Toward Error-bounded Algorithms for Infinite-Horizon DEC-POMDPs

Jilles S. Dibangoye
Ecole des Mines de Douai
Douai, France
jilles.dibangoye@mines-douai.fr

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

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

ABSTRACT

Over the past few years, attempts to scale up infinite-horizon DEC-POMDPs with discounted rewards are mainly due to approximate algorithms, but without the theoretical guarantees of their exact counterparts. In contrast, ε-optimal methods have only theoretical significance but are not efficient in practice. In this paper, we introduce an algorithmic framework (β-**PI**) that exploits the scalability of the former while preserving the theoretical properties of the latter. We build upon β-**PI** a family of approximate algorithms that can find (provably) error-bounded solutions in reasonable time. Among this family, **H-PI** uses a branch-and-bound search method that computes a near-optimal solution over distributions over histories experienced by the agents. These distributions often lie near structured, low-dimensional subspace embedded in the high-dimensional sufficient statistic. By planning only on this subspace, **H-PI** successfully solves all tested 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 local information. To formalize and solve such problems,[5, 13] suggest similar models that enable a set of n agents to cooperate in order to control a partially observable Markov decision process (POMDP), namely decentralized partially observable Markov decision process (DEC-POMDP).

Unfortunately, finding either optimal or even near-optimal solutions of such a problem has been shown to be particularly hard [5, 14]. Significant efforts have been devoted to developing near-optimal algorithms for DEC-POMDPs. These algorithms consist in searching in the entire space of all policies [16, 4]. State-of-the-art optimal or near-optimal methods such as **DEC-PI** [4], **DP** [10], **MAA*** [16], **PBDP** [17], or mathematical programs [3, 6] suggest first performing the exhaustive enumeration of all possible policies before they prune dominated ones. This is both an advantage and a liability. On the one hand, it preserves the ability to eventually find an ε-optimal policy, which is a key property. Yet it makes these methods impractical even for small toy problems. This is mainly because they quickly run out of memory.

Cite as: The title of your paper should be written here, J.S. Dibangoye, A.-I. Mouaddib and B. Chaib-draa, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Yolum, Tumer, Stone and Sonenberg (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. XXX–XXX.

To tackle the memory bottleneck, a number of memory-bounded algorithms have been suggested, and proven to be remarkably scalable, but without the theoretical guarantees of their exact counterparts [4, 15, 7, 9, 2]. Memory-bounded algorithms use a fixed amount of memory, *i.e.,* the size of the solution is fixed prior to the execution of the algorithm. Infinite-horizon memory-bounded techniques such as **NLP** and **BPI** rely on mathematical programs that search the best possible policy for a fixed size [1]. On the other hand, finite-horizon memory-bounded methods including **MBDP** [15], **MBDP-OC** [7], **PBIP** [9], **PBIP-IPG** [2] and **CBPB** [11] are mainly point-based algorithms. They compute approximate policies over a bounded number of beliefs[1] by selecting only a few policies for each point. While applying finite-horizon algorithms to infinite-horizon cases is non-trivial, they provide good insights on approximation methods. However, both infinite and finite approaches lack theoretical guarantees on the approximation. So it would seem we are constrained to either solving small toy problems near-optimally, or solving large problems but possibly doing so badly.

In this paper, we introduce an algorithmic framework (β-**PI**) that builds upon both approximate and near-optimal techniques. This provides the ability to preserve the theoretical properties of the former, while exploiting the scalability of the latter. To do so, β-**PI** incorporates the error β the decision-maker can sacrifice at each time step of the execution stage for computational tractability. A theoretical analysis of β-**PI** provides error-bounds on solutions produced by many approximate techniques. We then derive the **H-PI** algorithm that aims at reducing the error produced in approximate algorithms while improving the empirical performances. In order to reduce the error-bound, **H-PI** relies on the concept of distributions over histories, *i.e.,* the sufficient statistic for the selection of a decision rule[2] in general DEC-POMDPs [8] – page 199. By planning over distributions over histories experienced by the agents, **H-PI** considerably tightens the error-bound produced. These distributions often lie near structured, low-dimensional subspace embedded in the high-dimensional sufficient statistic. By maintaining only a single policy-node for each individual history, it circumvents the memory bottleneck. This is achieved by means of a branch-and-bound search method that tracks the best policy for each distribution over histories experienced by the agents. This paper also provides empirical results demonstrating the successful performance of **H-PI** algorithm on all tested benchmarks: outperforming standard algorithms, both in solution time and policy quality.

2. BACKGROUND AND RELATED WORK

[1] A belief is a probability distribution over the underlying states of the system.
[2] A decision rule is a mapping $d\colon H \to A$ from history set to action set.

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

A n-agent DEC-POMDP model can be represented using a tuple $(S, \{A^i\}, \{\Omega^i\}, h_0, P, O, R, \gamma)$, where: S is a finite set of states; A^i denotes a finite set of actions available for agent i, and $A = \otimes_{i=1}^{n} A^i$ is the set of joint actions, where $a = (a^1, \cdots, a^n)$ denotes a joint action; $P(s'|s,a)$ is a Markovian transition function, that 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 $\Omega = \otimes_{i=1}^{n} \Omega^i$ is the set of joint-observations, where $\omega = (\omega^1, \cdots, \omega^n)$ is a joint observation; $O(\omega|a,s')$ is an observation function, that denotes the probability of observing joint observation ω given that joint action a was taken and led to state s'; $R(s,a)$ is a reward function, that denotes the reward signal received when executing action a in state s. The DEC-POMDP model is parameterized by: h_0, the initial joint history, *i.e.,* the team joint action/joint-observation trace. 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 referred as infinite-horizon DEC-POMDPs with discounted rewards.

2.1.1 Sufficient Statistic

A key assumption of DEC-POMDPs is that during the online execution stage the true state of the world could not be sensed exactly and reliably: agents are imperfectly informed about the state of the world to differing degrees.

Given that the state is not directly observable, the agents can instead maintain a complete trace of all joint-observations and all joint-actions they ever executed during the offline planning stage, and use this to select their joint-actions. These joint-action/joint-observation traces are referred to as joint-history experienced by the agents. We formally define $h_\tau^i := (a_0^i, \omega_1^i, a_1^i, \omega_2^i, \cdots, a_{\tau-1}^i, \omega_\tau^i)$, $h_\tau := (h_\tau^1, \cdots, h_\tau^n)$, and $H_\tau := \{h_\tau\}$ to be the history of agent i, the joint history of the team, and the set of histories at time step τ, respectively.

We define the sufficient statistic at step τ, $\mu_\tau \in \triangle H_\tau$ to be a probability distribution of the team over joint-histories H_τ [8]. Furthermore, μ_τ at time step τ is calculated recursively, using only the distribution over histories one time step earlier, $\mu_{\tau-1}$, along with the most recent joint decision rule $d_{\tau-1}\colon H_{\tau-1} \to A$:

$$\mu_\tau(h_\tau) = \mu_{\tau-1}(h_{\tau-1}) \cdot p(h_{\tau-1}, d_{\tau-1}(h_{\tau-1}), \omega_\tau | h_{\tau-1})$$

for all $h_{\tau-1} \in H_{\tau-1}$ and $\omega_\tau \in \Omega$, where joint history h_τ is given by joint history one step earlier, along with its corresponding joint-action and a given joint-observation, *i.e.,* $(h_{\tau-1}, d_{\tau-1}(h_{\tau-1}), \omega_\tau)$. Notice that $p(h,a,\omega|h') = \sum_{s,s'} O(\omega|a,s') P(s'|a,s) \mu_{\tau-1}(h')$ and $\mu_0(h) = 1$ if and only if $h = h_0$. Finally, the distribution over individual history h^i is defined by $\mu^i(h^i) = \sum_h p(h^i|h)\mu(h)$, where $p(h^i|h) = 1$ if and only if there exists h^j such that $h^i h^j = h$, otherwise $p(h^i|h) = 0$. If not all joint histories are reachable, μ_τ yields a positive probability only for reachable histories denoted $\bar{H}_\tau$. Unfortunately, $\bar{H}_\tau$ can get very large as time goes on. More precisely, set of histories increases exponentially with increasing time step $\bar{H}_\tau = \mathcal{O}(|A^i||\Omega^i|^{n\tau})$. For this reason, we want to plan only over a small set of distributions over histories experienced by the agents. These distributions often lie near structured, low-dimensional subspace embedded in the high-dimensional sufficient statistic.

