Policy Iteration Algorithms for DEC-POMDPs with Discounted Rewards

Jilles S. Dibangoye
Laval University
Québec, Qc., Canada
gdibango@ift.ulaval.ca

Brahim Chaib-draa
Laval University
Québec, Qc., Canada
chaib@ift.ulaval.ca

Abdel-Illah Mouaddib
University of Caen
Caen, France
mouaddib@info.unicaen.fr

ABSTRACT

Over the past seven years, researchers have been trying to find algorithms for the decentralized control of multiple agent under uncertainty. Unfortunately, most of the standard methods are unable to scale to real-world-size domains. In this paper, we come up with promising new theoretical insights to build scalable algorithms with provable error bounds. In the light of the new theoretical insights, this research revisits the policy iteration algorithm for the decentralized partially observable Markov decision process (DEC-POMDP). We derive and analyze the first point-based policy iteration algorithms with provable error bounds. Our experimental results show that we are able to successfully solve all tested DEC-POMDP benchmarks: outperforming standard algorithms, both in solution time and policy quality.

1. INTRODUCTION

In recent years, there has been increasing interest in finding scalable algorithms for solving multiple agent systems where agents cooperate to optimize a joint reward function, while having different individual observations. To formalize and solve such problems, [3] suggest a model that enables a set of n agents to cooperate in order to control a partially observable Markov decision process. This framework can model environments under three constraints: *uncertainty, partial observability* and *decentralization*: uncertainty relies on the fact that the agents are imperfectly informed about action effects during the simulation; partial observability means that agents are imperfectly informed about the state of the process during the execution; and decentralization signifies that the agents are differently imperfectly informed during the execution. An environment that concurrently involves these three constraints is known as a decentralized partially observable Markov decision process (DEC-POMDP). Unfortunately, finding either optimal or even ϵ-approximate solutions of such problems has been shown to be particularly hard [12].

While some important progress has been made for solving finite horizon DEC-POMDPs, we still lack efficient algorithms with provable error bounds for the infinite horizon case. Indeed, the unique ϵ-optimal algorithm for the infinite horizon case runs quickly out of memory [2], as do optimal algorithms for the finite horizon case. This is mainly because they require the exhaustive enumeration of all possible joint policies at each time step, *i.e.,* the exhaustive backup. Unfortunately, the resulting set of joint policies requires an exponential space with respect to the number of joint observations and the number of agents. As a result, many attempts to solve infinite horizon DEC-POMDPs rely on memory-bounded algorithms [1, 4, 16]. These locally optimal algorithms use a fixed amount of memory, *i.e.,* the size of the solution is fixed prior to the execution of the algorithm. Even though the size of the solution is bounded, memory-bounded algorithms still suffer from the proved complexity of the problem. Moreover, choosing the right size for the solution is not obvious and dynamically adjusting it may raise non-negligible computation costs. Furthermore, though these algorithms tackle the space complexity efficiently, the time complexity remains too high and limits their ability to scale to medium solution sizes. Even more importantly, they fail to provide guarantees on the policy quality. Rather than constraining the size of the solution prior to the execution of the algorithm, it is equally possible to come up with a policy within a bound of the optimal policy.

In this paper, we design policy iteration (PI) algorithms that provide many desirable properties that current infinite horizon solvers lacked. First of all, we define exact and approximate Dynamic Programming (DP) backup operators that enable us to compute an improved value function. Secondly, we build up the joint policy based on the improved value function, this circumvents the problem of the exhaustive backup. Finally, we state and prove approximation error bounds on the resulted policy quality. The difficulty of this work lies in the definition of backup operators that guarantee: (1) the decentralization is preserved over the updates of the value function and the transformations of the corresponding policy while avoiding the exhaustive backup; (2) the updates of the value function are essentially DP updates. To leverage the first issue, we introduce new multi-agent concepts namely *basis objects* and *sets*, *i.e.,* partial joint information of the team that is sufficient to satisfy the decentralization. Even more importantly, these concepts help circumventing the problem of the exhaustive backup. To handle the second point, we perform essentially a single-agent DP update and keep track only on the value function that satisfies the decentralization.

2. BACKGROUND AND RELATED WORK

We review the DEC-POMDP model and the associated notation, and provide a short overview of the state-of-the-art algorithms.

2.1 The DEC-POMDP Model

DEFINITION 1. *A n-agent DEC-POMDP can be represented using a tuple $(I, S, \{A^i\}, P, \{\Omega^i\}, O, R)$, where: I is a finite set of agents indexed by $1 \cdots n$; $S = \{s\}$ is a finite set of joint states; A^i denotes a finite set of actions available for agent i, and $A = \otimes_{i \in I} A^i$ is the*

AAMAS 2009 Workshop on Multi-agent Sequential Decision-Making in Uncertain Domains, May 11, 2009, Budapest, Hungary.

set of joint actions, where $a = (a^1, \cdots, a^n)$ denotes a joint action; $P\colon S \times A \to \triangle S$ is a Markovian transition function. $P(s'|s,a)$ denotes the probability of transiting from state s to state s' when taking action a; Ω^i defines a finite set of observations available for agent i, and $\Omega = \otimes_{i\in I}\Omega^i$ is the set of joint-observations, where $o = (o^1, \cdots, o^n)$ is a joint observation; $O\colon A \times S \to \triangle\Omega$ is an observation function. $O(o|a,s')$ denotes the probability of observing joint observation o given that joint action a was taken and led to state s'; $R\colon A \times S \to \Re$ is a reward function. $R(a,s)$ denotes the reward signal received when executing action a in state s;

Optimization criterion. The DEC-POMDP model is parameterized by: $b_0 \in \triangle S$, the initial belief distribution, *i.e.,* the team belief over its initial state. The belief state (belief for short) $b \in \triangle S$ defines a probability distribution of the team over the underlying states. The next belief, denoted $b^{a,o} = \tau(b,a,o)$, that incorporates the latest joint action-observation pair (a,o) and the current belief b, is updated as follows:

$$b^{a,o}(s') = \eta O(o|s',a)\sum_s b(s)P(s'|s,a) \qquad (1)$$

where η is a normalizing constant. When the agents operate over an unbounded number of time-steps, the DEC-POMDP has a discount factor, $\gamma \in [0,1)$. This model is coined by the term infinite-horizon DEC-POMDP with discounted rewards. Solving such a DEC-POMDP means finding a *joint policy* δ that yields the highest expected value $V^\delta(b_0) = \max_\delta \mathbb{E}\left[\sum_{\tau=0}^{\infty}\gamma^\tau R(a_\tau,s_\tau)|b_0,\delta\right]$ where $V^\delta(b)$ denotes the expected sum of discounted rewards obtained given that joint policy δ is executed starting in belief b. Given the definition, the true value of a starting belief b at time step τ, which we write $V_\tau(b)$, is just $V^{\delta^*_\tau}(b)$ – where δ^*_τ is an optimal policy at time step τ. Since finding an ε-optimal joint policy is known to be intractable, we therefore state our optimization criterion as finding the best joint policy based on a small set of representative beliefs B and the amount of time that is allotted to the algorithm. With this criterion, we want to design novel infinite horizon DEC-POMDP algorithms that update alternatively all, a belief set B_τ, the optimal value function V_τ over B_τ and the corresponding joint policy δ_τ, until Bellman residual, *i.e.,* $\|V_\tau - V_{\tau-1}\|_\infty$, is less or equal to $2\varepsilon\gamma/(1-\gamma)$. That is the returned joint policy is an ε-optimal policy with respect to belief space B.

