Multi-Agent Planning with Baseline Regret Minimization

Feng Wu[†] **Shlomo Zilberstein**[‡] **Xiaoping Chen**[†]
[†]School of Computer Science and Technology, University of Science and Technology of China, CHN
[‡]College of Information and Computer Sciences, University of Massachusetts Amherst, USA
wufeng02@ustc.edu.cn, shlomo@cs.umass.edu, xpchen@ustc.edu.cn

Abstract

We propose a novel baseline regret minimization algorithm for multi-agent planning problems modeled as finite-horizon decentralized POMDPs. It guarantees to produce a policy that is provably at least as good as a given baseline policy. We also propose an iterative belief generation algorithm to efficiently minimize the baseline regret, which only requires necessary iterations so as to converge to the policy with minimum baseline regret. Experimental results on common benchmark problems confirm the benefits of the algorithm compared with the state-of-the-art approaches.

1 Introduction

Multi-agent planning under uncertainty is an active research area of AI. Decentralized POMDP (DEC-POMDP) is a natural extension of Markov decision process (MDP) and its partially observable counterpart (i.e., POMDP) to cooperative multi-agent settings. Although the DEC-POMDP framework is very expressive and useful for modeling many real-world applications, solving it optimally is NEXP-hard [Bernstein *et al.*, 2002] and even finding near-optimal solution is very challenging [Rabinovich *et al.*, 2003]. Therefore, optimal algorithms [Hansen *et al.*, 2004; Szer *et al.*, 2005; Aras and Dutech, 2010; Oliehoek *et al.*, 2013; Dibangoye *et al.*, 2016] can only solve small problems and approximate methods [Pajarinen and Peltonen, 2011; Wu *et al.*, 2011; Amato *et al.*, 2014; Kumar *et al.*, 2016] usually offer no theoretical guarantees on the quality of the obtained policies.

Since deploying new solutions in real-world settings such as business product deployment or healthcare delivery may be costly or unsafe, people are often unwilling to try them unless some performance guarantees can be provided. A tight error bound compared to the optimal solution is desirable, but may be difficult to compute because the optimal solution is often unknown. Alternatively, it is useful to produce policies that are guaranteed to outperform some baseline policies (e.g., the policies that are currently in use). Furthermore, for the sake of system stability, it is sometimes desirable to incrementally improve some aspects of the current policies that have been identified to be of low quality, instead of deploying completely new policies. While initial studies of such considerations of real-world deployments have been conducted recently with respect to MDPs [Petrik *et al.*, 2016], they have not been fully explored, particularly in the more complex setting defined by a DEC-POMDP.

Against this background, we propose a novel algorithm for multi-agent planning with baseline regret minimization. We focus on finite-horizon DEC-POMDP problems and assume that some baseline policies are given as input, either generated by an existing planner or created by experts in advance. In our approach, we define the baseline regret as the worst-case loss in expected value by adopting the given policies instead of the baseline policies. Then, we generate new policies with no regret relative to the baseline policies by minimizing the baseline regret. By doing so, we guarantee to produce policies that are provably better than or at least equivalent to the baseline policies in solution quality. As aforementioned, this is crucial for deployed real-world applications.

Due to the large policy and belief space, minimizing the baseline regret (worst-case loss) for DEC-POMDPs is computationally challenging. To address this, we propose an effective and efficient technique with *iterative belief generation*. Specifically, our algorithm repeatedly solve the primary and secondary optimization problems until it converges to the policies with minimum baseline regret. In this context, the primary optimization problem generates a policy that minimizes the baseline regret given a set of beliefs, while the secondary optimization problem tries to expand the belief set with a new belief point that maximizes the baseline regret of the current policies. Empirically, we tested our algorithm on several common DEC-POMDP benchmark problems and compared its results with the state-of-the-art solvers.

This paper advances the state-of-the-art by making the following two key contributions:

- We define the baseline regret in Section 3 as a novel solution criteria for solving finite-horizon DEC-POMDPs and establish a connection between regret minimization and policy optimization. In particular, we theoretically prove that a policy with minimum baseline regret guarantees to be not worse than the baseline policy.
- We propose an effective and efficient method based on iterative belief generation in Section 4 for finding policies with minimum baseline regret. We prove that our

algorithm will converge to the policies with minimum baseline regret and only requires necessary iterations to converge. We show that it can be used to improve parts of the polices as needed, or be systematically integrated with dynamic programming to improve entire policies.

In sum, we present a novel approach for finite-horizon DEC-POMDPs with theoretical guarantees. In the experiments, we empirically evaluate our approach on common benchmark problems to confirm its advantages.

