover the individual observation functions $\mathcal{O}_j$ for $j \in I$ do not depend on actions played by other agents, i.e., $\mathcal{O}_j : S \times \mathcal{A}_j \to \mathcal{D}(\mathcal{Z}_j)$. Let $\mathbb{A}$ denote this subclass.

In (Becker *et al.* 2003) a subclass called transition-independent DEC-POMDPs was defined, and we denote the subclass as $\mathbb{T}$. The relationship is as follows: $\mathbb{T} \subset \mathbb{A} \subset \mathbb{O} \subset \mathbb{D}$, and the inclusions are strict.

POMDPs. A *Partially Observable Markov Decision Process (POMDP)* is a special type of DEC-POMDP where the set of agents I is restricted to be a singleton.

Plays. A *play* in a DEC-POMDP is an infinite sequence $\rho = (s_0, \vec{a}_0, s_1, \vec{a}_1, s_2, \vec{a}_2, \ldots)$ of states and action sets $\vec{a}_n \in \overline{\mathcal{A}}$ such that $s_0 \in \mathrm{Supp}(\lambda)$ and for all $n \geq 0$ we have $\delta(s_n, \vec{a}_n)(s_{n+1}) > 0$. We write Ω for the set of all plays.

Strategies (or policies). A *local strategy (or a policy)* for agent $i \in I$ is a recipe to extend prefixes of plays and is a function $\sigma_i : (\mathcal{Z}_i \cdot \mathcal{A}_i)^* \cdot \mathcal{Z}_i \to \mathcal{D}(\mathcal{A}_i)$ that given a finite history of observations observed by agent i and actions played by agent i selects a probability distribution over the actions $\mathcal{A}_i$. For agent $i \in I$ we denote by σ_i^U the strategy that plays all the actions $\mathcal{A}_i$ uniformly at random[1], i.e., for all histories $\rho = (z_0, a_0, z_1, a_1, \ldots, a_{n-1}, z_n)$ and actions $a \in \mathcal{A}_i$ we have $\sigma_i^U(\rho)(a) = 1/|\mathcal{A}_i|$. Given a local strategy σ_i for every agent $i \in I$, we denote by σ_I the joint strategy $\prod_{i \in I} \sigma_i$ and σ_I^U for $\prod_{i \in I} \sigma_i^U$.

Cones and last state. For a finite prefix $w \in (S \cdot \overline{\mathcal{A}})^* \cdot S$ of a play, we denote by $\mathsf{Cone}(w)$ the set of plays with w as the prefix (i.e., the cone or cylinder of the prefix w), and denote by $\mathsf{Last}(w)$ the last state of w.

Probability and Expectation Measures. Given a joint strategy σ_I and an initial state distribution λ, the unique probability measure obtained given σ_I is denoted as $\mathbb{P}_\lambda^{\sigma_I}(\cdot)$. Let $\mathcal{K} : (S \cdot \overline{\mathcal{A}})^* \cdot S \to \mathcal{D}((\overline{\mathcal{Z}} \cdot \overline{\mathcal{A}})^* \cdot \overline{\mathcal{Z}})$ be the probability distribution that given a finite prefix w of a play gives a probability distribution over the joint observation and action sequence ρ observed by the joint strategy. We first define the measure $\mu_\lambda^{\sigma_I}(\cdot)$ on cones. For $w = s$, where $s \in S$, we have $\mu_\lambda^{\sigma_I}(\mathsf{Cone}(w)) = \lambda(s)$ and for $w' = w \cdot \vec{a} \cdot s$ we have $\mu_\lambda^{\sigma_I}(\mathsf{Cone}(w')) = \mu_\lambda^{\sigma_I}(\mathsf{Cone}(w)) \cdot \sum_{\rho \in \mathrm{Supp}(\mathcal{K}(w))} \mathcal{K}(w)(\rho) \cdot \sigma_I(\rho)(\vec{a}) \cdot \delta(\mathsf{Last}(w), \vec{a})(s)$. By Carathéodory's extension theorem, the function $\mu_\lambda^{\sigma_I}(\cdot)$ can be uniquely extended to a probability measure $\mathbb{P}_\lambda^{\sigma_I}(\cdot)$ over Borel sets of infinite plays (Billingsley 1995). We denote by $\mathbb{E}_\lambda^{\sigma_I}[\cdot]$ the expectation measure associated with strategy σ_I.

Objectives. We consider the following objectives:

(Un)Discounted Sum Objectives. The traditional optimization objective in (DEC-)POMDPs is defined as follows: given a discount factor $0 < \gamma \leq 1$ and a play $\rho = (s_0, \vec{a}_0, s_1, \vec{a}_1, s_2, \vec{a}_2, \ldots)$ the discounted *infinite-horizon* sum of the the play ρ is the value $\mathsf{Sum}_\gamma^\infty(\rho) = \sum_{n=0}^\infty \gamma^n \cdot c(s_n, \vec{a}_n)$. A discounted *finite-horizon* sum, where a horizon length m is given, only considers the prefix of length m of a play, i.e., $\mathsf{Sum}_\gamma^m(\rho) = \sum_{n=0}^m \gamma^n \cdot c(s_n, \vec{a}_n)$. The *undiscounted sum* is a special case where γ is required to be 1.