Policy representation. Throughout the paper, a policy for single agent i, *deterministic finite state controller* (DFSC), can be represented as policy graph $\delta^i = (X^i, \pi, \eta, x^i_0)$, where $X^i = \{x^i\}$ denotes a set of machine states; $\pi(x^i)$ is the individual action a^i selected in machine state x^i; $\eta(x^i, o^i)$ is the successor machine state when individual observation o^i is perceived in machine state x^i; and x^i_0 is the starting machine state. We denote a joint policy, *deterministic joint finite-state controllers* (DJFSC) by δ. A DJFSC can also be represented as joint policy graph $\delta = (X, \pi, \eta, x_0)$, where: $X := \otimes^n_{i=1} X^i$ denotes a set of machine states; $\pi(x)$, $\eta(x,o)$ and x_0 are defined as for DFSCs. This contrasts with the standard representation based on policy vectors, *i.e.,* vectors of individual policies, one for each agent. Each joint machine state x is associated to a hyperplane α_x – that is the vector value that denotes the expected sum of discounted rewards obtains when executing δ starting in machine state x.

2.2 Related Work

In this section, we first present POMDP relevant work that we use as a foundation for our PI algorithms. Then, we discuss the state-of-the-art ε-optimal approach to solving infinite horizon DEC-POMDPs.

Policy Iteration for POMDPs. A special case of DEC-POMDP, in which each agent shares its private information with its teammates at each time step, is called a multi-agent POMDP (MPOMDP). Because all the information available to the team at each time step is known, MPOMDPs can be solved using single-agent POMDP techniques. Over the past few years, exact and approximate PI algorithms have been proposed for POMDPs by [15, 6] and [8], respectively. These algorithms, summarized in Algorithm 1, share the same structure that consists of a threefold method: first, compute the value function V_τ of the current DJFSC δ_τ (*policy evaluation*); secondly, update the value function V_τ represented by a set $\Gamma_{\tau+1}$ of hyperplanes into an improved value function $V_{\tau+1}$ represented by a set Γ_τ of hyperplanes (*policy improvement*); and finally, transform the current DJFSC $\delta_{\tau-1}$ into an improved DJFSC δ_τ (*policy transformation*) with respect to $\Gamma_{\tau+1}$. These processes alternate until a convergence criterion is reached. The policy evaluation step is straightforward when a policy is represented as a DJFSC [6]. This can be achieved by solving the system of linear equations:

$$\alpha_x(s) = R(s,\pi(x)) + \gamma\sum_{s',o}P(s'|s,\pi(x))O(o|s',\pi(x))\alpha_{\eta(x,o)} \qquad (2)$$

where x and $\eta(x,o)$ are machine states. In addition, the policy improvement step relies on the fact that a POMDP can be reformulated as a belief MDP. Indeed, it is well-known that in POMDPs the belief is a sufficient statistic for a given history. Therefore, the value function V_τ can be updated using DP. [13] show how to implement the DP update of a value function V_τ by exploiting its piece-wise linearity and convexity. Because the value function is a mapping over a continuous $|S|$-dimensional space, $V_{\tau+1}$ cannot be directly computed. Instead, the corresponding set $\Gamma_{\tau+1}$ can be generated through a sequence of operations over Γ_τ.

Algorithm 1 Policy Iteration

```
1:  procedure PI(initial policy: $\delta_0$, belief space: $B$)
2:      Initialization: $\delta_{\tau+1} = \delta_\tau = \delta_0$
3:      repeat
4:          % policy evaluation
5:          Compute $\Gamma_\tau \leftarrow \{\alpha_i\}$, i.e., value function of $\delta_\tau$
6:          % policy improvement
7:          $\Gamma_{\tau+1} \leftarrow$ BACKUP$(\Gamma_\tau, B)$ where $B$ may be $\triangle S$
8:          % policy transformation: $\delta_\tau \to \delta_{\tau+1}$
9:          Re-initialization: $\tau = \tau + 1$
10:     until $\|V_{\tau+1} - V_\tau\|_\infty \le 2\varepsilon\gamma/(1-\gamma)$
```

DP update. We now describe the straightforward implementation of DP update for POMDPs [14]. First, we generate intermediate sets $\Gamma^{a,*}$ and $\Gamma^{a,o}$ $\forall a,o$ (*Step 1*): $\Gamma^{a,*} \leftarrow \alpha^{a,*}(s) = R(s,a)$ and $\Gamma^{a,o} \leftarrow \alpha^{a,o}_i(s) = \sum_{s'}P(s'|s,a)O(o|s',a)\alpha_i(s')$, $\forall\alpha_i \in \Gamma_\tau$. Next, we create Γ^a ($\forall a \in A$), the cross-sum over joint observations, which includes one $\alpha^{a,o}$ from each $\Gamma^{a,o}$ (*Step 2*): $\Gamma^a \leftarrow \Gamma^{a,*} \oplus_{o\in\Omega}\Gamma^{a,o}$. Finally, we take the union of Γ^a sets (*Step 3*): $\Gamma_{\tau+1} = \cup_{a\in A}\Gamma^a$. It is often the case that a hyperplane in $\Gamma_{\tau+1}$ is completely dominated by another hyperplane ($\alpha\cdot b \le \alpha'\cdot b$, $\forall b$) or by combination of other hyperplanes. Those hyperplanes can be pruned away at each steps of the update without affecting the policy quality. Finally, [6] suggests a number of rules that enables us to transform the current policy δ_τ into an improved one $\delta_{\tau+1}$. To the best of our knowledge, there is no known techniques that extend completely this approach to multiple agent settings.

Policy Iteration for DEC-POMDPs. [2] proposed recently an attempt to extend PI from single-agent into decentralized multi-agent settings. This algorithm builds over a series of steps a vector of stochastic finite state controllers, one for each agent. Each step consists of a twofold method: the exhaustive backup and the pruning of dominated stochastic machine states. For each agent i, the exhaustive backup takes as input the current set of stochastic machine states X^i_τ. Then, it builds new stochastic machine states $X^i_{\tau+1}$ for all possible joint actions, such that the successor links of the new stochastic machine state are associated only to stochastic machine states that appear in the current sets. The resulted set of stochastic machine states $X^i_{\tau+1}$ is exponential with respect to the number of observation, *i.e.,* $|X^i_{\tau+1}| = |A^i||X^i_\tau|^{|\Omega^i|} + |X^i_\tau|$. Thus, the joint stochastic controller grows by $\prod_{i=1}^n |A^i||X^i_\tau|^{|\Omega^i|}$, which grows exponentially with n. Thereafter, a pruning step eliminates dominated stochastic machine states without loosing the ability to eventually converging to an ε-optimal controller. Performing this pruning step, however, can be extensive since they require a linear program and its dual. While interesting, this PI algorithm laks many desirable properties of the PI algorithm in single-agent settings. Among many, the algorithm fails to define a DP backup operator for the decentralized multi-agent case. This would enable us to derive the Bellman residual $\| V_\tau - V_{\tau-1}\|_\infty$ – distance between two successive value functions and even more importantly approximation error bounds. Furthermore, as already discussed in the POMDP section, dominated stochastic machine states can be pruned away at earlier steps of the backup. Doing so, would avoid the exhaustive generation of all possible stochastic machine states which is expensive.