2 Background

This section briefly reviews the finite-horizon DEC-POMDP model and its solutions based on dynamic programming.

2.1 Decentralized POMDPs

A finite-horizon decentralized POMDP (DEC-POMDP) is defined as tuple $\langle I, S, b^0, \{A_i\}, P, \{\Omega_i\}, O, R, T\rangle$, where:

- I is a set of n agents.
- S is a finite set of states.
- $b^0 \in \Delta(S)$ is the initial state distribution.
- A_i is a finite set of actions for each agent $i \in I$ and $A = \times_{i\in I} A_i$ is the set of joint actions.
- $P : S \times A \times S \to [0, 1]$ is the transition function where $P(s'|s, a)$ is the probability of transiting to next state s' when taking joint action a in state s.
- Ω_i is a finite set of observations for each agent $i \in I$ and $\Omega = \times_{i\in I}\Omega_i$ is the set of joint observations.
- $O : S\times A\times\Omega \to [0, 1]$ is the observation function where $O(o|a, s')$ is the probability of observing joint observation o after taking joint action o with outcome state s'.
- $R : S \times A \to \Re$ is the reward function where $R(s, a)$ is the reward after taking joint action a in state s.
- T is the horizon.

A local policy $q_i : \bar{\Omega}_i \to A_i$ of agent $i \in I$ is a mapping from its local observation histories $\bar{\Omega}_i = (o_i^1, o_i^2, \cdots, o_i^t)$ to its actions A_i and a joint policy is a collection of local policies $q = \langle q_1, q_2, \cdots, q_n\rangle$, one for each agent. A joint policy q is evaluated based on the value function computed recursively:

$$V(s, q) = R(s, a_q) + \sum_{s'\in S, o\in\Omega} P(s'|s, a_q)O(o|a_q, s')V(s', q_o)$$

where a_q is the joint action specified by q and q_o is the joint sub-policy of q after receiving joint observation o.

The goal of solving a DEC-POMDP is to find a joint policy q^* that maximizes the expected value given the initial state distribution b^0: $q^* = \text{argmax}_q \sum_{s\in S} b^0(s)V(s, q)$. Note that the planning for a DEC-POMDP can be done centrally while the policies must be executed in a decentralized manner.

2.2 Dynamic Programming for DEC-POMDPs

For finite-horizon DEC-POMDPs, a local policy is usually represented as a policy tree. One of the techniques to generate policy trees is based on dynamic programming [Hansen *et al.*, 2004]. It starts from the leaf nodes of the policy tree and iteratively build the trees based on the one-step backup operation until the complete policies are constructed. Although the optimal policy trees can be constructed, the standard dynamic programming runs out memory very quickly even for small toy problems.

Memory-bounded dynamic programming (MBDP) is an approximate algorithm that can solve DEC-POMDPs with long horizon [Seuken and Zilberstein, 2007]. Instead of keeping all the policy tree at each iteration, it selects a fixed number of candidate trees (with parameter $maxTree$) identified by heuristics portfolio after the backup operation. MBDP-based successors have been proposed for large problems with good empirical performance [Amato *et al.*, 2009; Kumar and Zilberstein, 2010; Wu *et al.*, 2010a; Wu *et al.*, 2010b; Wu *et al.*, 2010c; Wu *et al.*, 2012]. However, they usually offer no theoretically guarantees on solution quality.

3 Baseline Regret Minimization

In this paper, we assume that a joint baseline policy for the DEC-POMDP has been generated either manually by domain experts or other algorithms. This joint baseline policy is represented as a collection of policy trees. The input of our algorithm is a local baseline policy q_i° of agent i and the outcome is a new policy q_i that guarantees to be not worse (possibly better) than the baseline policy. During the process, we assume that the policies of the other agents are fixed and only the given baseline policy of agent i is improved.

In DEC-POMDPs, a distribution over the underlying world state is not a sufficient statistic [Oliehoek, 2013] for the decision-making agents. Instead, we consider multi-agent belief state [Nair *et al.*, 2003] as a distribution over the states and the policies of the other agents, i.e., $b \in \Delta(S \times Q_{-i})$, where $b_{s,q_{-i}}$ is the probability at state $s \in S$ and the other agents following policies $q_{-i} \in Q_{-i}$. Then, given such a belief state b, the expected value of agent i's policy q_i can be computed based on the value function as:

$$V(b, q_i) = \sum_{s,q_{-i}} b_{s,q_{-i}}[R(s, a) + \sum_{s',o} Pr(s', o|s, a)V(s', q')]$$

where a is the joint action selected by joint policy $q = (q_i, q_{-i})$, q' is the next joint policy of q after observing o, and $Pr(s', o|s, a) = P(s'|s, a)O(o|a, s')$ is the short hand of the transition and observation probabilities.