Reachability Objectives. Goal-(DEC-)POMDPs are additionally equipped with a *reachability objective*. A reachability objective is a measurable set of plays $\varphi \subseteq \Omega$ defined by a set of absorbing *goal* states $G \subseteq S$ as follows: $\mathsf{Reach}(G) = \{(s_0, \vec{a}_0, s_1, \vec{a}_1, s_2, \vec{a}_2, \ldots) \in \Omega \mid \exists n \geq 0 : s_n \in G\}$. In the framework of Goal-(DEC-)POMDPs it is additionally required that for all states $s \in G$ and actions $\vec{a} \in \overline{\mathcal{A}}$, (i) we have $\delta(s, \vec{a})(s) = 1$ (i.e., s is absorbing); and (ii) the cost function assigns $c(s, \vec{a}) = 0$; i.e., once a play reaches a goal state it stays there and stops to accumulate costs. Observe that the cost restriction on the absorbing goal states of Goal-(DEC-)POMDPs allows the indefinite-horizon objectives to be expressed as infinite-horizon objectives.

Almost-sure winning. Given a (DEC-)POMDP D with a reachability objective $\mathsf{Reach}(G)$ a joint strategy σ_I is *almost-sure winning* iff $\mathbb{P}_\lambda^{\sigma_I}(\mathsf{Reach}(G)) = 1$. We denote by $\mathsf{Almost}_\mathsf{D}(G, \lambda)$ the set of all joint almost-sure winning strategies from the initial state distribution λ.

Optimal cost under almost-sure winning. Given a Goal-(DEC-)POMDP D with a reachability objective $\mathsf{Reach}(G)$ and a cost function c we are interested in minimizing the expected discounted (or undiscounted) sum of costs before reaching the goal set G, while ensuring that the goal set is reached almost-surely. Formally, the value of an almost-sure winning joint strategy $\sigma_I \in \mathsf{Almost}_\mathsf{D}(G, \lambda)$ is the expectation $\mathsf{Val}(\sigma_I) = \mathbb{E}_\lambda^{\sigma_I}[\mathsf{Sum}_\gamma^\infty]$, and the objective is to minimize $\mathsf{Val}(\sigma_I)$ over almost-sure winning strategies.

In our work we assume that the (DEC-)POMDP do not contain states from which the goal states are no longer reachable, i.e., for every initial state distribution λ the joint strategy σ_I^U belongs to the set of almost-sure winning strategies $\mathsf{Almost}_\mathsf{D}(G, \lambda)$.

3 Approximate Algorithm for Goal-POMDPs

In this section we summarize the key concepts of the RTDP-Bel algorithm (Bonet and Geffner 2009) that is an approximate solver for Goal-POMDPs. In the following section we show how to extend RTDP-Bel for solving Goal-DEC-POMDPs, inspired by an approach from the work (Banerjee *et al.* 2012) for finite-horizon objectives.

The state of the art algorithm for solving Goal-POMDPs is RTDP-Bel (Bonet and Geffner 2009), that is an adaptation of the older RTDP algorithm (Barto *et al.* 1995) to Goal-POMDPs. It uses the well-known approach of turning a POMDP into a completely observable continuous MDP over belief states (Åström 1965; Sondik 1978). Formally, a *belief state* is a probability distribution $b \in \mathcal{D}(S)$, that represents the information the agent has about the current state.

Belief state updates. Given the current belief state b, the action a played by the agent, and the observation z observed, it is straightforward to compute the newly reached information state b' (Cassandra *et al.* 1994) using the transition and observation probabilities. We write $b' = \mathsf{Update}(b, a, z)$ for the belief state update.

Belief state discretization. In order to bound the size of the MDP, the algorithm uses a discretization method Disc that maps belief states to *qbeliefs*. A *qbelief* is a function $q : S \to \mathbb{N}$. Given a belief state b the corresponding discretized qbelief $q = \mathsf{Disc}(b)$ is defined as follows: for every state $s \in S$ the value of $q(s)$ is $\mathrm{ceil}(D \cdot b(s))$, where D is a positive

[1] For simplicity and w.l.o.g., we consider that all actions in $\mathcal{A}_i$ are available in all states.

integer parameter of the method. Note, that the discretization method preserves the support of the belief states.

Algorithm. The algorithm runs simulations of the POMDP as follows. Every simulation starts in the initial belief state λ, and terminates when a goal state or an upper bound on the length of the simulations is reached. The information learned from the simulations is stored in a hashtable that maps qbeliefs to an estimation of the expected cost until a goal state is reached, i.e., the value of the qbelief.

During the simulation the algorithm maintains the current belief state b. In every step of the simulation for every action $a \in \mathcal{A}$ and observation $z \in \mathcal{Z}$ the new belief state $b' = \mathsf{Update}(b, a, z)$ is computed that is discretized into a qbelief $q' = \mathsf{Disc}(b')$. The next action to be played by the strategy is one that minimizes the estimations of the expected costs stored in the hashtable for the qbeliefs q'. The data from the simulation are used to update the hashtable, and to improve the strategy for the following simulations.