3. POLICY ITERATION REQUIEREMENTS

In order to extend PI algorithms along with its properties from single-agent to the decentralized multi-agent settings, we need to face two key issues. As we aim at finding a policy for DEC-POMDPs, we need to make sure that our backup operator $\mathbb{H}_B$ transforms set Γ_τ into an improved set $\Gamma_{\tau+1}$ while preserving the decentralization – that is all hyperplanes $\alpha \in \Gamma_{\tau+1}$ and the improved policy $\delta_{\tau+1}$ satisfy the decentralization constraint. On the other hand, to inherit POMDP properties our backup operator need essentially to be the DP backup operator – that is the value $V_{\tau+1}(b)$ at belief b depends on values $V_\tau(b')$ of its successor beliefs b': $V_{\tau+1}(b) = \mathbb{H}_B V_\tau(b) = \max_a \mathbb{E}\left[R(a,b) + \gamma \sum_o V_\tau(\tau(b,a,o))\right]$.

3.1 Satisfying the Decentralization

In this section, we introduce new multi-agent planning concepts namely *basis* objects and sets. These concepts help preserving the decentralization while updating hyperplanes or transforming policies.

Preliminary definitions. A set of basis joint observations also called *basis set* and denoted $\mathring{\Omega} \subseteq \Omega$ is a set of joint observations of the smallest size where each individual observation of any agent, *e.g.,* $o^i \in \Omega^i$, is included in at least one joint observation of the basis set, *e.g.,* $(\cdots, o^i, \cdots) \in \mathring{\Omega}$. Because all individual observations of all agents are represented in at least one joint observation of the basis set, the *cardinality of a basis set* $\mathring{\Omega}$, denoted $\kappa(\Omega)$ (κ for short), is given by:$\kappa = |\mathring{\Omega}| = \max_{i\in I} |\Omega^i|$. This holds for any basis set and even more importantly for any number of agents. In the remainder of the paper, we assume that each agent has the same number of individual observations [1]. Therefore, a straightforward way of building a basis set $\mathring{\Omega}$ is to include the largest set of joint observations such that any pair of joint observations is *component-wise different, e.g.,* (o_1, o_2) and (o_2, o_1). A *basis object* is any object, *e.g.,* policy or hyperplane, defined only over basis observations. A *basis joint policy* $\delta_B : \mathring{\Omega}^* \to A$ is a DJFSC defined only over basis joint observations. A *basis hyperplane* is a vector of hyperplanes, one for each basis joint observation. We call the *complement hyperplane*, a vector of hyperplanes, one hyperplane for each non basis joint observation $o \in \Omega \backslash \mathring{\Omega}$. Any object that satisfies the decentralization constraint is said to be valid.

[1]Fictitious individual observations are added to agents if necessary such that the assumption always holds.

Valid policies. We introduce below a criterion that checks whether a DJFSC δ satisfies the decentralization constraint, *i.e.,* it is a *valid* policy.

LEMMA 1. *A DJFSC $\delta = (X, \pi, \eta, x_0)$ satisfies the decentralization constraint if and only if it corresponds to a vector of DFSCs $(\delta^1, \cdots, \delta^n)$ such that: $\forall x \in X, \exists (x^1, \cdots, x^n) : \pi(x) = (\pi(x^1), \cdots, \pi(x^n))$ and $\eta(x, o) = (\eta(x^1, o^1), \cdots, \eta(x^n, o^n))$ where $o = (o^1, \cdots, o^n)$ and $\delta^i = (X^i, \pi, \eta, x^i_0)$.*

This lemma states that the joint action taken for a given sequence of joint observations when the vector of DFSCs $(\delta^1, \cdots, \delta^n)$ is executed is exactly the joint action taken when δ is executed after perceiving the same sequence of joint observations. Now we are ready to claim our main theorem.

THEOREM 1. *The basis DJFSC δ_B is the sufficient information to build a DJFSC δ that is valid.*

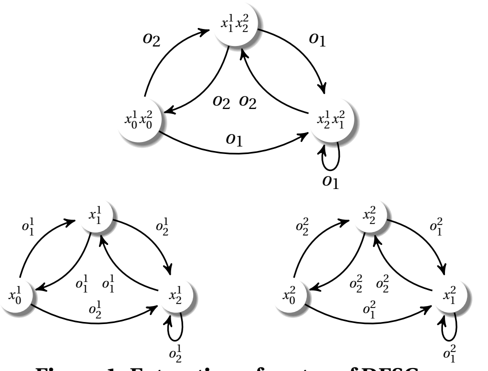

Figure 1: Extraction of vector of DFSCs.

To prove this we suggest a constructive two-step method that builds a unique DJFSC δ based on δ_B: first, we build the unique vector of DFSCs associated with a basis δ_B. To do so, we extract from δ_B the DFSC δ^i for each agent $i = 1 \cdots n$. This is achieved by removing from δ_B the components related to the other agents, such that only nodes and arcs that are labeled by individual actions and observations of agent i are kept. This step provides us with the vector of DFSCs $(\delta^1, \cdots, \delta^n)$. We then build DJFSC δ based on the vector of DFSCs $(\delta^1, \cdots, \delta^n)$ by using Lemma 1: $\forall x \in X, \pi(x) = (\pi(x^1), \cdots, \pi(x^n))$ and $\eta(x, o) = (\eta(x^1, o^1), \cdots, \eta(x^n, o^n))$, where $o = (o^1, \cdots, o^n)$ and $X = \otimes_{i=1}^n X^i$. One can merge together machine states $x \in X$ where the associated action $\pi(x)$ and the successor links $\eta(x, o)$ are the same. This two-step method proves the existence of such a DJFSC . But, we still need to prove that δ_B is the sufficient information to build δ. This is achieved by removing either a node or an arc from δ_B. In that case, the vector of DFSCs $(\delta^1, \cdots, \delta^n)$ will consist of DFSCs δ^i that lack a node or an arc. In accordance with this two-step method, we illustrate in Figure 1 the vector of DFSCs (down) extracted from a basis DJFSC (up) where the basis set is $\mathring{\Omega} = \{o_1 = (o^1_1, o^2_2), o_2 = (o^1_2, o^2_1)\}$.

Valid hyperplanes. A similar result for a basis hyperplane can be easily derived from the above theorem.

COROLLARY 1. *Let $\{\alpha^o\}_{o\in\mathring{\Omega}}$ be a basis hyperplane. There exists a unique complement hyperplane $\{\alpha^o\}_{o\in\Omega\setminus\mathring{\Omega}}$, such that hyperplane $\alpha = \sum_{o\in\Omega}\alpha^o$ is valid.*

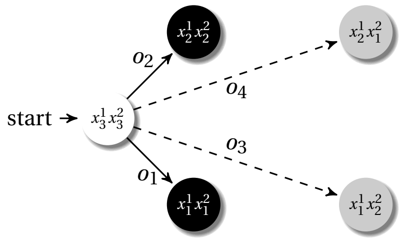

Figure 2: The construction of a valid hyperplane (in white) and the complement of basis hyperplane hyperplane (in gray) given the basis hyperplane (in black).