Let $\mathcal{R}(b, q_i)$ be the regret of agent i adopting policy q_i instead of the optimal policy q_i^* at belief state b defined as:

$$\mathcal{R}(b, q_i) = V(b, q_i^*) - V(b, q_i). \quad (1)$$

Let B be a subset of the overall belief space. We define the worst-case regret (loss) of policy q_i in the belief region B as: $\mathcal{R}(q_i) = \max_{b\in B} \mathcal{R}(b, q_i)$. Then the optimal policy q_i^* in B can be computed as: $q_i^* = \text{argmin}_{q_i\in Q_i} \mathcal{R}(q_i)$ where Q_i is the overall policy space. Note that the optimal policy q_i^* is also a no-regret policy because $\mathcal{R}(q_i^*) = 0$.

Here, we make a connection between regret minimization and policy optimization. We show that finding the no-regret policy is equivalent to computing the optimal policy. Therefore, it is as hard as solving the entire problem optimally. Instead of finding the optimal policy, our goal is to generate

a policy that is not worse than the baseline policy. In other words, we try to compute a policy q_i that has less or equal regret than the baseline policy.

Definition 1. The baseline regret of policy q_i is the maximum loss in expected values by adopting policy q_i instead of the baseline policy q_i° in the belief region B:

$$\mathcal{R}^\circ(q_i) = \max_{b \in B} [V(b, q_i^\circ) - V(b, q_i)] \tag{2}$$

Theorem 1. *A policy with minimum baseline regret has a value that is not worse than the one of the baseline policy for any belief state in the belief region.*

Proof. Let $q_i = \operatorname{argmin}_{q_i'} \mathcal{R}^\circ(q_i')$ be the policy with minimum baseline regret. we have $\mathcal{R}^\circ(q_i) \leq 0$ because:

$$\mathcal{R}^\circ(q_i) = \min_{q_i'} \mathcal{R}^\circ(q_i') \leq \mathcal{R}^\circ(q_i^\circ) = 0.$$

For any belief state $b \in B$ in the region B, we have:

$$\begin{aligned} V(b, q_i^\circ) - V(b, q_i) &\leq \max_{b' \in B} [V(b', q_i^\circ) - V(b', q_i)] \\ &= \mathcal{R}^\circ(q_i) \leq 0. \end{aligned}$$

Thus, we conclude with: $\forall_{b \in B}, V(b, q_i^\circ) \leq V(b, q_i)$. □

Now, our goal is to compute a new policy q_i that minimizes the baseline regret $\mathcal{R}^\circ(q_i)$ as follow:

$$q_i = \operatorname*{argmin}_{q_i'} \max_{b \in B} [V(b, q_i^\circ) - V(b, q_i')] \tag{3}$$

Here, the key challenge is how to determine the belief region B for the baseline regret minimization. Next, we propose an iterative approach to address this challenge.

4 Iterative Belief Generation

Recall that we want to compute a policy q_i for agent i that minimizes the regret given the baseline policy q_i°. This can be formulated as the following optimization problem:

$$\begin{array}{ll} \min_{q_i} & z \\ \text{s.t.} & \forall_{b \in B}\ V(b, q_i^\circ) - V(b, q_i) \leq z \leq 0 \end{array} \tag{4}$$

Note that this optimization problem is always feasible because q_i° is a solution if no other policy is better than q_i°.

As aforementioned, the key challenge of this optimization problem is to determine B. Our basic idea is to start with an initial (possibly small) set B and then grow it iteratively. Specifically, we solve two optimization problems: In the primary optimization problem, we compute a new policy given B; In the secondary optimization problem, we identify a new belief state and add it to B. We iteratively solve the primary and secondary optimization problems until convergence.