2.1.2 Optimization Criterion

The goal of DEC-POMDP planning is to find a sequence of joint-actions $\{a_0, \cdots, a_\tau\}$ maximizing the expected sum of rewards $\mathbb{E}[\sum_{\tau=0}^{\infty} \gamma^\tau R(s_\tau, a_\tau)]$. Given the initial belief, the goal is to find a joint-policy δ that yields the highest expected reward. Throughout the paper, a policy δ of the team is represented as a deterministic joint-policy graph. That is, a vector $(\delta^1, \delta^2, \cdots, \delta^n)$ of individual policy graphs as illustrated in Figure 1. We note $X := X^1 \times \cdots \times X^n$ the set of joint-policy nodes $x = (x^1, \cdots, x^n)$. A policy for a single agent i is therefore represented as a deterministic policy graph δ^i, where $X^i = \{x^i\}$ denotes a set of policy nodes. Solving such a problem usually relies on successive approximations of the joint-policy graph. After τ consecutive iterations, the solution consists of a set of hyperplanes $\Lambda_\tau = \{v^x\}$, together with the corresponding joint-policy graph δ_τ. The value function at iteration τ can be formulated as:

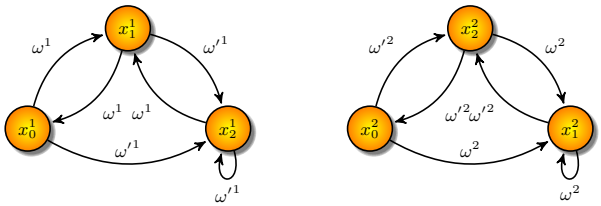

Figure 1: Deterministic policy graphs for two agents.

$$v_\tau(\mu_\tau) = \sup_{\substack{x_{1\tau}^1, x_{2\tau}^1, \cdots, x_{|H_\tau|\tau}^1 \\ \cdots \\ x_{1\tau}^n, x_{2\tau}^n, \cdots, x_{|H_\tau|\tau}^n}} \sum_{k=1}^{|H_\tau|} \mu_\tau(h_{k\tau}) \cdot v^{x_{k\tau}}(h_{k\tau}) \quad (1)$$

where $x_{k\tau} = (x_{k\tau}^1, x_{k\tau}^2, \cdots, x_{k\tau}^n)$ denotes the joint-policy node associated to joint-history $h_{k\tau}$. The resulting set of joint-policy nodes $X_\tau := \{x_{k\tau}\}_{k=1,\cdots,m}$ represents the next step joint-policy graph δ_τ. When joint-policy node x is associated to joint-action a, for all $s \in S$, hyperplane v^x follows:

$$\begin{aligned} v^x(h_\tau) &= \textstyle\sum_s p(s|h_\tau)(R(s,a) + \gamma \sum_\omega v^{a,\omega,x'}(s)) \\ p(s|h_\tau) &= \frac{\sum_{s'} O(\omega|a,s') P(s'|a,s) \mu_{\tau-1}(h_{\tau-1})}{p(h_\tau, a, \omega | h_{\tau-1})} \end{aligned}$$

where $p(s|h_\tau)$ denotes the probability of being in state s given history h_τ, this probability distribution is also referred to as belief state. The estimate value $v^{a,\omega,x'}$ denotes the value of taking joint-action a followed by a joint-observation ω conditional transition to joint-policy node x, and is given by:

$$v^{a,\omega,x'}(s) = \textstyle\sum_{s'} P(s'|s,a) O(\omega|a,s') v^{x'}(s')$$

It is worth noticing that Equation (1) denotes the supremum over all next step joint-policy graphs that selects both: on the one hand, the best[3] hyperplane $v^{x_{k\tau}}$ for each joint-history $h_{k\tau}$; and a single policy-node $x_{k\tau}^i$ for each individual history $h_{k\tau}^i$, on the other hand, thus preserving the ability to control the system in a distributed manner.

Throughout the paper, we will use superscripts either to name agent, *e.g.,* x^i, Ω^i or to distinguish estimate values between joint-policy graphs and joint-policy nodes, *e.g.,* v^δ, v^x, or $v^{a,\omega,x}$. In addition, we use subscripts to indicate time step or iteration, *e.g.,* v_τ, Λ_τ, δ_τ. Finally, except specific indications, $\|\cdot\|$ denotes the Chebyshev norm.

[3]The best hyperplane is not necessarily the maximal hyperplane for a given joint-history, this is why the sup operator is outside the $\sum$, and states the major difference with POMDPs both in terms of complexity and value function expression.

2.2 Related Work

In this section, we discuss near-optimal as well as approximate approaches to solving infinite horizon DEC-POMDPs with discounted rewards.

DEC-PI was the first attempt to compute a near-optimal policy for infinite-horizon DEC-POMDPs with discounted rewards. It builds over a series of exhaustive backups a vector of stochastic policy graphs, one for each agent [4]. However, the number of policy nodes generated by the exhaustive backups is double exponential in the number of iterations. In order to reduce the number of policy nodes generated, **DEC-PI** does pruning by using iterated elimination of dominated policies after each backup, without loosing the ability to eventually converge to a near-optimal policy. Performing this pruning, however, can be expensive, in addition the number of policy nodes still grows exponentially.

To alleviate these problems, we can use a heuristic search technique (referred to as **I-MAA***), which uses a best-first search in the space of deterministic joint-policy graphs with a fixed size. **I-MAA*** prunes dominated joint-policy graphs at earlier construction stages. This is done by calculating a heuristic for the joint-policy graph given known parameters and filling in the remaining parameters one at a time in a best-first fashion. Both **DEC-PI** and **I-MAA*** provide good guarantees on the solution quality, but they do not scale beyond small toy problems. This is mainly due to the explosion in memory. As such, researchers have turned their attention to a family of approximate memory-bounded algorithms.

Thus, a version of **DEC-PI** namely **DEC-BPI** was introduced to keep the policy size bounded over the iterations of the **DEC-PI** algorithm [4]. **DEC-BPI** iterates through the policy nodes of each agent's stochastic policy graph and attempts to find an improvement. A linear program searches for a probability distribution over actions and transitions into the agent's current policy graph that increases the value of the policy graph for any initial belief state and any initial policy node of the other agents' policy graph. If an improvement is discovered, the policy node is updated based on the probability distributions found. Each policy node of each agent is examined in turn and the algorithm terminates when no policy graph can be further improved. While this algorithm allows memory to remain fixed, it provides only a locally optimal solution. Unfortunately, this locally optimal solution can be arbitrarily far from the actual optimal solution.

In an attempt to address some of these problems, a set of optimal policy graphs given a fixed size with nonlinear program was defined in the **NLP** algorithm. Because it is often intractable to solve this **NLP** optimally, a locally optimal solver is used. Unlike **DEC-BPI**, this approach allows initial belief state to be used so smaller policy graphs may be generated and the improvement takes place in one step. While concise policy graphs with high value can be produced, large policy graphs, which may be required for some problems, cannot be produced by the current locally optimal solvers. Even more importantly, these locally optimal solvers do not provide any error-bound on their solutions. So it would seem we are constrained to either solving small toy problems near-optimally, or solving large problems but possibly doing so badly.