There are theoretical downsides of using the discretization method: the convergence is not guaranteed, and the values of the learned strategy may oscillate. Additionally, the learned expected cost is not guaranteed to be an upper bound on the optimal cost. However, the learned strategy can be evaluated (by fixing the strategy and obtaining the value in the Markov chain) and the obtained value is guaranteed to be an upper bound on the optimal cost. Despite the theoretical disadvantages, in practice the RTDP-Bel method works very well. In the experimental sections of (Bonet and Geffner 2009; Chatterjee *et al.* 2015) it is shown that RTDP-Bel scales to examples that could not be solved exactly before. Finally, on smaller examples, the computed values match the values obtained from an exact solver. Hence in practice RTDP-Bel algorithm is the state-of-the-art solver for Goal-POMDPs.

4 Approximate Algorithm for Goal-DEC-POMDPs

In this part we present our new approach that extends the RTDP-Bel algorithm to provide joint strategies for DEC-POMDPs. We build on the approach presented in (Banerjee *et al.* 2012) for finite-horizon objectives. Informally, we fix policies of all but one agent, obtain a POMDP, and use RTDP-Bel for the policy improvement in a POMDP.

Algorithm overview. Intuitively, we initialize every agent $i \in I$ with an initial strategy σ_i^0 and fix all the initial strategies of the agents in the DEC-POMDP except for one. As the strategies for all but one agent, say i, are fixed, the resulting model is a POMDP. By running RTDP-Bel on the POMDP we obtain a new strategy σ_i^1 for agent i. In the next round we choose another agent j and fix all the current strategies of the remaining agents, construct a POMDP, and improve the current strategy of agent j. We iteratively cycle through agents and improve their strategies until the desired value of the joint strategy is reached. For simplicity of presentation and without loss of generality we will consider only two agent DEC-POMDPs in the following text (see Remark 1).

Remark 1 *The case for more agents is as follows. In case of more agents, it is intuitively possible to merge the remaining agents into a single agent in every iteration and instead of a set of strategies consider the joint strategy of the merged agents. The presented methods can be easily extended to handle joint strategies.*

Main challenges. There are two key issues relevant to our algorithmic approach. First, the method that fixes strategies for agents in a DEC-POMDP and obtains a POMDP as a result. In the case of two agents i and j, the belief state update of the agent i whose strategy is fixed depends on the action played by agent j. Thus unless all actions are fixed, belief update interpretation is difficult and we take a cooperative approach since the agents have a joint objective. Second, we obtain a strategy in the reduced POMDP, and the challenge is to interpret the strategy of the POMDP in the DEC-POMDP. Below we present the solution for both the issues.

Fixing a strategy in a DEC-POMDP. Given a Goal-DEC-POMDP $\mathsf{D} = (\{i,j\}, S, (\mathcal{A}_i, \mathcal{A}_j), \delta, (\mathcal{Z}_i, \mathcal{Z}_j), \mathcal{O}, c, \lambda)$, let σ_i be a strategy for agent i that is for simplicity of the type $\sigma_i : \mathcal{D}(S) \to \mathcal{D}(\mathcal{A}_i)$, i.e., given a belief state gives a probability distribution over the actions to be played next. For a DEC-POMDP D with a set of goal states G, we denote by $\mathsf{P} = \mathsf{D}{\restriction}\sigma_i$ the Goal-POMDP P that is obtained from DEC-POMDP D by fixing strategy σ_i of agent i. The POMDP $\mathsf{P} = (\{j\}, S', \{\mathcal{A}_j\}, \delta', \{\mathcal{Z}_j\}, \mathcal{O}', c', \lambda')$ is defined as:

- The states are $S' = \{(s,b) \mid s \in S, b \in \mathcal{D}(S)\}$, i.e., a state of D and a probability distribution that mimics the belief state of agent i.
- The transition probability $\delta'((s,b), a_j)((s',b'))$ for $a_j \in \mathcal{A}_j$ is defined as the product of the follows. Let $P(b, \vec{a}) : \mathcal{D}(S) \times \overline{\mathcal{A}} \to \mathcal{D}(\mathcal{D}(S))$ denote the probability distribution of updating to the belief state b', given the current belief state is b and actions $\vec{a}$ are played. The transition probability is given as follows:

$$\delta'((s,b), a_j)((s',b')) = \sum_{a_i \in \mathcal{A}_i} \sigma_i(b)(a_i) \cdot \delta(s,(a_i,a_j))(s') \cdot P(b,(a_i,a_j))(b')$$

  i.e., for every action a_i of agent i the components are the probability that agent i plays action a_i, the transition probability of moving to state s', and the probability of updating to belief state b'. Note that due to the presence of multiple agents, the notion of a belief state update is not well defined, as it depends on the action of agent j. The intuitive explanation of the distribution stored in the state, is the belief state of agent i if agent j cooperates and plays the expected action.
- The observation function $\mathcal{O}'$ is a projection of the function $\mathcal{O}$, conditioned on strategy σ_i. Formally:

$$\mathcal{O}'((s,b), a_j)(z_j) = \sum_{a_i \in \mathcal{A}_i} \sigma_i(b)(a_i) \cdot \sum_{z_i \in \mathcal{Z}_i} \mathcal{O}(s, a_i, a_j)((z_i, z_j))$$

- The cost c' is weighted over the actions played by strategy σ_i, i.e., the function is:

$$c((s,b), a_j) = \sum_{a_i \in \mathcal{A}_i} \sigma_i(b)(a_i) \cdot c(s, (a_i, a_j))$$

- The initial belief state for agent i is λ, therefore the distribution is defined $\lambda'((s,b)) = \lambda(s)$ for $b = \lambda$ and 0 otherwise.