In the secondary optimization problem, we want to find a belief state b in some region that maximizes the regret of the current policy q_i as follow:

$$b = \operatorname*{argmax}_{b'} [V(b', q_i^\circ) - V(b', q_i)] \quad \text{s.t.} \quad b \in \mathcal{B} \tag{5}$$

Recall that our goal is to find a policy that is not worse than the baseline policy. Hence, we constrain the belief state in a belief region $\mathcal{B}$ where the baseline policy achieves the best value compared with the other policies in the baseline policy set Q_i°. In more detail, we define the belief region as follow:

$$\mathcal{B} = \{b | \forall_{q_i' \in Q_i^\circ}, V(b, q_i^\circ) \geq V(b, q_i')\} \tag{6}$$

Now, we can identify a new belief state and formulate the secondary optimization problem as follow:

$$\begin{array}{ll} \max_b & z \\ \text{s.t.} & V(b, q_i^\circ) - V(b, q_i) \geq z \geq 0 \\ & \forall_{q_i' \in Q_i^\circ}, V(b, q_i^\circ) \geq V(b, q_i') \end{array} \tag{7}$$

In more detail, the secondary optimization problem only considers the belief region B where the baseline policy q_i° is the best. In this region, we find a new belief b in which the policy q_i has the maximum regret.

The overall process is outlined in Algorithm 1. It starts with a baseline policy and an initial set of belief states. Then it interleaves solving the primary and secondary optimization problems until the secondary optimization problem has no solution or the solution is already in the belief set. In both cases, the algorithm converges and returns the improved policy.

Algorithm 1: Iterative Belief Generation

```
1  function IBG(q_i^∘)
2      B ← initialize()
3      repeat
4          q_i ← optimizePrimary(q_i^∘, B)
5          b ← optimizeSecondary(q_i, q_i^∘, B)
6          if b = ∅ or b ∈ B then
7              break # converged
8          B ← B ∪ {b}
9      until timeout()
10     return q_i
```

Theorem 2. *The iterative belief generation algorithm always converges to a policy with minimum baseline regret.*

Proof. Suppose that the algorithm converges to policy q_i with positive baseline regret, i.e., $\mathcal{R}^\circ(q_i) > 0$. By the definition of the baseline regret, there exists a belief state $b \notin B$ such that:

$$\mathcal{R}^\circ(q_i) = V(b, q_i^\circ) - V(b, q_i) > 0$$

If b is in the target belief region, i.e., $b \in \mathcal{B}$, then it will be generated by the secondary optimization problem in Equation 7. This is contradictory to the fact that the algorithm converged. If the belief set B is complete, then the primary optimization procedure guarantees to generate a policy with minimum baseline regret. Thus, we conclude that the iterative belief generation algorithm will always converge to a policy with minimum baseline regret. □

Definition 2. A belief state b is non-dominated by the other belief states if there exists a policy in which b has larger baseline regret than the other belief states as follow:

$$\exists q_i, \forall b', V(b, q_i^\circ) - V(b, q_i) > V(b', q_i^\circ) - V(b', q_i) \tag{8}$$

Lemma 1. *For any given policy, the corresponding non-dominated belief state is necessary and sufficient to determine the baseline regret of the policy.*

Proof. This can be proved by the definition of the baseline regret for any given policy q_i:

$$\mathcal{R}^\circ(q_i) = \max_{b \in B}[V(b, q_i^\circ) - V(b, q_i)]$$

If b is the corresponding non-dominated belief state of policy q_i but $b \notin B$, then the regret between the baseline policy and q_i is not maximized because by definition:

$$V(b, q^\circ) - V(b, q_i) > \max_{b' \in B}[V(b', q_i^\circ) - V(b', q_i)]$$

Thus, b is necessary to determine the baseline regret of q_i.

If b' is a dominated belief state of policy q_i, but $b' \in B$, then it can be safely removed from B because there exists a non-dominated belief state $b \in B$ such that:

$$V(b', q_i^\circ) - V(b', q_i) < V(b, q_i^\circ) - V(b, q_i) = \mathcal{R}^\circ(q_i)$$

Thus, b is sufficient to determine the baseline regret of q_i. □

Lemma 2. *All the belief states generated by the secondary optimization process are non-dominated belief states.*

Proof. For any belief state b generated by the secondary optimization process, we have a policy q_i such that:

$$\begin{aligned} V(b, q_i^\circ) - V(b, q_i) &= \max_{b' \in \mathcal{B}}[V(b', q_i^\circ) - V(b', q_i)] \\ &> V(b', q_i^\circ) - V(b', q_i), \forall b' \in \mathcal{B} \backslash b \end{aligned}$$

Thus, b is a non-dominated belief state for policy q_i. □

Lemma 3. *All the necessary non-dominated belief states will be generated by the secondary optimization.*

Proof. Suppose that a belief state $b \in \mathcal{B}$ is non-dominated for policy q_i, but not generated by the secondary optimization, i.e., $b \notin B$. If $\forall b' \in B$, q_i has the minimum baseline regret, then q_i will be generated by the primary optimization. After that, b will be generated by the secondary optimization. If $\forall b' \in B$, q_i does not have the minimum baseline regret, then q_i is not the target policy with overall minimum baseline regret. Thus, it is not necessary to consider b. □