3. A NEW NEAR-OPTIMAL ALGORITHM

In this section, we present a new near-optimal algorithm (β-**PI**) for solving infinite-horizon DEC-POMDPs with discounted rewards. β-**PI** has only theoretical significance and is not efficient in practice. However, its framework serves as a foundation to derive methods that are both error-bounded and very efficient in practice, as discussed in the next section (Section 4).

β-**PI** algorithm (Figure 2) consists of a two-step method: the policy-evaluation (step 2); and the policy-improvement (step 3). At each iteration τ, the policy-evaluation estimates the set of hyperplanes Λ_τ of the current joint-policy graph δ_τ, while the policy improvement updates set Λ_τ into set $\Lambda_{\tau+1}$. Thereafter, it transforms the current joint-policy graph δ_τ into an improved one $\delta_{\tau+1}$ through comparison of Λ_τ and $\Lambda_{\tau+1}$. Finally, duplicated, dominated, and unreachable old policy nodes are pruned.

Policy Iteration Algorithm β-PI

1. Set parameters (β, ε) and joint-policy graph δ_0.
2. (*Policy Evaluation*) Obtain Λ_τ by evaluating δ_τ.
3. (*Policy Improvement*) Transform δ_τ to $\delta_{\tau+1}$ through

$$\Lambda_{\tau+1} = (\beta\text{-}\mathbb{H}) \cdot \Lambda_\tau$$

4. If $\|\upsilon_{\tau+1} - \upsilon_\tau\| < \varepsilon(1-\gamma)/2\gamma$, stop and return $\delta_{\tau+1}$. Otherwise set $\tau = \tau + 1$ and return to step 2.

Figure 2: β-PI Algorithm.

The above procedure is similar to classical ε-optimal policy-iteration algorithms [4], when parameter $\beta = 0$. This parameter denotes the decision-maker's preference on the quality of the returned solution. Indeed, β-**PI** is designed to return a solution with error bounded by β when compared to the ε-optimal solution, so as to satisfy the decision-maker's preferences. To this end, β-**PI** replaces the exhaustive backup operator performed in classical policy-iteration algorithms by a new backup operator (β-$\mathbb{H}$) that builds up the improved value function with error bounded by β.

3.1 Backup Operator

A backup operator aims at computing an improved set $\Lambda_{\tau+1} = \{\upsilon^{x'}\}$ given set Λ_τ. Each joint-policy node x' corresponds to a joint-action choice a (resp. a^i) followed by a joint-observation choice ω (resp. ω^i) conditional transition to joint-policy node x (resp. x^i). As a result, one can represent a joint-policy node x' as a set of action-observation-node trios $\{(a^i - \omega^i - x^i)\}_{i=1,\cdots,n}$.

$\Lambda_{\tau+1} = (\beta\text{-}\mathbb{H}) \cdot \Lambda_\tau$

1. $\forall (a^i, \omega^i)$ compute set $X_{\tau+1}^{a^i,\omega^i} = \mathbf{IEDT}(a^i, \omega^i, X_\tau^i)$.
2. $\forall a$, compute set $X_{\tau+1}^a = \bigotimes_{i=1}^n (\bigotimes_{\omega^i \in \Omega^i} X_{\tau+1}^{a^i,\omega^i})$.
3. Compute set $\Lambda_{\tau+1} = \{\upsilon^{x'} \mid x' \in \cup_{a \in A} X_{\tau+1}^a\}$.

Figure 3: Backup Operator β-$\mathbb{H}$.

Following this idea, backup operator β-$\mathbb{H}$, described in Figure 3, prunes those dominated trios, but without actually constructing joint-policy nodes x' exhaustively (step 1). Next, it creates set $X_{\tau+1}^a$ ($\forall a \in A$), a cross-product over agents and individual observations, which includes one trio $(a^i - \omega^i - x^i)$ from each $X_{\tau+1}^{a^i,\omega^i}$ (step 2). Finally, it takes the union of X^a sets and creates the improved value function represented by set $\Lambda_{\tau+1}$ (step 3).

The intuition behind β-$\mathbb{H}$ is that rather than adding all possible trios $A^i \times \Omega^i \times X_\tau^i$ as suggested in classical algorithms [4], we only add trios that would be part of non β-dominated joint-policy node x'. That is, trio $(a^i - \omega^i - x^i)$ is pruned if its value goes up over β by changing policy node x^i by another one for some distribution $\zeta(\cdot)$ of states s; policy node x^j of the other agents; action a^j; and observation ω^j. For the sake of simplicity we use the abbreviation $\rho^j = (s, a^j, \omega^j, x^j)$. Our β-dominance criterion is therefore

formulated as follows: maximize ξ, s.t.: $\forall y^i \in X_{\tau+1}^{a^i,\omega^i} \backslash \{x^i\}$, in

$$\sum\nolimits_{\rho^j} \zeta(\rho^j) v^{a,\omega,x}(s) + \xi + \beta \quad \leq \quad \sum\nolimits_{\rho^j} \zeta(\rho^j) v^{a,\omega,y}(s) \quad (2)$$

where $\sum_{\rho^j} \zeta(\rho^j) = 1$ and $\zeta(\rho^j) \geq 0$ and $a = a^i a^j$, $\omega = \omega^i \omega^j$, $x = x^i x^j$ and $y = y^i x^j$. We provide in Figure 4 an example on how the concept of β-dominance can be used in practice.

Algorithm 1 Iterative Elimination of β-Dominated Trios

```
1: procedure PRUNE(a^i, ω^i, X^i_τ)
2:     Initialize: X^{a^i,ω^i}_{τ+1} ← X^i_τ;
3:     repeat
4:         for i = 1, 2, ··· , n do
5:             for x^i ∈ X^{a^i,ω^i}_{τ+1} do
6:                 Maximize ξ, s.t.: ∀y^i ∈ X^{a^i,ω^i}_{τ+1} \ {x^i}, in Eq. (2).
7:                 if (ξ ≤ 0) then Remove y^i from X^{a,ω^i}_{τ+1};
8:     until no changes occur
```

We are now ready to present the iterative elimination of β-dominated trios algorithm – called **IEDT** Algorithm 1. This algorithm loops over each trio (a^i, ω^i, x^i) and tests whether there exists a probability distribution ζ such that (a^i, ω^i, x^i) β-dominates any other trio, *e.g.,* (a^i, ω^i, y^i). Those trios are kept in sets $X_{\tau+1}^{a^i,\omega^i}$. Then it repeats this procedure for all agents, until no more changes occur. **DEC-PI** and **DP** introduce a similar pruning mechanism namely iterative elimination of dominated policies. However, ours remains fundamentally different. The key difference lies in when this pruning takes place and what we actually prune. **DEC-PI** and **DP** iterative elimination procedures take place after each exhaustive backup, *i.e.,* they first generate all possible policies before they prune dominated ones. **IEDT**, however, takes place before the exhaustive backup, providing the ability to prune all β-dominated trios before we actually build the next step policies. As a result, all policies generated by **IEDT** are non β-dominated policies, as discussed below.

3.2 Theoretical Analysis

This section states and proves the theoretical properties of our β-**PI** algorithm, including: error-bound and convergence. We first show that our β-**PI** algorithm does not prune trios that would be part of a non β-dominated joint-policy node.