The goal states of the POMDP $\mathsf{P} = \mathsf{D} \restriction \sigma_i$ are all states (s,b), where s is an goal state of DEC-POMDP D. The output POMDP P is then used as an input for the RTDP-Bel algorithm. The tool outputs a deterministic strategy σ_j for the POMDP, i.e., given a belief state the strategy returns a single action to be played next. Note, that the number of states in the POMDP does not increase with the number of iterations. In the next part we show how to interpret a strategy σ_j of POMDP P in the DEC-POMDP D.

Interpreting the POMDP strategy. Let the current two strategies for agents i and j be σ_i^k and σ_j^ℓ, respectively. We fix a strategy σ_i^k in the DEC-POMDP D and obtain a strategy σ_j for the POMDP $\mathsf{P} = \mathsf{D} \restriction \sigma_i^k$. In this part we present how to interpret the strategy σ_j in the context of the DEC-POMDP and obtain an improved strategy $\sigma_j^{\ell+1}$ for agent j. The strategy σ_j is of type $\mathcal{D}(S \times \mathcal{D}(S)) \to \mathcal{D}(\mathcal{A}_j)$, whereas the strategy $\sigma_j^{\ell+1}$ needs to be of type $\mathcal{D}(S) \to \mathcal{D}(\mathcal{A}_j)$.

We define a transformation function Reduce that reduces belief states b' from the POMDP to belief states b of the DEC-POMDP as follows: Given a belief $b' \in \mathcal{D}(S \times \mathcal{D}(S))$ in P, the reduced belief $b = \mathsf{Reduce}(b')$, where $b \in \mathcal{D}(S)$ in D, for a state $s \in S$ is defined as follows: $b(s) = \sum_{\bar{b} \in \mathcal{D}(S)} b'(s, \bar{b})$. In other words, given a state s we sum the probabilities of all states, where the first component is s, i.e., the states differ only in the belief state of agent i.

Given a belief state b in the DEC-POMDP D, the new strategy $\sigma_j^{\ell+1}(b)$ is defined as follows:

1. Compute all belief states b' such that $b = \mathsf{Reduce}(b')$, where the corresponding qbeliefs $q = \mathsf{Disc}(b')$ are stored in the hashtable. If there are no such beliefs play all the available actions uniformly at random.
2. For every belief state b' compute the discretized qbelief $q = \mathsf{Disc}(b')$ and utilize the hashtable stored by RTDP-Bel that for any given qbelief stores its value, i.e., the expected sum of costs before the goal state is reached, and selects an optimal belief b'.
3. The strategy $\sigma_j^{\ell+1}(b)$ is defined to play the action according to the optimal value over the considered qbeliefs stored in the hashtable with probability 99% (according to the optimal belief b'). With the remaining probability 1% plays all the other actions from $\mathcal{A}_j$ uniformly at random. Note that this ensures that all actions are played with positive probability, and as a consequence we can show (in item 1 of Theorem 1) that the POMDP we obtain is a Goal-POMDP.

Our approach is summarized as Algorithm 1. In the first step all the agents are initialized with a strategy that always plays all available actions uniformly at random. The algorithm is parametrized with an integer k, that specifies how many times should a strategy of an individual agent be improved.

Theorem 1 *The following assertions hold:*

Algorithm 1 GOAL-DEC-POMDPS

Input: Goal-DEC-POMDP D, Integer k
Output: A joint strategy σ_I

▷ Initialize the agents with initial strategies
for $i \in I$ **do**
  $\sigma_i^0 \leftarrow \sigma_i^U$
▷ Iteratively improve the strategies
for $0 \leq u < k$ **do**
  for $j \in I$ **do**
    POMDP $\mathsf{P} \leftarrow \mathsf{D} \restriction (\prod_{i \in I \setminus \{j\}} \sigma_i^u)$
    $\sigma_j \leftarrow$ RTDP-Bel(P)
    $\sigma_j^{u+1} \leftarrow$ InterpretOnDEC-POMDP(σ_j)
return $\sigma_I \leftarrow \prod_{i \in I} \sigma_i^k$

1. *During the computation of Algorithm 1, the constructed POMDPs are Goal-POMDPs.*
2. *The value of the computed joint strategy σ_I is an upper bound on value of the optimal-cost joint strategy.*

Proof. For the first item, note that for every agent $i \in I$ and every $0 \leq u \leq k$ the strategy σ_i^u satisfies that for every belief state b we have $\mathrm{Supp}(\sigma_i^u(b)) = \mathcal{A}_i$. Therefore, any path in the DEC-POMDP can be mapped to a path in the POMDP that is obtained by fixing the strategies of the agents. Since we consider Goal-DEC-POMDP, it follows that in the POMDP there is a path from every state to a target state, and hence we obtain a Goal-POMDP. For the second item, note that the value learned from RTDP-Bel is not guaranteed to be an upper bound on the optimal value, however, we can evaluate the joint strategy by constructing the corresponding Markov chain. The expected cost to reach the goal state in the chain is an upper bound on the optimal value. □