One can look at $\{\alpha^o\}_{o\in\mathring{\Omega}}$ as a set of hyperplanes that represents leaf nodes of a one-step basis joint policy δ_B. Therefore, using Theorem 1, we are able to build the one-step joint policy δ associated to δ_B and thus the corresponding hyperplane. Consider the example depicted in Figure 2, where each agent has observation set $\Omega^i = \{o^i_1, o^i_2\}$ for $i = 1,2$. Let $\mathring{\Omega} := \{o_1 = (o^1_1, o^2_2), o_2 = (o^1_2, o^2_1)\}$ be the basis set, therefore its complement is $\Omega\setminus\mathring{\Omega} := \{o_3 = \{o^1_1, o^2_1\}, o_4 = \{o^1_2, o^2_2\}\}$. We know $\alpha_{(x^1_3, x^2_3)}$ is built based on basis hyperplanes $\alpha_{(x^1_2, x^2_2)}$ and $\alpha_{(x^1_1, x^2_1)}$ that are associated to machine states $\eta(x, o_1) = (x^1_2, x^2_2)$ and $\eta(x, o_2) = (x^1_1, x^2_1)$, respectively. Thus, using Theorem 1, it turns out that the DFSC of agent 1 is transformed by adding machine state x^1_3 where $\eta(x^1_3, o^1_2) = x^1_2$ and $\eta(x^1_3, o^1_1) = x^1_1$, and the DFSC of agent 2 is transformed by adding machine state x^2_3 where $\eta(x^2_3, o^2_2) = x^2_2$ and $\eta(x^2_3, o^2_1) = x^1_2$. As a consequence, the complement hyperplane $\{\alpha_{\eta(x,o_3)}, \alpha_{\eta(x,o_4)}\}$ is a vector of hyperplanes associated to machine states (x^1_1, x^2_2) and (x^1_2, x^2_1), respectively. For instance to determine the machine state associated to $\alpha_{\eta(x,o_3)}$, we use $\eta(x, o_3) = (\eta(x^1_3, o^1_1), \eta(x^2_3, o^2_1))$, *i.e.,* (x^1_1, x^2_2). We call this procedure the construction of valid hyperplanes.

Preserving the decentralization. We are now ready to state the update and transformation rules that preserve the decentralization. It is straightforward to see that the construction of valid hyperplanes discussed above preserves the decentralization. Therefore, an update that preserves the decentralization consists in generating hyperplanes in $\Gamma_{\tau+1}$ using the construction of valid hyperplanes based on hyperplanes in the current set Γ_τ. On the other hand, we identify two transformation rules that preserve the decentralization. The first rule is to remove individual machine states x^i along with all joint machine states $x = x^i x^{-i}$ and hyperplanes α_x associated. The second rule is to replace an individual machine state x^i by another one $\hat{x}^i$ every where x^i appears. Because these are essentially transformations of individual controllers, the resulting joint controller is still valid.

3.2 The Sufficient Statistic

In this section, we explain how planning only over reachable beliefs allows the optimal policy of a DEC-POMDP to be found.

Planning over reachable beliefs. In order to be optimal, the Markov assumption requires that a policy depends on all the information available to the team at each time step. In DEC-POMDPs, at the execution time the agents are unaware of private information of their team-mates. However, in simulation each agent can divulge its private information to its team-mates. Therefore, the agents can maintain a complete joint history trace of all joint observations and joint actions they ever simulated. This joint history can get very long as time goes on. A well-known fact is that this joint history can be summarized via a belief. Unfortunately, because of the decentralization constraint a belief alone is not sufficient to condition the selection of a joint action. Nevertheless, we can still show that planning only over reachable beliefs allows the optimal policy to be computed. To better understand this, let b_0 be an initial belief we need to compute the optimal value. We know that the machine state x whose hyperplane α_x yields the highest value for belief b_0 depends on machine states whose hyperplanes $\{\alpha_{\eta(x,o)}\}_{o\in\Omega}$ are selected for its successor beliefs $\{b_{\pi(x),o}\}_{o\in\Omega}$. Because of the decentralization constraint, the hyperplanes selected for successor beliefs $\{b_{\pi(x),o}\}_{o\in\Omega}$ are dependent on each other – that is the selection of hyperplanes for some successor beliefs in $\{b_{\pi(x),o}\}_{o\in\Omega}$ constrains the selection of hyperplanes for the remaining. As previously discussed, if we select a basis hyperplane for successor belief $b' \in \{b_{\pi(x),o}\}_{o\in\mathring{\Omega}}$ associated to basis joint observations, we determine directly the hyperplanes that are assigned to the remaining successor beliefs $\{b_{\pi(x),o}\}_{o\in\Omega\setminus\mathring{\Omega}}$. A similar argument can be used to show that successor beliefs of beliefs $\{b_{\pi(x),o}\}_{o\in\Omega}$ are also dependent on each other, and so on. Thus, in the decentralized multi-agent settings, the value of any belief $b_{\tau+1}$ depends on the value of all beliefs reachable starting in its precedent belief b_τ. It is worth noting that because the initial belief b_0 does not have a precedent belief, its value depends only on all reachable beliefs starting in b_0. This permits us to claim that planning only over reachable beliefs allow us to compute the optimal policy for a given initial belief. Although, the optimal value $V_{\tau+1}(b)$ cannot be computed directly for each reachable belief (since it may depends on infinitely many other beliefs), the corresponding set $\Gamma_{\tau+1}$, that includes the hyperplane that is maximal for b while satisfying the decentralization constraint, can be generated through a sequence of operations on the set Γ_τ.

Selection of belief set B. Since the selection of a finite set of beliefs B is crucial to the solution quality of all point-based algorithms, we rely on sampling techniques whose efficiency has been proven. In particular, [10] described a forward simulation that generates the sample belief set. The procedure starts by selecting the initial belief set $B_0 := \{b_0\}$ including the initial belief b_0 at time $\tau = 0$. Then, for time $\tau = 1, 2, \cdots$, it expands B_τ to $B_{\tau+1}$ by adding all possible $b_{a,o}$ produced, and this $\forall b \in B_\tau$, $\forall a \in A$, $\forall o \in \Omega$, such that $pr(o|b,a) > 0$. Then, the belief set $\bar{\Delta} := \cup_{\tau=0}^{\infty} B_\tau$ is the set of beliefs reachable during the simulation time. It is therefore sufficient to plan only over these beliefs in order to find an optimal joint policy customized for a team of agents that starts from belief b_0, since $\bar{\Delta}$ constitutes a closed inter-transitioning belief set. Unfortunately, it is likely that $\bar{\Delta}$ is infinitely large. Thus, rather than keeping all possible $b^{a,o}$ we keep only a single $b^{a,*}$ that is the one that has the maximum $\mathbb{L}_1$ distance to the current B. And we add it into B only if its $\mathbb{L}_1$ distance is beyond a given threshold ε.

4. POLICY ITERATION FOR DEC-POMDPS

Our PI ($\mathbb{H}_B$-PI) algorithm is summarized in Algorithm 1, with the principal structure shared with its POMDP counterparts [6, 8]. Similarly to the single-agent case, $\mathbb{H}_B$-PI consists of a threefold method: policy evaluation; policy improvement; and policy transformation. While the policy evaluation step is straightforward as previously discussed, processing the two later steps while providing guarantees on the satisfaction of the decentralization constraint is not trivial.