LEMMA 1. *Any joint-policy node x' that includes β-dominated trio (a^i, ω^i, x^i) is β-dominated by some probability distribution over some joint-policy nodes.*

PROOF. We will prove this result for 2 agents (although its holds for more), and from the agent 1's perspective. Let trio (a^i, ω^i, x^i) be a β-dominated trio. We now show that any joint-policy node x' that includes (a^i, ω^i, x^i) is β-dominated by some probability distribution over a set of joint-policy nodes $\{y'\}$ that are identical to x' except instead of including (a^i, ω^i, x^i), some probability distribution over trios $\{(a^i, \omega^i, y^i)\}$ is chosen. That is, $\forall s,\ \forall \omega^2, \forall x^2$,

$$v^{a,\omega^1\omega^2,x^1x^2}(s) \quad \leq \quad \sum_{y^1 \neq x^1} p(y^1) \cdot v^{a,\omega^1\omega^2,y^1x^2}(s) + \beta \quad (3)$$

$$\sum_{\omega\in\Omega} v^{a,\omega,x^1x^2}(s) \quad \leq \quad \sum_{\omega\in\Omega} \sum_{y^1 \neq x^1} p(y^1) \cdot v^{a,\omega,y^1x^2}(s) + \beta \quad (4)$$

$$v^{x'}(s) - \beta \quad \leq \quad \sum_{\omega^1, y^1} p(\omega^1, y^1) \left(R(s,a) + \sum_{\omega\in\Omega} v^{a,\omega,y^1x^2}(s) \right) \quad (5)$$

$$v^{x'}(s) - \beta \quad \leq \quad \sum_{y'} p(y') \cdot v^{y'}(s) \quad (6)$$

where $\omega = (\omega^1, \omega^2)$ and $a = (a^1, a^2)$. The inequality (3) results from inequality (2) where trio (a^i, ω^1, x^1) is supposed stochastically β-dominated; in the inequality (4), we consider the sum over all joint observations $\omega \in \Omega$; in inequality 5, we add the immediate reward, and build joint-policy nodes x' and $\{y'\}$. The cross-product between the probability distribution $p(y^1)$ and the uniform probability distribution over Ω^1 enables us to build a probability distribution $p(\omega^1, y^1)$ and we build in a similar way a probability distribution $p(y')$ so that x' is stochastically dominated by $\{y'\}$. This ends the proof. □

We now show that our β-**PI** algorithm returns a near-optimal solution. To do so, we apply the *Banach Fixed-Point Theorem* [12] to prove that β-$\mathbb{H}$ is a contraction mapping on the space $\mathcal{V}$ of bounded functions on S with supremum norm. The proof that β-**PI** is near-optimal follows from properties of norms and contraction mappings.

THEOREM 1. *Our β-**PI** algorithm returns a near-optimal solution for any initial history, with error bounded by $(\varepsilon/2 + \beta/(1-\gamma))$.*

PROOF. First, we prove that β-$\mathbb{H}$ is a contraction mapping on the space of value functions $\mathcal{V}$ for any positive scalar β. Since S is discrete, β-$\mathbb{H}$ maps $\mathcal{V}$ into $\mathcal{V}$.

Let v and u be estimate values of value functions V and U in $\mathcal{V}$ respectively. Fix $h_0 \in H_0$, assume that $(\beta\text{-}\mathbb{H})v(\mu_0) \geq (\beta\text{-}\mathbb{H})u(\mu_0)$, and let $x_{h_0} = \underset{x:\ v^x \in (\beta\text{-}\mathbb{H})V}{\arg\max} \sum_s v^x(h_0)$. Denote a_{h_0} the joint-action associated to joint-policy node x_{h_0}, and $\xi = (\beta\text{-}\mathbb{H})v(\mu_0) - (\beta\text{-}\mathbb{H})u(\mu_0)$. Then, $0 \leq \xi \leq \sum_s (R(s,a_{h_0}) + \gamma \sum_{s',\omega} P(s'|s,a_{h_0}) O(\omega|a_{h_0},s') v^x(s')) p(s|h_0) - \sum_s (R(s,a_{h_0}) + \gamma \sum_{s',\omega} P(s'|s,a_{h_0}) O(\omega|a_{h_0},s') u^y(s')) p(s|h_0)$. Finally,

$$\begin{aligned}
\xi &\leq \gamma \sum_{s,s',\omega} P(s'|s,a_{h_0}) O(\omega|a_{h_0},s') p(s|h_0) [v^x(s') - u^y(s')] \\
&\leq \gamma \sum_{s,s',\omega} P(s'|s,a_{h_0}) O(\omega|a_{h_0},s') p(s|h_0) \|v - u\| \\
&= \gamma \|v - u\|
\end{aligned}$$

Repeating this argument in the case $(\beta\text{-}\mathbb{H})u(\mu_0) \geq (\beta\text{-}\mathbb{H})v(\mu_0)$ implies that $|\beta\text{-}\mathbb{H})v(\mu_0) - (\beta\text{-}\mathbb{H})u(\mu_0)| \leq \gamma \|v - u\|$ for all initial distributions $\mu_0 \in \triangle H_0$. Taking the supremum over μ_0 in the above expression gives the result.

We now able to show that β-**PI** returns a joint-policy graph δ that is near-optimal. Suppose that $\|v_{\tau+1} - v_\tau\| \leq \varepsilon(1-\gamma)/2\gamma$ holds for some iteration. Then, the overall error in β-**PI** is bounded by $\|v^\delta - v_{\tau+1}\| + \|v_{\tau+1} - v^*\|$. Since δ is a fixed point of $(\beta\text{-}\mathbb{H})$, the first expression is bounded as follows: $\|v^\delta - v_{\tau+1}\|$

$$\begin{aligned}
&= \|(\beta\text{-}\mathbb{H}) \cdot v^\delta - v_{\tau+1}\| \\
&\leq \|(\beta\text{-}\mathbb{H}) \cdot v^\delta - (\beta\text{-}\mathbb{H}) \cdot v_{\tau+1}\| + \|(\beta\text{-}\mathbb{H}) \cdot v_{\tau+1} - v_{\tau+1}\| \\
&\leq \gamma \|v^\delta - v_{\tau+1}\| + \|(\beta\text{-}\mathbb{H}) \cdot v_{\tau+1} - (\beta\text{-}\mathbb{H}) \cdot v_\tau\| \\
&\leq \gamma \|v^\delta - v_{\tau+1}\| + \gamma \|v_{\tau+1} - v_\tau\|
\end{aligned}$$

where inequalities follow because $(\beta\text{-}\mathbb{H})$ is a contraction mapping on $\mathcal{V}$. Rearranging terms yields:

$$\|v^\delta - v_{\tau+1}\| \leq \frac{\gamma}{1-\gamma} \|v_{\tau+1} - v_\tau\|.$$

Then, the second expression follows because (0-$\mathbb{H}$) and (β-$\mathbb{H}$) are contraction mappings on $\mathcal{V}$: $\|v_{\tau+1} - v^*\|$

$$\begin{aligned}
&\leq \|(\beta\text{-}\mathbb{H})v_{\tau+1} - (0\text{-}\mathbb{H})v_{\tau+1}\| + \|(0\text{-}\mathbb{H})v_{\tau+1} - v^*\| \\
&\leq \beta + \|(0\text{-}\mathbb{H})v_{\tau+1} - (0\text{-}\mathbb{H})v^*\| \\
&\leq \beta + \gamma \|v_{\tau+1} - v^*\|
\end{aligned}$$

Rearranging terms yields: $\|v_{\tau+1} - v^*\| \leq \beta/(1-\gamma)$. Thus when $\|v_{\tau+1} - v_\tau\| \leq \varepsilon(1-\gamma)/2\gamma$ holds, the first expression is bounded by ε and the second expression is bounded by $\beta/(1-\gamma)$, so that the error produced by β-**PI** is bounded by $\varepsilon/2 + \beta/(1-\gamma)$. □

To better understand the significance of the error-bound in β-**PI**, let's consider its terms. The first term $\varepsilon/2$ denotes the error produced by the stopping criterion in β-**PI** algorithm in Figure 2, step 4. This criterion stops the algorithm before a fixed point of β-$\mathbb{H}$ has been found. It is required when optimal joint-policies do not exist, so we seek ε-optimal joint-policies for $\beta = 0$. This criterion also guarantees that β-**PI** terminates after a finite number of iterations. The second term $\beta/(1-\gamma)$ defines the error the decision-maker's preference produced by pruning all β-dominated hyperplanes. Decreasing the decision-maker's parameter β reduces the error-bound and increases the solution size, but it is usually worthwhile to avoid an explosion of the solution size.