Theorem 3. *The iterative belief generation algorithm requires only iterations that are necessary for convergence.*

Proof. According to Lemma 3, all the necessary non-dominated belief states will be generated by the secondary optimization. Suppose that an unnecessary belief state $b \in B$ is also generated by the secondary optimization. According to Lemma 2, b is a non-dominated belief state. Let q_i be the corresponding policy for b. According to Lemma 1, the baseline regret of q_i cannot be correctly computed without b. Therefore, b is also necessary for the algorithm. Thus, the algorithm requires only iterations necessary for convergence. □

Note that the size of non-dominated belief states is often much less than the overall belief space, i.e., $|B| \ll |\mathcal{B}|$. Therefore, it is effective and efficient by considering only non-dominated beliefs that are necessary for convergence.

In the following sections, we describe how the primary and secondary optimization problems can be solved in details.

4.1 Primary Optimization for Policy Improvement

In the primary optimization procedure, we try to improve the current policy q_i by solving the optimization problem as shown in Equation 4. Note that we represent an agent's local policy as a policy tree where each tree node is associated with an action and branches, one for each observation.

In finite-horizon DEC-POMDPs, the number of possible trees grows doubly exponentially with the horizon. It is usually computational intractable to search over the entire policy space with full horizon. Therefore, in our primary optimization, we perform one-step improvement for the current policy. Specifically, we represent policy q_i as x, y-variables where:

- $\forall_{a_i \in A_i}, x_{a_i} \in \{0, 1\}$ is a binary variable where $x_{a_i} = 1$ if action a_i is taken by the root node of the policy tree and 0 otherwise. Since in deterministic policy trees each node can have only an action, we have the constraint for a valid policy as: $\sum_{a_i \in A_i} x_{a_i} = 1$.
- $\forall_{a_i \in A_i, o_i \in \Omega_i, q_i' \in Q_i'}, y_{a_i, o_i, q_i'} \in \{0, 1\}$ is also a binary variable where $y_{a_i, o_i, q_i'} = 1$ if action a_i is taken by the root node and sub-policy q_i' is associated with the branch o_i, and 0 otherwise. Since each observation branch can have ony a sub-policy, we have the constraint for a valid policy as: $\forall_{a_i \in A_i, o_i \in \Omega_i} \sum_{q_i' \in Q_i'} y_{a_i, o_i, q_i'} = x_{a_i}$.

Given the x, y-variables for the new policy and z as its regret, we realize the optimization problem in Equation 4 with the following mixed integer linear programming (MILP):

$$\begin{aligned} \min_{x,y} \quad & z \\ \text{s.t.} \quad & \forall_{b \in B}, V(b, q_i^\circ) - \sum_{s, q_{-i}} b_{s, q_{-i}} [\sum_{a_i} x_{a_i} R(s, a) + \\ & \sum_{s', o} \sum_{a_i, q_i'} y_{a_i, o_i, q_i'} Pr(s', o | s, a) V(s', q')] \leq z \leq 0 \\ & \textstyle\sum_{a_i} x_{a_i} = 1; \forall_{a_i, o_i}, \sum_{q_i'} y_{a_i, o_i, q_i'} = x_{a_i} \\ & \forall_{a_i}, x_{a_i} \in \{0, 1\}; \forall_{a_i, o_i, q_i'}, y_{a_i, o_i, q_i'} \in \{0, 1\}. \end{aligned}$$

For each belief state b, we have the following short-hands:

$$R_{b, a_i} = \sum_{s, q_{-i}} b_{s, q_{-i}} R(s, a) \tag{9}$$

where the joint action $a = (a_i, a_{-i})$ with a_{-i} selected by q_{-i}.

$$V_{b, a_i, o_i, q_i'} = \sum_{s, q_{-i}} b_{s, q_{-i}} \sum_{s', o_{-i}} Pr(s', o | s, a) V(s', q') \tag{10}$$

where the joint sub-policy $q' = (q_i', q_{-i}')$ with q_{-i}' selected by q_{-i} given observation o_{-i}.