Policy Improvement. To show the importance of using basis objects and sets for DP updates in decentralized multi-agent settings, we first consider state-of-the-art alternative strategies from either single-agent or multi-agent cases. In the single-agent case, we need first to generate the whole set $\Gamma_{\tau+1}$ and thereafter prune non valid hyperplanes [14]. This approach is, of course, hopelessly computationally intractable, as it requires the generation of $|A||\Gamma_\tau|^{\kappa^n}$ hyperplanes which is doubly exponential in κ and n. In the multi-agent case, we build first all possible individual policies, that induces set $\Gamma_{\tau+1}$ of valid hyperplanes and thereafter prune dominated hyperplanes [2]. While this approach is more tractable than the previous one, the exhaustive backup limits its scalability. Indeed, set $\Gamma_{\tau+1}$ grows by $|A|\prod_{i=1}^n |X_\tau^i|^{|\Omega^i|}$, which grows exponentially with n.

We are now ready to present the implementation of our backup operator $\mathbb{H}_B$. Similarly to the single-agent case, we start by creating intermediate sets $\Gamma^{a,*}$ and $\Gamma^{a,o}$, $\forall a \in A$, $\forall o \in \mathring{\Omega}$ (*step* 1): $\Gamma^{a,o} \leftarrow \alpha_i^{a,o}(s) = \sum_{s'} P(s'|s,a)O(o|s',a)\alpha_i(s')$, $\forall \alpha_i \in \Gamma_\tau$ and $\Gamma^{a,*} \leftarrow \alpha^{a,*}(s) = R(s,a)$. Next we create Γ^a ($\forall a \in A$), the cross-product over basis joint-observations, which includes one $\alpha^{a,o}$ from each $\Gamma^{a,o}$ (*step* 2): $\Gamma^a = \left(\otimes_{o\in\mathring{\Omega}} \Gamma^{a,o}\right) \otimes \Gamma^{a,*}$. Then, we create for each basis hyperplane $\alpha = \{\alpha^{a,o_1}, \cdots, \alpha^{a,o_\kappa}, \alpha^{a,*}\} \in \Gamma^a$ its complement hyperplane $\{\alpha^{a,o_{\kappa+1}}, \cdots, \alpha^{a,o_{|\Omega|}}\}$ with respect to Corollary 1 discussed above. We therefore replace in Γ^a, the hyperplane α by the hyperplane built as a cross-sum over hyperplanes in (*step* 3): $\forall \alpha \in \Gamma^a$, $\Gamma^a \leftarrow \Gamma^a \backslash \{\alpha\}$ and $\Gamma^a \leftarrow \alpha' = \alpha^{a,*} + \gamma \sum_o \alpha^{a,o}$. Afterwards, we take the union of Γ^a sets (*step* 4): $\Gamma_{\tau+1} = \cup_a \Gamma^a$. Finally, we include the initial set of hyperplanes (*step* 5): $\Gamma_{\tau+1} = \Gamma_{\tau+1} \cup \Gamma_\tau$ to preserve the integrity of the solution. In practice, many of the hyperplanes α_i in the final set $\Gamma_{\tau+1}$ may be completely dominated by another hyperplane. To prune away those hyperplanes while preserving the decentralization, we rely on an individual machine state dominance criterion:

THEOREM 2. *Let X^i and X^{-i} be sets of machine states of agent i and the other agents except agent i, respectively. A machine state $x^i \in X^i$ is dominated iff: $\exists \hat{x}^i \in X^i \backslash x^i : \alpha_{\hat{x}^i x^{-i}} \cdot b \geq \alpha_{x^i x^{-i}} \cdot b$, $\forall b \in B, \forall x^{-i} \in X^{-i}$.*

PROOF. We rely on a proof by contradiction. Assume (1) machine state $x^i \in X^i$ is non dominated and (2) $\exists \hat{x}^i \in X^i \backslash x^i : \alpha_{\hat{x}^i x^{-i}} \cdot b \geq \alpha_{x^i x^{-i}} \cdot b$, $\forall b \in B, \forall x^{-i} \in X^{-i}$. From the first claim, we derive that some of the reachable beliefs yield their optimal values under the decentralization constraint at hyperplanes $\alpha_{x^i x^{-i}}$, where $x^{-i} \in X^{-i}$. From the second argument, we derive that there exists a machine state $\hat{x}^i \in X^i$ such that hyperplane $\alpha_{\hat{x}^i x^{-i}}$ point-wise dominates hyperplane $\alpha_{x^i x^{-i}}$ for any machine state $x^{-i} \in X^{-i}$. In addition, by replacing machine state x^i by machine state $\hat{x}^i$ the decentralization constraint still holds. As a result, this modification transforms the initial policy into a policy with a value function that increases for at least one belief state $b \in B$ and decreases for no $b \in B$ – which is a contradiction. $\square$

The pruning then alternates from one agent to another until no more machine states are pruned. We extend the machine state dominance criterion to prune successively intermediate sets $\Gamma^{a,o}$ and Γ^a, for all $a \in A$ and $o \in \mathring{\Omega}$ and set $\Gamma_{\tau+1}$. As κ and $|\Gamma_\tau|$ grow the overhead of performing these pruning mechanisms is non-negligible.

The point-based B&B backup $\bar{\mathbb{H}}_B$. This method aims at building the next set of hyperplanes $\Gamma_{\tau+1}$ using hyperplanes in Γ_τ. The problem of finding the next set of hyperplanes $\Gamma_{\tau+1}$ given a finite set of beliefs B and intermediate sets $\Gamma^{a,*}$ and $\Gamma^{a,o}$ ($\forall a \in A$ and $\forall o \in \mathring{\Omega}$), corresponds to the problem of finding basis hyperplane $(\alpha_b^{a,o_1}, \cdots, \alpha_b^{a,o_\kappa})$ such that the corresponding valid hyperplane $\alpha_b \in \Gamma_{\tau+1}$ is maximal for belief b, and this for each belief in B. One may suggest to solve such a problem using essentially the single-agent point-based backup operator while preserving the decentralization. That is, solving the problem by selecting hyperplane $\alpha_b^{a,o}$ that is maximal at b, one hyperplane for each basis joint observation. Then, we build the corresponding valid hyperplane. Unfortunately, the resulting hyperplane yields a lower-bound value. This is because its complement hyperplane can provide poor rewards. As a result, the overall contribution of the basis hyperplane is diminished by the poor contribution from its complement hyperplane. If we relax the problem by skipping the decentralization constraint, we build hyperplane α_b that is an upper-bound value at belief b and potentially a non valid hyperplane. Hence, applying single-agent point-based methods do not lead to the best valid hyperplane for each belief. As this is essentially a combinatorial problem, we rely on a point-based branch-and-bound backup operator.