Unfortunately, it is worth noticing that the number of preserved hyperplanes in β-**PI** would be very large in many practical cases. This is mainly because it is likely that there exists a probability distribution for which many hyperplanes β-dominate any other. Indeed, there are infinitely many possible probability distributions. In the next section, we provide two enhancements that overtake the limitations of β-**PI** to scale up while preserving its ability to bound the error produced.

4. ERROR-BOUNDED ALGORITHMS

First, we want to plan only over distributions of histories experienced by the agents $B := \{\mu\}$ as a means of reducing the infinitely many possible probability distributions considered into β-**PI**. Then, we want to arbitrarily increase parameter β such that at some point only one policy node will be preserved for each individual history as a means of reducing the solution size.

Thereafter, those distributions can be used to build non β-dominated hyperplanes at each iteration of β-**PI** algorithm. In particular, the linear program (inequality 2) can be replaced by a series of comparisons over joint-histories h where distributions $\mu \in B$ is positive, without affecting the ability to find a near-optimal solution with respect to B. More formally, if inequality

$$v^{a,\omega^i\omega^j,x^ix^j}(h) \quad \leq \quad v^{a,\omega^i\omega^j,y^ix^j}(h) + \beta$$

holds for any (ω^j, x^j), then trio (a^i, ω^i, x^i) is β-dominated by trio (a^i, ω^i, y^i) for h.

To better understand the pruning procedure using joint-histories, consider the example illustrated in Figure 4. This figure shows the steps of pruning β-dominated trios for a problem with 2 agents; 2 individual observations $\{\omega^i, \omega'^i\}$ and policy-nodes $\{x^i, x'^i\}$; a joint-history $h = h^1h^2$; joint-action $a = a^1a^2$, and $\beta = 0.9$. The set of trios are represented in a form of a bayesian game Figure (4.**A**). Figure (4.**B**) illustrates the first pruning of β-dominated trios for each agent (red lines). As an example, trio (a^1, ω^1, x^1) is pruned since it is β-dominated by trio (a^1, ω^1, x'^1). The pruning process continues until no more pruning occurs. Figure (4.**D**) shows that for joint-history h, joint-action a and $\beta = 0.9$, there is only one possible non β-dominated hyperplane (each agent has only one possible policy node for each observation – a single policy node for each individual history). This remark is crucial to bound the error produced by algorithms that keep only one hyperplane for each joint-history h or its corresponding belief state $p(\cdot|h)$, such as point-based solvers **MBDP** and **PBIP**. That is, there exists a possible large scalar β_B such that there is only one non β_B-dominated hyperplane for each joint-history h.

The next section presents **H-PI**, which provides an efficient and scalable derivation of β-**PI**, while preserving the ability to bound the error produced.

4.1 Heuristic PI Algorithm

The heuristic policy-iteration (**H-PI**) algorithm replaces backup operator β-$\mathbb{H}$ in β-**PI** by a more scalable backup operator denoted β-$\mathbb{H}_B$. This operator performs the backup only over a set of distributions over histories $\mu \in B$, by means of a branch-and-bound search in the space of non β-dominated policy nodes.

As **H-PI** proceeds in the same way for every iteration, and each distribution $\mu \in B$, we therefore restrict our attention to the following problem that occurs at each iteration and for each μ: the problem of assigning trio (a^i, ω^i, x^i) for each individual history h^i where $\mu^i(h^i) > 0$, and this for every individual observation $\omega^i \in \Omega^i$ and all agents $i = 1, 2, \cdots, n$, and such that the resulting joint-policy nodes $\{x_h\}_{h:\ \mu(h)>0}$ are the best possible. That is, the corresponding set of hyperplanes $\{v^{x_h}\}_{h:\ \mu(h)>0}$ β-dominates any other at μ. More formally, β-$\mathbb{H}_B$ computes $\{v^{x_h}\}_{h:\ \mu(h)>0}$, such that for any other set of hyperplanes $\{v^{x'_h}\}_h$ the following holds: $\sum_h \mu(h)v^{x_h}(h) + \beta \geq \sum_h \mu(h)v^{x'_h}(h)$.

The idea behind **H-PI** is to build a search tree in which nodes θ are sets of partially specified mappings $\{(d^i, \sigma^i)\}_i$, where $d^i\colon H^i \to A^i$ is a mapping from individual history set $H^i = \{h^i | \mu^i(h^i) > 0\}$ to individual action set A^i; and $\sigma^i\colon H^i \times \Omega^i \to \cup_{a^i} X^{a^i,\omega^i}$ is a mapping from pairs of individual history and observation to non β-dominated policy nodes.

Notice that by assigning a value to a variable we often constrain the possible assignments of the other variables. To better understand this, let's consider the assignment of value a^i to variable $d^i(h^i)$, as a result variables $\sigma^i(h^i, \omega^i)$ are constrained to choose their values in X^{a^i,ω^i}. Thereafter, it is likely that trios that were non β-dominated before the assignment become β-dominated after. For this reason, **H-PI** interleaves each search node expansion step with an iterative elimination of β-dominated trios for each expanded nodes in the search tree. This provides **H-PI**'s first pruning mechanism. The second one prunes nodes based on upper and lower bounds.

Algorithm 2 Heuristic Backup Operator

```
1:  procedure β-H_B(μ)
2:      Initialize: Incumbent := v(μ); Live := {θ_0}
3:      repeat
4:          Select θ_k ∈ Live with the highest v̂(θ_k, μ)
5:          Live := Live \ {θ_k}
6:          Branch on θ_k generating θ_{k_1}, ··· , θ_{k_m}
7:          for 1 ≤ p ≤ m do
8:              if v̂(θ_{k_p}, μ) > Incumbent +β then
9:                  if θ_{k_p} is completely defined then
10:                     Incumbent := v̂(θ_{k_p}, μ)
11:                     Solution := θ_{k_p}
12:                 else Live := Live ∪ {θ_{k_p}}
13:     until Live = ∅
14:     return Solution
```

We define the upper-bound based on the decomposition of the exact estimate into two estimates. The first estimate, $G(\theta, \mu)$, is the exact estimate coming from variables where θ is constrained[4]. The second estimate, $H(\theta, \mu)$, is the upper-bound value coming from variables where θ is not constrained. That is, $\hat{v}(\theta, \mu) = G(\theta, \mu) +$

[4]A partially specified mapping $\phi\colon \mathcal{X} \to \mathcal{Y}$ is said to be constrained at $x \in \mathcal{X}$ if $\phi(x)$ has been assigned a value $y \in \mathcal{Y}$, otherwise it is said to be non constrained.

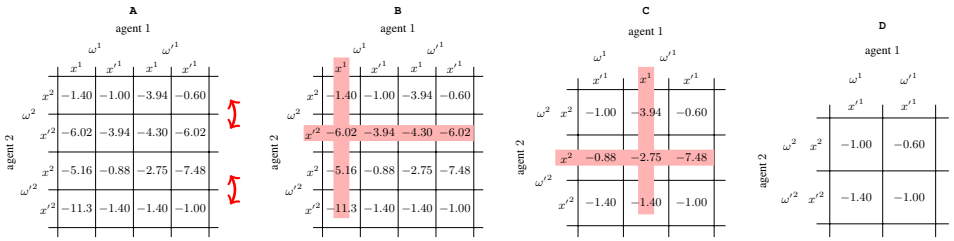

Figure 4: Illustration of the β-dominance criterion, where $\beta = .9$.