Remark 2 (Discussion) *The crucial property that affects the performance of Algorithm 1 is the* estimation of the current belief *of agents given the currently fixed strategy. Consider that agent i is improving its strategy. The knowledge of the belief of agent j allows agent i to estimate what action agent j is going to play, and cooperate with agent j. Thus the accurate belief estimation allows for efficient cooperation. For the general class of DEC-POMDPs the estimation can be imprecise due to two reasons: (1) the observation function $\mathcal{O}$, as the probability distribution over observations for agent i depends on the observation for agent j, which is not known; and (2) the randomized strategy, the observation distribution for agent i depends on the action played by agent j which in case of a randomized strategy is not known.*

However, in the special case of observation–independent *DEC-POMDPs there is no imprecision due to observation function (due to independence), and the only imprecision in this setting is due to a potentially randomized strategy. Finally, in case of* action–independent *DEC-POMDPs even though the imprecision due to strategy can appear as well, but unlike in the previous case the error in the estimation is*

only local *(i.e., in each step), and the error does not propagate in the further steps for estimation. Thus for the subclasses $\mathbb{O}$ and $\mathbb{A}$ our algorithm is likely to perform well.*

Remark 3 (Comparison with JESP Algorithms.) *The presented algorithm is related to the JESP algorithm (Joint Equilibrium-based Search for Policies) (Nair* et al. *2003). The main differences are as follows: (i) JESP algorithms are designed for finite-horizon objectives and crucially rely on the knowledge of the horizon length, whereas our algorithm works for indefinite-horizon objectives, and (ii) for finite-horizon objectives the strategy search space is finite and well-understood as compared to indefinite-horizon objectives (Szer and Charpillet 2006).*

Remark 4 (Technical remark) *Note that in our approach we store state and belief state pairs, instead of states and qbelief pairs. We remark that the simpler approach to replace belief states with qbeliefs in the states of the POMDPs, that are obtained by fixing the strategies of the agents, might not yield satisfactory results. Intuitively, the stored belief state may evolve for a sequence of steps, even when the corresponding qbelief remains unchanged. The qbelief is changed only after a certain number of steps. This behavior is not captured if qbeliefs instead of the belief states are considered.*

5 Heuristics

In this section we present a number of possible heuristics to make our approach practical. Intuitively, any joint strategy σ_I, that satisfies that the individual strategies of the agents always play all the available actions with positive probability, is almost-sure winning for the reachability objective. Therefore, we focus on heuristics that reduce the size of the state space of the POMDPs constructed in Algorithm 1, to speed up the construction of the POMDP and run RTDP-Bel on smaller instances.

Merge states with close belief states. Given a DEC-POMDP D and a strategy σ_i for agent $i \in I$ the constructed POMDP $\mathsf{P} = \mathsf{D} \restriction \sigma_i$ has a continuous set of states due to the fact that the states s are elements in $S \times \mathcal{D}(S)$. We exploit the assumptions, that if two probability distributions are similar, then it suffices to keep one of them as representative and reduce the size of the POMDP. Given a threshold $\delta > 0$ we say that two belief states $b, b' \in \mathcal{D}(S)$ are δ-close if the following two conditions hold:

1. $\mathrm{Supp}(b) = \mathrm{Supp}(b')$; and
2. for all $s \in S : |b(s) - b'(s)| \leq \delta$.

In the constructed POMDP, whenever a new state (s, b) is to be added, it is checked that whether there exists a state (s, b'), where b and b' are δ-close. If it is the case, then transitions to (s, b) are redirected to (s, b'), otherwise a new state (s, b) is added. This heuristic ensures that the constructed POMDP has a finite state set, and can be used to adjust the size of the POMDP.

Relax the interpretation of the strategy. As defined in Section 4 (item 1 of Interpreting the POMDP strategy), whenever no matching qbelief is found in the hashtable from the RTDP-Bel computation, a default naive option of playing all the available actions uniformly at random is chosen. In this heuristic we consider instead other qbeliefs stored in the hashtable, that are close to the qbelief that is missing (in a similar sense as in the previous heuristic). The resulting distribution over the actions will still play all the available actions with positive probability, but the resulting distribution is a weighted sum based on the closeness of the other qbeliefs and their probability distributions over played actions.

Initial strategies of the agents. It is possible to initialize individual agents with different initial strategies, that can affect the performance of the computation. Without prior knowledge of the model, the strategy that plays all the available actions uniformly at random performs well, as it explores the largest state space of the DEC-POMDP with high probability.

More aggressive reductions of the state space. For larger examples one can significantly reduce the state space of the POMDP by deploying more aggressive reductions. One can view the belief state component of the state as an information that the strategy uses to make decisions. There are various possibilities to reduce the size, we give a few examples:

- Remove states s from the belief state b (assign $b(s) = 0$), when their probability $b(s)$ is a below a certain threshold and distribute the probability mass to the other states.
- For a given integer n keep only the n highest probable states in the belief state and remove all the others.

6 Implementation and Experimental Results

We have implemented our approach presented in Section 4 together with the heuristics from Section 5 in Java. We have modified the existing implementation of RTDP-Bel in C++ to export the strategy together with the content of the hashtable storing the values and actions for individual qbeliefs to a file.