To describe this method, the following definitions are required: (a) $\vec{\alpha}^a$ is a vector of hyperplanes one for each joint-observation; (b) $\vec{\alpha}^a(o)$ is the selected hyperplane $\alpha_i^{a,o} \in \Gamma^{a,o}$. The forward search in the space of vectors of hyperplanes can be considered as an incremental construction of the best vector of hyperplanes based on optimistic evaluations of only partially completed vectors of hyperplanes. In each step of the search, the most promising partially completed vector of hyperplanes is selected and further developed, hence the best first approach. For a completely defined vector of hyperplanes $\vec{\alpha}^a$, we are able to find the corresponding hyperplane $\alpha = \alpha^{a,*} + \gamma \sum_o \vec{\alpha}^a(o)$. We then state our maximization problem as follows: $\alpha_b = \arg\max_\alpha (\alpha \cdot b)$. Notice that a partially completed vector of hyperplane is a vector where $\vec{\alpha}^a$ is only partially defined. Moreover, any partially completed vector of hyperplanes can be completed by assigning hyperplane $\alpha^{a,o} \in \Gamma^{a,o}$ that yields the highest value for belief b at points $\vec{\alpha}^a(o)$ where $\vec{\alpha}^a$ is not constrained. In order to determine whether or not to expand the leaf node of the search tree corresponding to a partially completed vector of hyperplanes, we compute an upper-bound value of any partially completed vector of hyperplanes for a given belief b. We define the upper-bound based on the decomposition of the exact estimate into two estimates. The first estimate, $G(\vec{\alpha}^a, b)$, is the exact estimate coming from points $\vec{\alpha}^a(o)$ where $\vec{\alpha}^a$ is constrained. The second estimate, $H(\vec{\alpha}^a, b)$, is the upper-bound value coming from points $\vec{\alpha}^a(o)$ where $\vec{\alpha}^a$ is not constrained. We introduce sets of joint-observations Ω_1 and Ω_2 (such that $\Omega = \Omega_1 \cup \Omega_2$) that correspond to joint-observations that lead to constrained points and non-constrained points, respectively.

$$\bar{V}(\vec{\alpha}^a, b) = \underbrace{\left(\alpha^{a,*} + \gamma \sum_{o\in\Omega_1} \vec{\alpha}^a(o)\right) \cdot b}_{G(\vec{\alpha}^a, b)} + \underbrace{\left(\gamma \sum_{o\in\Omega_2} \max_{\alpha^{a,o}} (\alpha^{a,o} \cdot b)\right)}_{H(\vec{\alpha}^a, b)}$$

In the following, we draw our attention to a single search tree of our point-based B&B backup operator with the following entries: a belief b; a joint action a and intermediate sets $\Gamma^{a,o}$ ($\forall o \in \Omega$), see Algorithm 2. We start by initializing the pool of *live* nodes with a partially defined vector $\vec{\alpha}_0^a$ where none of the points $\vec{\alpha}_0^a(o)$ ($\forall o \in \Omega$) is defined, and the value hereof is used as the value (called *incumbent*) of the current best solution (line 1). In each iteration of the heuristic, the node $\vec{\alpha}_i^a$ that yields the highest upper-bound is selected for exploration from the pool of live nodes (lines 3 – 4). Then, a branching is performed: two or more children of the node are constructed through the definition of a single point

$\vec{\alpha}^a_i(o)$ (line 5), for $o \in \mathring{\Omega}$. Furthermore, for each of the generated child nodes $\vec{\alpha}^a_i$, points $\vec{\alpha}^a_i(o)$ for $o \in \Omega \backslash \mathring{\Omega}$ that can be defined based on already constrained points are defined and the upper-bound is calculated. In case the node corresponds to a completely defined vector of hyperplanes its upper-bound is its exact estimate value, the value hereof is compared to the incumbent, and the best solution and its value are kept (lines 7 – 10). If its upper-bound is not better than the incumbent, the node is discarded, since no completely defined descendant nodes of that node can be better than the incumbent. Otherwise the possibility of a better solution in the descendant nodes cannot be ruled out, and the node is then joined to the pool of live nodes (line 11). When the search tree has been completely explored, the heuristic starts a new search tree with a new joint-action and the current best solution, until all joint-actions have been processed and this for each belief $b \in B$.

Algorithm 2 Point-based branch and bound backup

```
 1: procedure $\bar{\mathbb{H}}_B$-BACKUP($a, b, \{\Gamma^{a,o}\}_{o\in\Omega}$)
 2:     Initialize: Incumbent := $V(b)$; Live := $\{\vec{\alpha}^a_0\}$
 3:     repeat
 4:         Select $\vec{\alpha}^a_i \in$ Live : $\forall \vec{\alpha}^a_j \in$ Live, $\bar{V}(\vec{\alpha}^a_j, b) \le \bar{V}(\vec{\alpha}^a_i, b)$
 5:         Live := Live \ $\{\vec{\alpha}^a_i\}$
 6:         Branch on $\vec{\alpha}^a_i$ generating $\vec{\alpha}^a_1, \cdots, \vec{\alpha}^a_{i_k}$
 7:         for $1 \le p \le k$ do
 8:             if $\bar{V}(\vec{\alpha}^a_{i_p}, b) >$ Incumbent then
 9:                 if $\vec{\alpha}^a_{i_p}$ is completely defined then
10:                     Incumbent := $\bar{V}(\vec{\alpha}^a_{i_p}, b)$
11:                     Solution := $\alpha^{a,*} + \sum_o \vec{\alpha}^a_{i_p}(o)$
12:                 else Live := Live $\cup \{\vec{\alpha}^a_{i_p}\}$
13:     until Live = $\emptyset$
14:     return Solution
```

Policy Transformation. [6] describes a policy transformation mechanism based on a simple comparison of $\Gamma_{\tau+1}$ and Γ_τ, for the single-agent case. These rules are extended here for multiple agent settings. The procedure below iterates between agents until no more machine states can be eliminated. Notice that these transformation rules preserve the decentralization and decrease the value for no $b \in B$.

Transformation rules

For all $y^i \in X^i_{\tau+1}$:

- *rule* 1: If the action and successor links associated with y^i duplicate those of a machine state of δ^i, then keep that machine state unchanged in $\delta^i_{\tau+1}$;
- *rule* 2: Else if the machine y^i dominates a machine state $x^i \in X^i_\tau$ associated with a machine state in δ^i_τ, change the action and successor links of that machine state to those that correspond to x^i;
- *rule* 3: Else add a machine state to $\delta^i_{\tau+1}$ that has the action and successor links associated with y^i;

Finally, prune any machine state for which there is no corresponding hyperplane in $\Gamma_{\tau+1}$, as long as it is not reachable from a machine state to which an hyperplane in $\Gamma_{\tau+1}$ does correspond.

5. THEORETICAL ANALYSIS

This section presents convergence, error bound and the complexity arguments that draw on earlier approaches in single agent settings.

Convergence Properties. Point-based PI algorithms, despite being approximate methods, inherit many desirable properties of PI algorithms from POMDPs [6, 8], including: (1) If a DJFSC has not converged, the policy improvement transforms it into a DJFSC with a value function that increases for at least on belief $b \in B$ and decreases for no $b \in B$; (2) Point-based PI algorithms converge to an approximate DJFSC after a finite number of iterations; (3) the exact PI $\mathbb{H}$-PI algorithm ($B = \triangle S$) converges to an ε-optimal policy.