$H(\theta, \mu)$, where:

$$G(\theta, \mu) = \sum_h R(h, d(h)) + \gamma \sum_{(h,\omega)} \mu(h) \upsilon^{d(h),\omega,\sigma(h,\omega)}(h)$$

where $d(h)$ and $\sigma(h, \omega)$ are constrained,

$$H(\theta, \mu) = \sum_h \max_a R(h, a) + \gamma \sum_{(h,\omega)} \mu(h) \max_{\upsilon^{a,\omega,x}} \upsilon^{a,\omega,x}(h)$$

where $d(h)$ and $\sigma(h, \omega)$ are not constrained. Notice that $R(h, d(h))$ is given by $\sum_s \mu(h)p(s|h)R(s, d(h))$.

H-PI search starts with a pool of live nodes with a partially specified mapping θ, where none of the variables are specified, see Algorithm 2. Moreover, the value hereof is used as the value (called incumbent) of the current best solution, (line 2). At each iteration of the search, a node θ that yields the highest upper-bound is selected for exploration from the pool of live nodes, (lines 4-5). Then, a branching is performed: two or more children of the node are constructed through the specification of a single variable, (line 6). Furthermore, for each of the generated child nodes θ_k, the upper-bound is computed. In this case, the current node corresponds to a completely specified node, its upper-bound is its exact value at μ, the value hereof is compared to the incumbent, and the best solution and its value are kept, (lines 8-11). If its upper-bound is not better than the incumbent, the node is discarded, since no completely specified 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 12). When the search tree has been completely explored, the algorithm starts a new search tree with a new distribution over histories μ, until all have been processed, and this at each iteration.

H-PI may be considered as an extension and generalization of either near-optimal search methods such as **I-MAA***, or point-based search techniques for solving finite-horizon DEC-POMDPs including **MBDP**, **PBIP**. Indeed, **H-PI** is designed to provide error-bounds on the solution produced as does near-optimal methods. **H-PI** meets this requirement either by planning only other a small set of distributions μ or by using parameter β, or doing both. In addition, **H-PI** is able to scale up through the selection of a small set of distributions μ, and by planning only over a small number of histories among those where $\mu(h) > 0$. In particular, when we plan separately over histories h where $\mu(h) > 0$, we actually perform a point-based search method as does **MBDP**, **PBIP**. Even within the latter case, **H-PI** remains fundamentally different with respect to other point-based search methods. The key difference lies in how the heuristic function is computed. While finite-horizon DEC-POMDP heuristic functions are all based only on the current state of assignments of values to variables, **H-PI** performs an additional step of iterative elimination of β-dominated trios after each node expansion step, thus tightening its heuristic function. The following provide theoretical guarantees on the solution produced by **H-PI**.

4.2 Convergence and Error-bound

For any set of distributions over histories B and iteration τ, **H-PI** produces an estimate υ_τ with the corresponding set of hyperplanes Λ_τ. The error between υ_τ and the true value function υ^*_τ is bounded. The bound depends on four parameters: the density ε_B of the set of distributions over histories B, where ε_B is the maximum distance from any legal distribution μ to B, that is: $\varepsilon_B = \max_{\mu' \in \triangle\bar{H}} \min_{\mu \in B} \|\mu' - \mu\|_1$; the distance β_B between hyperplanes that compose υ_τ, where β_B is the maximum Chebyshev distance of any pair or hyperplanes into Λ_τ, that is: $\beta_B = \max_{\upsilon^x, \upsilon^y \in \Lambda_\tau} \|\upsilon^x - \upsilon^y\|$; the probability $\mu_B = 1 - \min_\tau \sum_{h \in H_\tau} \mu_\tau(h)$ that a history is visited during the online execution stage, but not taken into account during the offline planning stage; and the Chebyshev distance $\|r\| = \max_{s,a} |R(s, a)|$ over the rewards $R(s, a)$, which defines maximum possible rewards that occur after a one step decision.

That is, by keeping all non-dominated hyperplanes over a denser sampling of distribution set $\triangle\bar{H}$, υ_τ converges to υ^*_τ, the true value function. Cutting off **H-PI** iterations at any sufficiently large time step, we know that the divergence between υ_τ and the optimal value function υ^* is bounded. The following lemma states and proves a bound on the error $\|(\beta_B\text{-}\mathbb{H}_B)\upsilon_\tau - (0\text{-}\mathbb{H})\upsilon_\tau\|$ produced by one application of the backup operator β_B-$\mathbb{H}_B$.

LEMMA 2. *The error η_{prune} produced by (β_B-$\mathbb{H}_B$) when performing the value function backup over B instead of $\triangle\bar{H}$, is bounded by: $\eta_{prune} \leq \mu_B \cdot (\beta_B + \varepsilon_B \|r\|/(1-\gamma))$.*

PROOF. First, we note that applying a similar argument to that used to derive that β-$\mathbb{H}$ is a contraction mapping, we prove that β_B-$\mathbb{H}_B$ is also a contraction mapping on $\mathcal{V}$. Let υ be a value function in $\mathcal{V}$, and $(0\text{-}\mathbb{H}_B)$ be the backup operator that plans only over distributions $\mu \in B$ but keeps all non dominated hyperplanes for each distribution $\mu \in B$. Using the triangle inequality, we know that the error $\|(\beta_B\text{-}\mathbb{H}_B)\upsilon - (0\text{-}\mathbb{H})\upsilon\|$ produced by β_B-$\mathbb{H}_B$ is bounded by $\|(\beta_B\text{-}\mathbb{H}_B)\upsilon - (0\text{-}\mathbb{H}_B)\upsilon\| + \|(0\text{-}\mathbb{H}_B)\upsilon - (0\text{-}\mathbb{H})\upsilon\|$. We thus propose a two-fold step method that bounds the two expressions above.

On the one hand, we establish the error $\phi_1 = \|(\beta_B\text{-}\mathbb{H}_B)\upsilon - (0\text{-}\mathbb{H}_B)\upsilon\|$ made by preserving only one non β-dominated policy node for each individual history. Let h be a joint history where β_B-$\mathbb{H}_B$ makes it worst error. This is achieved by pruning away policy-node x^i and hyperplane $\upsilon^{x^i x^j}$. Let $\upsilon^{x^i_h x^j_h}$ be the hyperplane that is maximal for h. By pruning $\upsilon^{x^i x^j}$, β_B-$\mathbb{H}_B$ makes an error of at most $\mu(h)[\upsilon^{x^i x^j}(h) - \upsilon^{x^i_h x^j}(h)]$. Furthermore, we know that

$v^{x^i x^j}(h) \leq v^{x^i_h x^j_h}(h)$. Therefore, ϕ_1

$$
\begin{aligned}
&\leq \mu(h)[v^{x^i x^j}(h) - v^{x^i_h x^j}(h)] && (7)\\
&= \mu(h)[v^{x^i x^j}(h) - v^{x^i_h x^j}(h) + (v^{x^i x^j}(h) - v^{x^i x^j}(h))] && (8)\\
&\leq \mu(h)[v^{x^i x^j}(h) - v^{x^i_h x^j}(h) + v^{x^i_h x^j_h}(h) - v^{x^i x^j}(h)] && (9)\\
&= \mu(h)[v^{x^i_h x^j_h} - v^{x^i_h x^j}] \cdot p(h) && (10)\\
&\leq \|v^{x^i_h x^j_h} - v^{x^i_h x^j}\| \cdot \mu(h) && (11)\\
&\leq \beta_B \cdot \mu_B && (12)
\end{aligned}
$$

The equation (8) results from adding zero $(v^{x^i x^j}(h) - v^{x^i x^j}(h))$ to equation (7). In inequality (9), we replace the third expression $v^{x^i x^j}$ on the right hand side by $v^{x^i_h x^j_h}$, since $v^{x^i_h x^j_h}$ is maximal for h. Rearranging terms in equation (9) yields equation (10) where $p(h)$ is the matrix form of $p(s|h)$. Applying the Chebyshev norm and the definition of β_B result in inequalities (11) and (12), respectively.