Key difference to existing tools. Several benchmarks presented in the literature are naturally expressed as Goal-DEC-POMDPs. However, in the literature the infinite-horizon discounted-sum objective is considered, rather than indefinite-horizon (discounted-sum or undiscounted-sum) objectives. Thus our setup is different from most existing tools. Since in the literature the objective is infinite-horizon discounted-sum, for the benchmarks the objectives have been expressed as follows: whenever a target state of the DEC-POMDP is reached a negative cost (reward) is acquired and the DEC-POMDP is restarted back to the initial distribution. Note that in this approach undiscounted-sum objectives are not allowed, and the discounted-sum costs are accumulated even after the goal has been reached. Our framework allows to explicitly specify goal states in the input file and can handle undiscounted-sum (as well as discounted-sum) indefinite horizon problems, and provide a natural and convenient framework to model the problems.

Experimental setup. For every example we run a number of iterations of strategy improvements. The constructed strategies are then evaluated by constructing a Markov chain and

the value of the strategies is computed by running simulations within the RTDP-Bel algorithm. In iteration 0 the strategies are initialized to the naive setting, where all available actions are played uniformly at random. The time needed to evaluate the strategies after the last iteration finishes is very small and is not included in the running times. We have experimented with a number of examples from the literature on an Intel Core i7 (2.5 GHz) CPU equipped with 16GB of RAM. We have extended the `.dpomdp` file format with the possibility to specify goal states. For every POMDP constructed we run 20k simulations with RTDP-Bel, to construct a strategy. We use the built-in standard cut-off length 250 for the length of simulations. The membership of individual benchmarks in the DEC-POMDP subclasses is summarized in Table 1.

Benchmarks	DEC-POMDP class			
	$\mathbb{T}$	$\mathbb{A}$	$\mathbb{O}$	$\mathbb{D}$
Meet in the grid	×	✓	✓	✓
Decentralized Tiger	×	×	✓	✓
Box-pushing	×	✓	✓	✓
Hallway 1–3	×	×	✓	✓
RockSample 1–3	×	×	✓	✓

Table 1: Classification of the DEC-POMDP benchmarks.

Meeting in a grid problem. The problem is based on the Meeting in the Grid problem of (Bernstein *et al.* 2005) and used later in (Amato and Zilberstein 2009; Dibangoye *et al.* 2014). We require two agents that move in a 2x2 grid to meet at the same grid position. The state where the two agents a share a single grid position is a goal state. Initially the agents are placed uniformly at random among all the positions in the grid, where they do not share the same position. We assign a cost of 1 to every move of the agents until they meet in the grid. The objective is to minimize the undiscounted-sum of costs, i.e.,the expected number of steps the agents need to meet at a grid position (in other words, this is the stochastic shortest path problem). In the Table 2 we report the obtained values for individual iterations, where iteration 0 corresponds to the naive joint strategy, where all the agents play all available action uniformly at random. The values of the strategies after the 3rd iteration do not change. The results are presented in Table 2.

Decentralized Tiger problem. The problem is an extension of the well-known tiger problem POMDP (Kaelbling *et al.* 1998) to the multi-agent setting and is used extensively in the literature (Dibangoye *et al.* 2013; Amato and Zilberstein 2009). The problem models two agents that are standing in a hallway with two doors. Behind one of the doors is a tiger, behind the other a treasure. Therefore there are two states: the tiger is behind the left door or behind the right door. Both agents have three actions at their disposal: open the left door, open the right door, and listen. They cannot observe the action of the other agent. Whenever both agents open jointly a door with the treasure the model reaches a goal state. Otherwise the model is restarted (tiger is placed again randomly). Every move before reaching the goal state has cost 1, we consider the undiscounted-sum objective. The results are presented in Table 3. Unlike in the previous example, one iteration was sufficient to obtain strategies that do not improve with more iterations (in this example the optimal value is 2). The results are presented in Table 3.

Meeting in the Grid problem $\lvert S\rvert = 16, \lvert\mathcal{A}\rvert = 3, \lvert\mathcal{Z}\rvert = 2, \lvert I\rvert = 2$		
Num of iter.	**Val.**	**Tot. time**
0	10.28	-
1	6.5	4.1s
2	6.5	8.2s
3–5	4.87	9.64s

Table 2: Results for the *Meeting in the Grid problem*

Decentralized Tiger problem $\lvert S\rvert = 3, \lvert\mathcal{A}\rvert = 3, \lvert\mathcal{Z}\rvert = 2, \lvert I\rvert = 2$		
Num of iter.	**Val.**	**Tot. time**
0	17.37	-
1	2.05	0.37s
2-4	2.04	0.58s

Table 3: Results for the *Decentralized Tiger problem*

Box-pushing problem. The cooperative box-pushing problem is a well-known robotics problem introduced by (Kube and Zhang 1997). Two agents have to cooperate to move a big object (the box) that they could not move on their own. Even though the robots cannot communicate with each other, they have to achieve a certain degree of coordination to move the box at all. The problem was considered as a DEC-POMDP in (Seuken and Zilberstein 2007). We define the goal states as the situation when some of the boxes is pushed in its goal position, and we assign only positive costs (instead of positive and negative costs used in the discounted-sum setting). See Table 4 for results.

Box Pushing problem $\lvert S\rvert = 100, \lvert\mathcal{A}\rvert = 4, \lvert\mathcal{Z}\rvert = 5, \lvert I\rvert = 2$		
Num of iter.	**Val.**	**Tot. time**
0	556.68	-
1	414.82	73.4s
2-4	252.52	136.3s

Table 4: Results for the *Box Pushing problem*

Rover example. In our experimental result we compare with the examples studied in (Amato and Zilberstein 2009). Along with the examples we presented, (Amato and Zilberstein 2009) also considers another (Mars Rover) example. For this example our algorithm performs poorly, mainly because the constructed POMDP is quite large. After two iterations we could improve the value of the naive strategy (that plays all actions uniformly) by 5% and it took around 40 minutes.