Error bounds. We now show that the error between V_τ and the optimal value function V^* is bounded. The bound depends on how densely B samples the belief set $\bar{\triangle}$; with denser sampling, V_τ converges to V^*. Cutting off the point-based PI ($\mathbb{H}_B$-PI) iterations at any sufficiently large time step, we know that the difference between V_τ and the optimal value function V^* is not too large. The overall error in $\mathbb{H}_B$-PI, $\|V_\tau - V^*\|_\infty$, is bounded by: $\|V_\tau - V^*_\tau\|_\infty + \|V^*_\tau - V^*\|_\infty$. Because, $\mathbb{H}$ (resp. $\mathbb{H}_B$) is a contraction mapping, $\mathbb{H}V_\tau(b) = \max_a \mathbb{E}\left[R(a,b) + \gamma \sum_o V^*_\tau(b_{a,o})\right]$, the second term $\|V^*_\tau - V^*\|_\infty$ is bounded by $\gamma^\tau \|V^*_0 - V^*\|$ (see [5]). The remainder of this section states and proves a bound on the first term $\|V_\tau - V^*_\tau\|_\infty$. We prove that [10]'s bound stated for POMDPs holds for DEC-POMDPs. in the remainder of this paper, we state $\varepsilon_B = \max_{b' \in \bar{\triangle}} \min_{b \in B} \|b - b'\|_1$, and $\|r\|_\infty = \max_{s,a} R(s,a)$. First of all, let us prove the following lemma:

LEMMA 2. *The error introduced in $\mathbb{H}_B$-PI when performing one iteration of value function update over B, instead of $\bar{\triangle}$, is bounded by: $\eta \le \frac{\|r\|_\infty}{1-\gamma}\varepsilon_B$*

PROOF. To have an intuition of the proof we provide an illustrative example Figure 3. Let $b' \in \bar{\triangle} \backslash B$ be the belief where $\mathbb{H}_B$-PI makes its worst error in the value function update, and $b \in B$ be the closest ($\mathbb{L}_1$ norm) sampled belief to b'. Let α_y the hyperplane that would be maximal at b', where $y = \{y^i\}$ and $y \notin X$. Let α_x be the hyperplane that is maximal for b', where $x = \{x^i\}$ and $x \in X$. By failing to include α_y in its solution set, $\mathbb{H}_B$-PI makes an error of at most $\alpha_y \cdot b' - \alpha_x \cdot b'$. On the other hand, we know that $\alpha_x \cdot b \ge \alpha_y \cdot b$. This is mainly because in order to remove hyperplane α_y Theorem 2 requires each machine state y^i to be dominated or equal to machine state x^i over belief set B. So,

$$\begin{array}{lll}
\eta & \le & \alpha_y \cdot b' - \alpha_x \cdot b' \\
& = & \alpha_y \cdot b' - \alpha_x \cdot b' + (\alpha_y \cdot b - \alpha_y \cdot b) \quad \text{add zero} \\
& \le & \alpha_y \cdot b' - \alpha_x \cdot b' + \alpha_x \cdot b - \alpha_y \cdot b \quad \alpha_x \text{ maximal at } b' \\
& = & (\alpha_y - \alpha_x) \cdot (b' - b) \quad \text{collect terms} \\
& \le & \|\alpha_y - \alpha_x\|_\infty \|b' - b\|_1 \quad \text{Holder inequality} \\
& \le & \|\alpha_y - \alpha_x\|_\infty \varepsilon_B \quad \text{definition of } \varepsilon_B \\
& \le & \frac{\|r\|_\infty}{1-\gamma}\varepsilon_B \quad \square
\end{array}$$

The remainder of this section states and proves a bound for $\mathbb{H}_B$-PI algorithm.

THEOREM 3. *For any belief set B and any time step τ, the error of the $\mathbb{H}_B$-PI algorithm $\eta_\tau = \|V_\tau - V^*_\tau\|_\infty$ is: $\eta_\tau \le \frac{\|r\|_\infty}{(1-\gamma)^2}\varepsilon_B$.*

PROOF.

$$\begin{array}{lll}
\eta_\tau & = & \|V_\tau - V^*_\tau\|_\infty \quad \text{(definition of } \eta_\tau) \\
& = & \|\mathbb{H}_B V_{\tau-1} - \mathbb{H}V^*_{\tau-1}\| \quad \text{(definition of } \mathbb{H}_B \text{ and } \mathbb{H}) \\
& \le & \|\mathbb{H}_B V_{\tau-1} - \mathbb{H}V_{\tau-1}\|_\infty + \|\mathbb{H}V_{\tau-1} - \mathbb{H}V^*_{\tau-1}\|_\infty \\
& \le & \eta + \|\mathbb{H}V_{\tau-1} - \mathbb{H}V^*_{\tau-1}\|_\infty \quad \text{(definition of } \eta) \\
& \le & \eta + \gamma\|V_{\tau-1} - V^*_{\tau-1}\|_\infty \quad \text{(contraction)} \\
& \le & \eta + \gamma\eta_{\tau-1} \quad \text{(definition of } \eta_{\tau-1}) \\
& \le & \frac{\|r\|_\infty}{(1-\gamma)^2}\varepsilon_B \quad \text{(series sum)} \quad \square
\end{array}$$

Notice that a tighter approximation error bound of $\mathbb{H}_B$-PI can be derived. Indeed, we estimate only the pruning error that incurs during the policy improvement step and ignore the policy

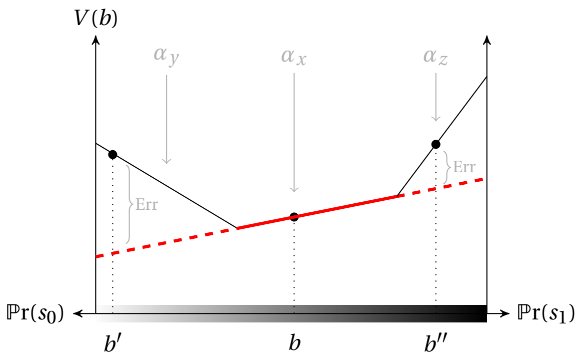

Figure 3: Errors made on a 2-states DEC-POMDP by $\bar{\mathbb{H}}_B$-PI for beliefs b' with respect to their closest belief b.

evaluation step as well as hyperplanes from the earlier updates that are kept for the integrity of the controller.

6. EXPERIMENTAL RESULTS

As memery-bounded algorithms, NLP and BPI, are known to perform better than other approximate solvers, we compare $\bar{\mathbb{H}}_B$-PI. only to NLP and BPI. The comparison is made according to DEC-POMDP benchmarks from the literature: the multi-access broadcast channel (MABC) problem [7], the multi-agent tiger problem [9], the MEETING GRID problem [4], and the COOPERATIVE BOX-PUSHING problem [11]. The COOPERATIVE BOX-PUSHING problem provides an opportunity for testing the scalability of different algorithms. The reader can refer to the references above for an exact specification of the benchmarks. The performances of memory-bounded algorithms where sent by the authors, and run on an Intel(R) Pentium(R) 4 CPU 3.4GHz with 2GB of memory. Our algorithm was run on a Intel Core Duo 2.4GHz with 2GB of memory.

Results. Figure 4 presents our experimental results. For each problem and solver, we report: the value $V^\delta(b_0)$ at the initial belief b_0, the CPU time (in seconds) and the size $|B|$ belief set B used for $\bar{\mathbb{H}}_B$-PI. We depict for each benchmark two graphs. On the right-hand side, we show the runtimes of all solvers over iterations. As performance results of memory-bounded solvers are built based on the solution size, we report on the x-axis the individual controller size. On the other hand, since $\bar{\mathbb{H}}_B$-PI is an iterative algorithm we report on the x-axis the number of iterations performed so far. As a result, we use iteration – that is the label of the x-axis, either for the size of individual controllers or the number of iterations performed so far, depending on the solver. Because all solvers do not have the same x-axis, the reader should mostly focus on the best value obtained by the solver and the amount of time it takes to reach that value. On the left-hand side, we illustrate the value at the initial belief over the iterations. Overall it appears that $\bar{\mathbb{H}}_B$-PI outperforms all NLP and BPI in all tested DEC-POMDP domains and in both computation time and solution quality. For problem of small-size, $\bar{\mathbb{H}}_B$-PI reaches a fixed point quite fast while neither NLP nor BPI were able to even reach the same solution quality for TIGER-A, MEETING-GRID, and COOPERATIVE BOX-PUSHING. As an example, $\bar{\mathbb{H}}_B$-PI takes 11.5 seconds to converge into a DJFSC of value 1.91 for the TIGER-A problem. The best memory-bounded solver, NLP, takes 1059 seconds to find its best joint stochastic controller of value −2.36. For this same problem, BPI could not find a stochastic joint controller of value higher than −52.63, this also explains why it does not appear in the graph. Even more importantly, $\bar{\mathbb{H}}_B$-PI converges into a fixed-point for MABC, TIGER-A, and MEETING-GRID benchmarks. This suggests that it inherited some of the properties of single-agent solvers.