On the other hand, we establish the error $\phi_2 = \|(0\text{-}\mathbb{H}_B)v - (0\text{-}\mathbb{H})v\|$ produced by $(0\text{-}\mathbb{H}_B)$ by planning only over B instead of $\triangle\bar{H}$. Let $\mu' \in \triangle\bar{H}\backslash B$ be the distribution where β_B-$\mathbb{H}_B$ makes its worst error, and $\mu \in B$ be the closest sampled distribution to μ'. Let u be the value function that would be maximal at μ'. Let v be the value function that is maximal at h. By failing to include hyperplanes that compose u in its solution set, $(0\text{-}\mathbb{H}_B)$ makes an error of at most $u(\mu') - v(\mu')$. In addition, we know that $v(\mu) \geq u(\mu)$. So, ϕ_2

$$
\begin{aligned}
&\leq u(\mu') - v(\mu') && (13)\\
&= u(\mu') - v(\mu') + (u(\mu) - u(\mu)) && (14)\\
&\leq u(\mu') - v(\mu') + v(\mu) - u(\mu) && (15)\\
&= (u - v) \cdot (\mu' - \mu) && (16)\\
&\leq \|u - v\| \cdot \|\mu' - \mu\|_1 \cdot \mu_B && (17)\\
&\leq \varepsilon_B \cdot \mu_B \cdot \|r\|/(1-\gamma) && (18)
\end{aligned}
$$

The equation (14) results from adding zero $(u(\mu) - u(\mu))$. In inequality (15), we replace the third expression $u(\mu)$ on the right hand side by $v(\mu)$, since v is maximal at μ. Rearranging terms in equation (15) yields equation (16). Inequality (17) follows from Hölder inequality and inequality (18) results from the definition of ε_B. This ends the proof. □

THEOREM 2. *For any distribution set B and any iteration τ, the error of **H-PI** algorithm, $\|v_\tau - v^*\|$, is bounded by: $\eta_\tau \leq \varepsilon/2 + \left(\beta_B/(1-\gamma) + \|r\|\varepsilon_B/(1-\gamma)^2\right)\mu_B$.*

PROOF. The overall error η_τ in **H-PI** at iteration τ is bounded by $\|v_\tau - v^*_\tau\| + \|v^*_\tau - v^*\|$. Because β_B-$\mathbb{H}_B$ is a contraction mapping, when the stooping criterion $\|v_\tau - v_{\tau-1}\| \leq \varepsilon(1-\gamma)/\gamma$ holds, the second term $\|v^*_\tau - v^*\|$ is bounded by $\varepsilon/2$. The remainder of this proof states and demonstrates a bound on the first term $\eta_\tau = \|v_\tau - v^*_\tau\|$ as follows: $\eta_\tau = \|(\beta_B\text{-}\mathbb{H}_B)v_{\tau-1} - (0\text{-}\mathbb{H})v^*_{\tau-1}\|$

$$\leq \|(\beta_B\text{-}\mathbb{H}_B)v_{\tau-1} - (0\text{-}\mathbb{H})v_{\tau-1}\| + \|(0\text{-}\mathbb{H})v_{\tau-1} - (0\text{-}\mathbb{H})v^*_{\tau-1}\|$$

This follows from the definition of backup operators β_B-$\mathbb{H}_B$ and 0-$\mathbb{H}$, as well as the norm properties. We note that the first term on the right hand side of the last inequality is in fact error estimate $\eta_{\texttt{prune}}$. Moreover, as 0-$\mathbb{H}$ is a contraction mapping, the second term on the right hand side of the last inequality is bounded by $\gamma\|v_{\tau-1} - v^*_{\tau-1}\|$. Replacing these terms yields:

$$\eta_\tau \leq \eta_{\texttt{prune}} + \gamma\|v_{\tau-1} - v^*_{\tau-1}\| \quad (19)$$

Then, the error-bound follows as a consequence of Lemma 2, the definition of $\eta_{\tau-1} = \|v_{\tau-1} - v^*_{\tau-1}\|$ and series sum properties:

$$
\begin{aligned}
\eta_\tau &\leq \eta_{\texttt{prune}} + \gamma\eta_{\tau-1}\\
&\leq \left(\beta_B + \tfrac{\|r\|\varepsilon_B}{1-\gamma}\right)\cdot\mu_B + \gamma\eta_{\tau-1}\\
&\leq \left(\tfrac{\beta_B}{1-\gamma} + \tfrac{\|r\|\varepsilon_B}{(1-\gamma)^2}\right)\mu_B
\end{aligned}
$$

This ends the proof. □

This result is rather intuitive. Indeed, the error produced by the **H-PI** relies on three terms. The first $\varepsilon/2$ denotes the error produced by cutting off **H-PI** iterations when the stopping criterion is reached. The second term $\beta_B/(1-\gamma)$ represents the error produced by adding only non β-dominated hyperplanes – and in some case only a single hyperplane for each joint history. In other words, by pruning all β_B-dominated hyperplanes for each joint history. The last term $\varepsilon_B\|r\|/(1-\gamma)^2$ illustrates the error produced by planning only over a small set B. The overall error states the relationship between exact **PI**, β-**PI** and **H-PI** algorithms.

This result synthesizes error-bounds for policy iteration algorithms with respect to three criteria: the backup operator used; the distribution set, and the pruning criterion. This general error-bound can be used to bound the error produced by any algorithm designed within β-**PI**'s algorithmic framework. In particular, when we plan only over a single joint history h at a time using **H-PI** – as does point-based algorithms including **MBDP** [15], **MBDP-OC** [7], **PBIP** [9], and **PBIP-IPG** [2], the error is bounded by $\varepsilon/2 + \frac{\beta_B}{(1-\gamma)} + \frac{\|r\|\varepsilon_B}{(1-\gamma)^2}\mu_B$. However, if we plan over the entire distribution μ, **H-PI** yields a tighter error-bound, *i.e.,* $\varepsilon/2 + \frac{\|r\|}{(1-\gamma)^2}\cdot\varepsilon_B\cdot\mu_B$.

This error-bound also suggests that **H-PI** can tightens the error even more when its distributions set B is uniformly dense in the set of reachable distributions $\triangle\bar{H}$. Selecting the best distribution set in this sense would require the generation of all possible distributions given all possible decision rule and the distributions at hand. As it current stands, we do not address this problem. The selection of our distribution set, is based on trajectories of distributions. We create trajectories based on the current value function. Each such trajectory starts with the initial distribution μ_0, we then executes the greedy decision rule specified by the current value function, and finally select the successor distribution.

5. EMPIRICAL EVALUATIONS

We now evaluate the performance of **H-PI** in comparison with other recent approximate solvers, such as **NLP** and **BPI**. Experiments have been run on Intel Core Duo 1.83GHz CPU processor with 1Gb main memory.

5.1 Results

We begin by demonstrating the advantage of **H-PI** with respect to **NLP** and **BPI**. As we can see in graphs in Figure 5, **H-PI** outperforms **NLP** and **BPI** in all tested DEC-POMDP domains and in both computation time and solution quality.