Now, we can simplify the aforementioned MILP as follow:

$$\begin{aligned} \min_{x,y} \quad & z \\ \text{s.t.} \quad & \forall_{b \in B}, V(b, q_i^\circ) - [\sum_{a_i} x_{a_i} R_{b, a_i} + \\ & \sum_{a_i, o_i, q_i'} y_{a_i, o_i, q_i'} V_{b, a_i, o_i, q_i'}] \leq z \leq 0 \quad (a) \\ & \textstyle\sum_{a_i} x_{a_i} = 1; \forall_{a_i, o_i}, \sum_{q_i'} y_{a_i, o_i, q_i'} = x_{a_i} \quad (b) \\ & \forall_{a_i}, x_{a_i} \in \{0, 1\}; \forall_{a_i, o_i, q_i'}, y_{a_i, o_i, q_i'} \in \{0, 1\} \quad (c) \end{aligned}$$

where constraint (a) guarantees that the regret for the new policy is minimized and no more than the baseline, and constraints (b) and (c) ensure that the variables for the new policy are valid. Therefore, this MILP has $(|A_i| + |A_i||\Omega_i||Q_i'| + 1)$ variables and $(|B| + |A_i||\Omega_i| + 2)$ constraints.

Algorithm 2: Dynamic Programming with IBG

```
function IBG-DP(q°)
    for t = 1 to T do # policy initialization
        Q_i^t ← initialize(q_i^°), ∀i ∈ I
        V(q^t) ← evaluate(q^t), ∀q^t ∈ Q^t
    for t = T to 1 do # policy improvement
        repeat
            foreach i ∈ I and q_i^t ∈ Q_i^t do
                q_i^t ← IBG(q_i^t)
        until timeout()
        V(q^t) ← evaluate(q^t), ∀q^t ∈ Q^t
    return q° # the improved joint policy
```

4.2 Secondary Optimization for Belief Generation

In the secondary optimization problem, we aim at finding a new belief state by solving the optimization problem shown in Equation 7. Recall that a belief state is a probability distribution over the states and the other agents' policies. Therefore, it must satisfy the probability constraints as follow: $\forall_{s,q_{-i}}, x_{s,q_{-i}} \geq 0$ and $\sum_{s,q_{-i}} x_{s,q_{-i}} = 1$.

As discussed above, we expect the worse-case belief state for the current policy q_{-i}. In other words, we want to compute the belief state that has the maximum regret for q_{-i}. By considering such belief state, we are able to gain the maximum improvement on the current policy. Besides, we only consider the belief region where the baseline policy has the best value. This is important because our goal is to find a policy that is not worse than the baseline policy. Therefore, the belief region beyond the scope of the currently considered baseline policy should be excluded.

Now, we realize the secondary optimization problem in Equation 7 with the following linear program (LP):

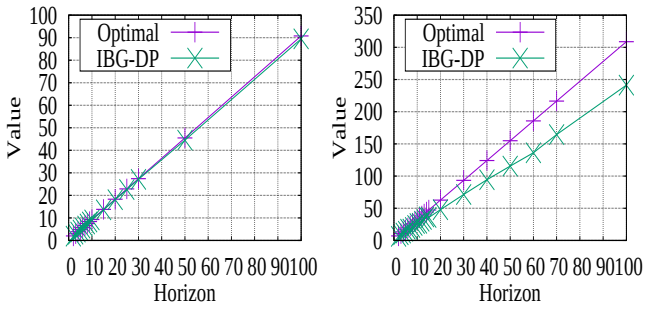
$$
\begin{array}{lll}
\max\limits_{x} & z & \\
\text{s.t.} & \sum\limits_{s,q_{-i}} x_{s,q_{-i}}[V(s,q^\circ) - V(s,q)] \geq z \geq 0 & (a) \\
& \forall_{q_i' \in Q_i^\circ}, \sum\limits_{s,q_{-i}} x_{s,q_{-i}}[V(s,q^\circ) - V(s,q')] \geq 0 & (b) \\
& \sum_{s,q_{-i}} x_{s,q_{-i}} = 1; \forall_{s,q_{-i}}, x_{s,q_{-i}} \geq 0 & (c)
\end{array}
$$

where the joint policies $q = (q_i, q_{-i})$, $q^\circ = (q_i^\circ, q_{-i})$, and $q' = (q_i', q_{-i})$. This LP has $(|S||Q_{-i}| + 1)$ variables and $(|Q_i^\circ| + |S||Q_{-i}| + 3)$ constraints.

4.3 Integrating with Dynamic Programming

Generally, we can randomly choose a node in the policy tree representing the baseline policy and improve it using our method. Note that due to the huge policy space, we propose a more practical technique to perform only one-step improvement for the given node. Therefore, a more systematic way to improve the whole baseline policy is to integrate our method with dynamic programming. The main procedures of our dynamic programming algorithm are outlined in Algorithm 2.