7. CONCLUSION AND FUTURE WORK

We have derived and analyzed the first PI algorithms with provable error bounds. This research also provides interesting new theoretical insights, including the ability to perform DP updates while preserving the decentralization. Even though the experiment results demonstrate impressive improvement over the current standard methods, we are still far from being able to deal with real-world-size DEC-POMDPs. To this end, we are planning to use the new theoretical insights exhibited in this paper as foundations of more scalable algorithms facing some of the bottlenecks of our approach.

8. ADDITIONAL AUTHORS

9. REFERENCES

[1] Christopher Amato, Daniel S. Bernstein, and Shlomo Zilberstein. Optimizing memory-bounded controllers for decentralized POMDPs. In *UAI*, 2007.

[2] Daniel S. Bernstein, Christopher Amato, Eric A. Hansen, and Shlomo Zilberstein. Policy iteration for decentralized control of Markov decision processes. *JAIR*, 34:89–132, 2009.

[3] Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of Markov decision processes. *Math. Oper. Res.*, 27(4), 2002.

[4] Daniel S. Bernstein, Eric A. Hansen, and Shlomo Zilberstein. Bounded policy iteration for decentralized POMDPs. In *IJCAI*, pages 1287–1292, 2005.

[5] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 2005.

[6] Eric A. Hansen. An improved policy iteration algorithm for partially observable mdps. In *NIPS*, 1997.

[7] Eric A. Hansen, Daniel S. Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI*, pages 709–715, 2004.

[8] Shihao Ji, Ronald Parr, Hui Li, Xuejun Liao, and Lawrence Carin. Point-based policy iteration. In *AAAI*, pages 1243–1249, 2007.

[9] Ranjit Nair, Pradeep Varakantham, Milind Tambe, and Makoto Yokoo. Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. In *AAAI*, pages 133–139, 2005.

[10] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *IJCAI*, 2003.

[11] Sven Seuken and Shlomo Zilberstein. Improved Memory-Bounded Dynamic Programming for DEC-POMDPs. In *UAI*, 2007.

[12] Sven Seuken and Shlomo Zilberstein. Formal models and algorithms for decentralized decision making under uncertainty. *JAAMAS*, 17(2):190–250, 2008.

[13] R. D. Smallwood and E. J. Sondik. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, 21(5):1071–1088, 1973.

[14] E. Sondik. The optimal control of partially observable Markov decision processes. Technical report, Standford, 1971.

[15] E. J. Sondik. The optimal control of partially observable Markov decision processes over the infinite horizon : Discounted cost. *Operations Research*, 12:282–304, 1978.

[16] Daniel Szer and François Charpillet. An optimal best-first search algorithm for solving infinite horizon DEC-POMDPs. In *ECML*, pages 389–399, 2005.

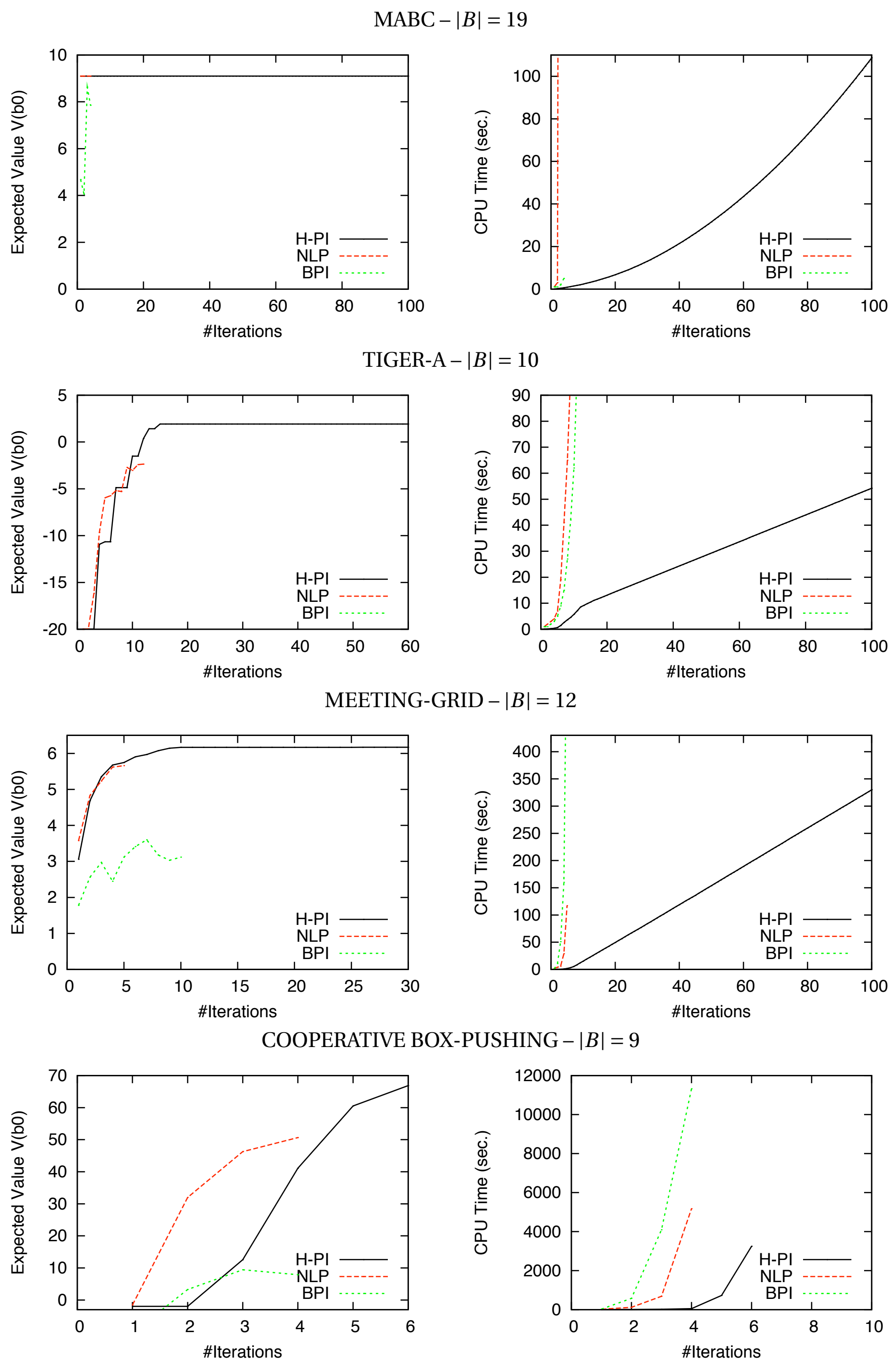

Figure 4: Performance results for DEC-POMDP benchmark problems from the literature.