As we explain above, **H-PI** plans only over a small set of selected distributions experienced by the agents during the offline planning stage. Such distributions often lie near a structured, low-dimensional subspace as illustrated in Table 1. For example, in the boxpushing domain, we only consider 6 distributions since the domain is very structured – more precisely often there is only a single possible next observation for a given history, this considerably limits the number of possible next distributions.While **NLP** and **BPI** compute the policy based on a continuum, by planning over this low-dimensional subspace, **H-PI** saves considerable computational efforts – thus increasing its ability to find good solutions

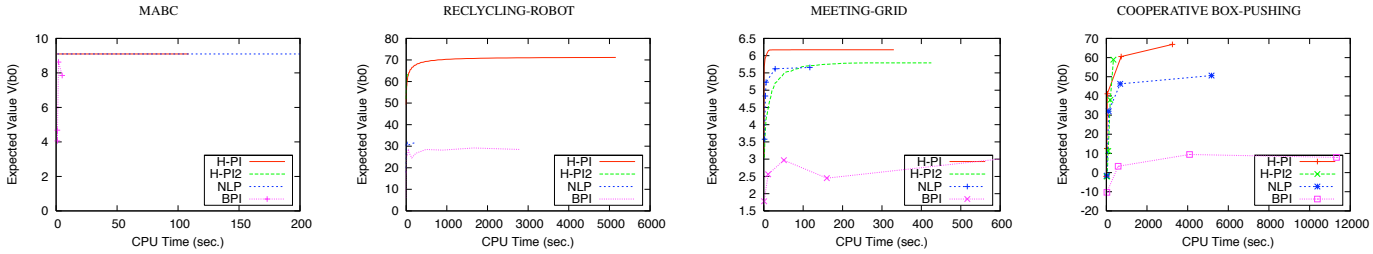

Figure 5: Performance results for DEC-POMDP benchmark problems from the literature

very quickly. Furthermore, it builds the best possible joint-policy graph – assigning a single policy node for each individual history. This tightens the size of the solution produced. Notice, however, that the size of the solution is not bounded by B. Indeed, in the infinite-horizon case the policy nodes are interconnected – that is by keeping policy node x we also keep policy nodes that are reachable starting from x. Finally, because of all its enhancements, **H-PI** does not need to bound the size of the solution produced – it is able to provide larger policy graphs for problems that require such policies so as to achieve reasonable performances. For example, in the reclycing robot domain, **H-PI** produces twice the expected value produced using either **NLP** or **BPI** but it requires policies that are 250 times larger than those in either **NLP** or **BPI**. It is worthwhile to notice that policy graphs produced by **NLP** and **BPI** are stochastic – thus even though the number of policy nodes is reduced, the equivalent deterministic policy graph would be much larger.

We continue to study the performance of **H-PI** with respect to the distribution set B. When we plan other belief states corresponding to histories where $\mu(h) > 0$, **H-PI** is referred to as **H-PI**$_2$, otherwise we use **H-PI**. When evaluating the performance of **H-PI** in comparison with **H-PI**$_2$ – see graphs Figure 5 and Table 1, we note that **H-PI**$_2$ is faster but keeps too much joint policy nodes – this limits its performances in comparison to **H-PI**. This is mainly because, **H-PI**$_2$ often keeps many policy nodes of each individual history, while **H-PI** only keeps a single one. Moreover, as we already discussed, by planning over belief states rather than distributions over histories the error-bound is larger. This explains why the expected value produced by **H-PI** is always superior to the one produced using **H-PI**$_2$. However, by increasing the number of belief states considered we may increase the expected value. As illustrated in Figure 6, as the number of belief states grows the solution quality improves and the computation time also grows (and vice versa). Even more importantly, at some point increasing the number of belief states do not provide significant improvement in the solution quality. These observations support the theoretical results on the error produced by **H-PI**, that is a denser sampling of the set $\triangle\bar{H}$ produces more distributions and results in a tighter error bound. It also highlights the impetus of using a sampling method that selects good distributions or belief states.

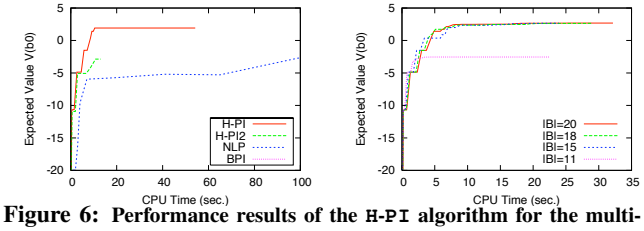

Figure 6: Performance results of the H-PI algorithm for the multi-agent tiger domain, and different belief space sizes.

6. CONCLUSION

Method	$\|B\|$	$\|\Lambda\|$	Method	$\|B\|$	$\|\Lambda\|$	Method	$\|B\|$	$\|\Lambda\|$
TIGER								
H-PI	29	552	**H-PI**$_2$	23	1476	**NLP** & **BPI**	–	4
RECLYCLING-ROBOT								
H-PI	45	1020	**H-PI**$_2$	70	2256	**NLP** & **BPI**	–	4
MEETING-GRID								
H-PI	12	784	**H-PI**$_2$	22	1672	**NLP** & **BPI**	–	4
COOPERATIVE BOX-PUSHING								
H-PI	6	272	**H-PI**$_2$	9	598	**NLP** & **BPI**	–	4
MABC								
H-PI	10	2	**H-PI**$_2$	59	3	**NLP** & **BPI**	–	4

Table 1: Performance measurements on DEC-POMDP domains.

We have introduced a new algorithmic framework (β-**PI**) that exploits the scalability of the approximate methods while preserving the theoretical properties of the near-optimal techniques. In particular, it provides the ability to bound the error produced when we approximate the solution using the sufficient statistic in general DEC-POMDPs. We introduce a heuristic derivation of β-**PI**, namely **H-PI**. We have demonstrated how **H-PI** outperforms state-of-the-art infinite-horizon DEC-POMDP solvers in all tested domains.

In this paper we identify the general requirements from a β-**PI** solver, and suggested a possible implementation for DEC-POMDPs. In the future, we will investigate the integration of methods for the selection of good distributions. We also intend to apply β-**PI** to factored domains such as *fire-fighting* or *network distributed sensors* [13], reducing the dimensionality of the sufficient statistic – thus enabling us to scale to even larger domains.

7. REFERENCES

[1] C. Amato, D. S. Bernstein, and S. Zilberstein. Optimizing memory-bounded controllers for decentralized pomdps. In *UAI*, 2007.

[2] C. Amato, J. S. Dibangoye, and S. Zilberstein. Incremental policy generation for finite-horizon dec-POMDPs. *in Proceedings of the Ninetieth International Conference on Automated Planning and Scheduling*, 2009.

[3] R. Aras and A. Dutech. An investigation into Mathematical Programming for Finite Horizon Decentralized POMDPs. *Journal of Artificial Intelligence Research*, 2010. to appear.

[4] D. S. Bernstein, C. Amato, E. A. Hansen, and S. Zilberstein. Policy iteration for decentralized control of Markov Decision Processes. *JAIR*, 34:89–132, 2009.

[5] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of Markov Decision Processes. *Math. Oper. Res.*, 27(4), 2002.

[6] A. Boularias and B. Chaib-draa. Exact dynamic programming for decentralized POMDPs with lossless policy compression. In *ICAPS*, pages 20–27, 2008.

[7] A. Carlin and S. Zilberstein. Value-based observation compression for DEC-POMDPs. In *AAMAS*, 2008.

[8] J. S. Dibangoye. *Contribution à la résolution des problèmes décisionnels de Markov centralisés et décentralisés: algorithmes et théorie*. PhD thesis, UCBN – Laval University., Québec Q.C., Canada., December 2009.

[9] J. S. Dibangoye, A.-I. Mouaddib, and B. Chaib-draa. Point-based incremental pruning heuristic for solving finite-horizon DEC-POMDPs. *in AAMAS*, 2009.

[10] E. A. Hansen, D. S. Bernstein, and S. Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI*,

pages 709–715, 2004.

[11] A. Kumar and S. Zilberstein. Point-based backup for decentralized pomdps: complexity and new algorithms. In *AAMAS*, pages 1315–1322, 2010.

[12] M. L. Putterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, New York, NY, 1994.

[13] D. V. Pynadath and M. Tambe. Multiagent teamwork: analyzing the optimality and complexity of key theories and models. In *AAMAS*, pages 873–880, 2002.

[14] Z. Rabinovich, C. V. Goldman, and J. S. Rosenschein. The complexity of multiagent systems: the price of silence. In *AAMAS*, pages 1102–1103, 2003.

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

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

[17] D. Szer and F. Charpillet. Point-based dynamic programming for DEC-POMDPs. In *AAAI*, pages 16–20, July 2006.