Theorem 4. *The proposed algorithm has linear time complexity with respect to the horizon.*



(a) Broadcast Channel (b) Recycling Robots

Figure 1: Results of IBG-DP vs. Optimal solution.

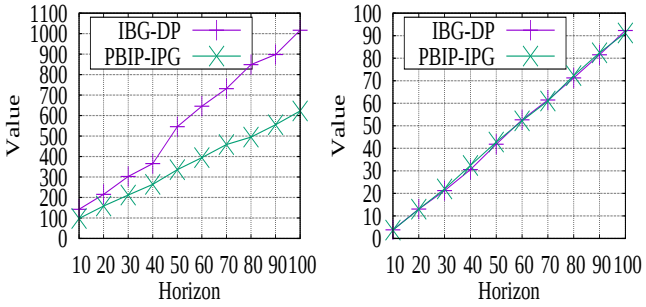
Note that the output of iterative belief generation in Algorithm 2 is a new policy that replaces the baseline policy. In other words, the total number of policy trees does not grow with the iteration steps. Thus, the proposed algorithm share the same linear time complexity w.r.t. the horizon with the common MBDP-based approximate algorithms.

5 Experiments

We empirically evaluated our algorithm on four common benchmark problems[1] widely used in the DEC-POMDP literature: Broadcast Channel, Recycling Robots, Cooperative Box Pushing, and Meeting in a 3×3 Grid. We ran our algorithm on each problem instance multiple times until the results were statistically meaningful and reported the average policy values. Our algorithm (i.e., IBG-DP) was implemented in Java 1.8 and ran on a machine with 3.5GHz Intel Core i7 CPU and 8GB of RAM. The MILP and LP were solved by IBM CPLEX 12.61. We conducted two sets of experiments to illustrate the performance of our algorithm.

In the first set of our experiments, we compared our results with the optimal values obtained by existing optimal algorithms[2] for DEC-POMDPs. The baseline policies that we used here are random policy trees with $maxTree = 3$. Our goal is to show that IBG-DP is able to get near-optimal values by minimizing the baseline regret starting with random policies. Figure 1 summarizes our results. As shown in Figure 1a, IBG-DP obtained the optimal values for the Broadcast Channel problem with different horizons. In Figure 1b, we see that IBG-DP achieved near-optimal values for the Recycling Robots domain. Specifically, the bound w.r.t. the optimal value is tight for the instances with short horizon and becomes loose as the horizon increases. This is because we fixed the number of subtrees in the baseline policies to be three (i.e., $maxTree = 3$). IBG-DP does not increase the number of subtrees for scalability consideration when it improves the baseline policies at each iteration. This may be sufficient for some problems (e.g., Broadcast Channel) but may significantly reduce the value for some other domains (e.g., Recycling Robots). Nevertheless, IBG-DP still achieved competitive values (e.g., 80% optimal at $T = 100$).

[1] http://masplan.org/problem_domains
[2] http://masplan.org/optimal_values



(a) Cooperative Box Pushing (b) Meeting in 3×3 Grid

Figure 2: Results of IBG-DP vs. PBIP-IPG.

In the second set of experiments, we compared our algorithm with a leading approximate algorithm for finite-horizon DEC-POMDPs. Specifically, we compared with PBIP-IPG [Amato *et al.*, 2009] that consistently outperforms other algorithms such as MBDP, IMBDP, MBDP-OC and PBIP. TBDP and DecRSPI are faster than PBIP-IPG but have similar values for the tested domains. Our goal is to show that IBG-DP scales well for large problems as existing approximate algorithms and can get similar or often better values. Figure 2 summarizes our results. In Figure 2a, we show that IBG-DP improved the baseline policies with large margin for the Cooperative Box Pushing problem. This problem is considerably harder than the aforementioned two problems. IBG-DP can further improve the baseline policies by minimizing the baseline regret. For the Meeting in a 3×3 Grid problem as depicted in Figure 2b, IBG-DP achieved similar values to the policies computed by PBIP-IPG. This may be because the policies obtained by PBIP-IPG have already been the best ones for $maxTree = 3$.