Large examples. We present two new large *observation–independent* DEC-POMDP domains with number of states up to 1600.

Hallway. The problem is inspired by Hallway problems (Littman *et al.* 1995). We assume there are agents placed into a maze with several places of interest. The objective is to visit all places of interest in any order by some agent. The agents are equipped with sensors that enable it to detect walls and other agents immediately adjacent to its current location. Every agent has four noisy actions that represent the movement in the four compass directions. We consider three different sizes of the Hallway problem. The results are presented in Table 5. The size of the constructed POMDPs did not exceed 2000 states.

Hallway 1 $\|S\| = 101, \|\mathcal{A}\| = 4, \|\mathcal{Z}\| = 17, \|I\| = 2$		
Num of iter.	**Val.**	**Tot. time**
0	16.4	-
1–3	3.04	9.78s

Hallway 2 $\|S\| = 576, \|\mathcal{A}\| = 4, \|\mathcal{Z}\| = 17, \|I\| = 2$		
Num of iter.	**Val.**	**Tot. time**
0	78.34	-
1	6.03	452.97s
2	4.53	563.34s
3–5	4.02	702.64s

Hallway 3 $\|S\| = 1569, \|\mathcal{A}\| = 4, \|\mathcal{Z}\| = 17, \|I\| = 2$		
Num of iter.	**Val.**	**Tot. time**
0	98.22	-
1	56.2	2513.88s
2–4	5.48	4717.81s

Table 5: Results for the *Hallway problems*

RockSample. The problem is inspired by RockSample POMDPs (Smith and Simmons 2004). As in the case of Hallway problems, we assume there are agents placed into a maze with places of interest. In some of the places randomly mining sites are placed. In order to mine a site all the agents must be at the same time at the mining site. Whether there is a mining site on a place of interest is not known to the agents beforehand. The agents can detect whether there are walls or other agents adjacent to their current location. The movement is deterministic in the four compass directions. We consider three different sizes of the RockSample problem. The results are presented in Table 6.

6.1 Comparison with other DEC-POMDP solvers.

The other existing related tools for DEC-POMDPs are the *goal Directed* algorithm introduced in (Amato and Zilberstein 2009) and the state-of-the art infinite-horizon discounted-sum DEC-POMDP solver introduced in (Dibangoye *et al.* 2013). These implementations are not (publicly) available for comparison. We compare our results with the results reported from (Dibangoye *et al.* 2013; Amato and Zilberstein 2009). Note that since we have indefinite-horizon

RockSample 1 $\|S\| = 102, \|\mathcal{A}\| = 4, \|\mathcal{Z}\| = 15, \|I\| = 2$		
Num of iter.	**Val.**	**Tot. time**
0	10.89	-
1	1.99	2.96s
2–4	1.99	5.19s

RockSample 2 $\|S\| = 402, \|\mathcal{A}\| = 4, \|\mathcal{Z}\| = 15, \|I\| = 2$		
Num of iter.	**Val.**	**Tot. time**
0	17.72	-
1	11.18	90.41s
2	11.09	160.34s
3–5	3.42	222.46s

RockSample 3 $\|S\| = 1602, \|\mathcal{A}\| = 4, \|\mathcal{Z}\| = 15, \|I\| = 2$		
Num of iter.	**Val.**	**Tot. time**
0	18.51	-
1	18.51	305.8s
2	17.15	802.53s
3	17.15	1145.05s
4–6	12.7	1599.66s

Table 6: Results for the *RockSample problems*

undiscounted-sum objectives, and the cost assignments are different, the computed values are different. We have also modified DEC-POMDPs by adding new goal states. Nevertheless, the remaining states, transitions, and observations are similar. Also the results were computed on different but similar platforms. Hence the time comparison maybe approximate at best. In the table, we report the runtime until the iteration after which the values do not improve.

Algorithm	**Dec. Tiger**		**Meet in the Grid**		**Box pushing**	
	Val.	Time (s)	Val.	Time (s)	Val.	Time (s)
BFS	-14.1	12007	4.2	17	-2	1696
DEC-PBI	-52.6	102	3.6	2227	9.4	4094
NLP	-1.1	6174	5.7	117	54.23	1824
Goal Directed	5.0	75	5.6	4	149.9	199
FB-HSVI	13.5	6	-	-	199.4	15.2
Our algorithm		0.37		9.64		136.3

Table 7: Time comparison of the existing algorithms

7 Conclusions

In this work we presented a new approach, based on RTDP-Bel for Goal-POMDPs and the principle of policy iteration algorithm, for Goal-DEC-POMDPs. Along with our approach we presented a number of heuristics. One bottleneck of our approach is that the POMDP constructed can be large, and an important direction of future work would be to explore methods for state-space reduction of the POMDP. In DEC-POMDP an important concern is to obtain small-size policies, which was considered in (Amato and Zilberstein 2009). Whether our approach can be extended to obtain small policies is another direction of future work.

References

M. Allen and S. Zilberstein. Complexity of decentralized control: Special cases. In *Advances in Neural Information Processing Systems*, pages 19–27, 2009.

C. Amato and S. Zilberstein. Achieving goals in decentralized POMDPs. In *AAMAS*, 2009.

C. Amato, D. S. Bernstein, and S. Zilberstein. Optimizing Memory-Bounded Controllers for Decentralized POMDPs. In *UAI*, pages 1–8, 2007.

K. J. Åström. Optimal control of Markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10(1):174–205, 1965.

B. Banerjee, J. Lyle, L. Kraemer, and R. Yellamraju. Sample Bounded Distributed Reinforcement Learning for Decentralized POMDPs. In *AAAI*, 2012.

A. G. Barto, S. J. Bradtke, and S. P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1):81–138, 1995.

R. Becker, S. Zilberstein, V. Lesser, and C.V. Goldman. Transition-independent decentralized markov decision processes. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 41–48. ACM, 2003.

D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The Complexity of Decentralized Control of Markov Decision Processes. *Math. Oper. Res.*, 27(4):819–840, 2002.

D. S. Bernstein, E. A. Hansen, and S. Zilberstein. Bounded policy iteration for decentralized POMDPs. In *IJCAI*, pages 52–57, 2005.

P. Billingsley, editor. *Probability and Measure*. Wiley-Interscience, 1995.

B. Bonet and H. Geffner. Solving POMDPs: RTDP-Bel vs. Point-based Algorithms. In *IJCAI*, pages 1641–1646, 2009.

A. R. Cassandra, L. P. Kaelbling, and M. L. Littman. Acting optimally in partially observable stochastic domains. In *AAAI*, volume 94, pages 1023–1028, 1994.

P. Cerný, K. Chatterjee, T. A. Henzinger, A. Radhakrishna, and R. Singh. Quantitative synthesis for concurrent programs. In *Proc. of CAV*, LNCS 6806, pages 243–259, 2011.

K. Chatterjee, M. Chmelik, R. Gupta, and A. Kanodia. Optimal Cost Almost-sure Reachability in POMDPs. In *AAAI*, 2015.

J. S. Dibangoye, C. Amato, O. Buffet, and F. Charpillet. Optimally Solving Dec-POMDPs as Continuous-State MDPs. In *IJCAI*, 2013.

J. S. Dibangoye, O. Buffet, and F. Charpillet. Error-Bounded Approximations for Infinite-Horizon Discounted Decentralized POMDPs. In *ECML/PKDD*, pages 338–353, 2014.

J. Goldsmith and M. Mundhenk. Complexity Issues in Markov Decision Processes. In *Conference on Computational Complexity*, pages 272–280, 1998.

E.A. Hansen, D. S. Bernstein, and S. Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI*, volume 4, pages 709–715, 2004.

H. Howard. *Dynamic Programming and Markov Processes*. MIT Press, 1960.

L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *JAIR*, 4:237–285, 1996.

L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1):99–134, 1998.

Mykel J. Kochenderfer. *Decision Making Under Uncertainty: Theory and Application*. The MIT Press, 2015.

A. Kolobov, Mausam, D.S. Weld, and H. Geffner. Heuristic search for generalized stochastic shortest path MDPs. In *ICAPS*, 2011.

R. C. Kube and H. Zhang. Task modelling in collective robotics. In *Robot colonies*, pages 53–72. Springer, 1997.

H. Kurniawati, D. Hsu, and W.S. Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *RSS*, pages 65–72, 2008.

M. L. Littman, A. R. Cassandra, and L. P. Kaelbling. Learning policies for partially observable environments: Scaling up. In *ICML*. Citeseer, 1995.

O. Madani, S. Hanks, and A. Condon. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artif. Intell.*, 147(1-2):5–34, 2003.

R. Nair, M. Tambe, M. Yokoo, D. Pynadath, and S. Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *IJCAI*, 2003.

Frans A Oliehoek. Decentralized pomdps. In *Reinforcement Learning*, pages 471–503. Springer, 2012.

C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12:441–450, 1987.

A. Paz. *Introduction to probabilistic automata (Computer science and applied mathematics)*. Academic Press, 1971.

J. Pineau, G. Gordon, S. Thrun, et al. Point-based value iteration: An anytime algorithm for POMDPs. In *IJCAI*, volume 3, pages 1025–1032, 2003.

S. Seuken and S. Zilberstein. Improved Memory-Bounded Dynamic Programming for Decentralized POMDPs. In *UAI*, pages 344–351, 2007.

S. Seuken and S. Zilberstein. Formal models and algorithms for decentralized decision making under uncertainty. *AAMAS*, 17(2):190–250, 2008.

T. Smith and R. Simmons. Heuristic search value iteration for POMDPs. In *UAI*, pages 520–527. AUAI Press, 2004.

E. J. Sondik. The optimal control of partially observable Markov processes over the infinite horizon: Discounted costs. *Operations Research*, 26(2):282–304, 1978.

D. Szer and F. Charpillet. An optimal best-first search algorithm for solving infinite horizon DEC-POMDPs. In *ECML*, pages 389–399. Springer, 2005.

D. Szer and F. Charpillet. Point-based dynamic programming for dec-pomdps. In *AAAI*, 2006.

D. Szer, F. Charpillet, and S. Zilberstein. MAA*: A Heuristic Search Algorithm for Solving Decentralized POMDPs. In *UAI*, pages 576–590, 2005.