Table 1: Runtime Results for Broadcast Channel

Horizon	5	10	30	50	100
IBG-DP	1.47s	1.72s	**3.39s**	**5.09s**	**10.14s**
FB-HSVI	0.33s	0.78s	14.0s	41.7s	473.3s

Generally, IBG-DP takes more time than MBDP based algorithms because it has to optimize the policies w.r.t. a set of beliefs instead of a single one. Table 1 illustrates our runtime results for the Broadcast Channel problem comparing to FB-HSVI [Dibangoye *et al.*, 2016] – a leading optimal algorithm. For instances with short horizons (e.g., 5, 10), IBG-DP took more time than FB-HSVI because it must call the external CPLEX solver with JNI multiple times. As expected, IBG-DP outperformed FB-HSVI in runtime for instances with long horizons (e.g., 30, 50, 100) because IBG-DP has linear time complexity w.r.t. the horizon as with MBDP-based algorithms. In practice, it can be used as an anytime algorithm, which may be stopped before convergence with the risk of violating the guarantees. This is still useful as the policies have been optimized over a set of belief states.

6 Related Work

Regret minimization is an online learning technique originally used for the multi-armed bandit (MAB) problem. It has been broadly used to solve complex games with incomplete information [Zinkevich *et al.*, 2007; Wu and Jennings, 2014]. In the context of decision-theoretic planning, several approaches [Xu and Mannor, 2009; Regan and Boutilier, 2010; Ahmed *et al.*, 2013; Petrik *et al.*, 2016] have been proposed to find robust solutions for uncertain MDPs with the maxmin/minmax regret criteria in order to provide some guarantees on the worst-case performance. Among them, Regan and Boutilier [2010] use an iterative constraint generation approach to solve large LP. Similar to ours, Petrik *et al.* [2016] proposed a safe policy improvement method by minimizing robust baseline regret for MDPs. By contrast, our algorithm is designed to solve DEC-POMDPs, requiring substantially different solution and analysis techniques. Recently, Banerjee and Kraemer [2013] proposed a decentralized planning approach based on counterfactual regret minimization for DEC-POMDPs. They represent the DEC-POMDP as a strategic game and run regret decomposition in the tree in order to minimize the overall regret. In contrast, we try to find policies with minimum baseline regret w.r.t. the given baseline policies.

7 Conclusion

We proposed a novel approach for solving finite-horizon DEC-POMDPs, where the baseline regret of a policy is introduced as a new solution criterion. We show that a policy with minimum baseline regret guarantees to be not worse than the baseline policy. Due to the large policy space of DEC-POMDPs, minimizing the baseline regret is computational challenging. To address this, we proposed the IBG algorithm with the procedures for iteratively improving the baseline policy and finding the non-dominated beliefs. We proved that IBG will eventually converge to a policy with minimum baseline regret with only necessary iterations. We also presented an implementation of IBG based on MILP and LP, which can be systematically integrated with dynamic programming to improve the entire policies (namely IBG-DP). Our experiments on four common DEC-POMDPs benchmark problems show that IBG-DP obtained near-optimal policies (e.g., Broadcast Channel) or made significant improvement (e.g., Cooperative Box Pushing) compared to the state-of-the-art. In future work, we will further explore the baseline regret minimization technique for DEC-POMDPs in model-free settings, building on our earlier work [Wu *et al.*, 2010b; Wu *et al.*, 2013] where the baseline solution may be encoded in the form of a hand-crafted controller.

Acknowledgments

This work was supported in part by National Natural Science Foundation of China under grant No. 61603368, the Youth Innovation Promotion Association of CAS (No. 2015373), and Natural Science Foundation of Anhui Province under grant No. 1608085QF134.

References

[Ahmed *et al.*, 2013] Asrar Ahmed, Pradeep Varakantham, Yossiri Adulyasak, and Patrick Jaillet. Regret based robust solutions for uncertain Markov decision processes. In *Proc. of NIPS*, pages 881–889, 2013.

[Amato *et al.*, 2009] Chistopher Amato, Jilles S. Dibangoye, and Shlomo Zilberstein. Incremental policy generation for finite-horizon DEC-POMDPs. In *Proc. of ICAPS*, pages 2–9, 2009.

[Amato *et al.*, 2014] Christopher Amato, George D. Konidaris, and Leslie P. Kaelbling. Planning with macro-actions in decentralized POMDPs. In *Proc. of AAMAS*, pages 1273–1280, 2014.

[Aras and Dutech, 2010] Raghav Aras and Alain Dutech. An investigation into mathematical programming for finite horizon decentralized POMDPs. *Journal of Artificial Intelligence Research*, 37(1):329–396, 2010.

[Banerjee and Kraemer, 2013] Bikramjit Banerjee and Landon Kraemer. Counterfactual regret minimization for decentralized planning. In *Workshop on Multiagent Sequential Decision Making*, pages 32–39, 2013.

[Bernstein *et al.*, 2002] Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of Markov decision processes. *Math. of Operations Research*, 27(4):819–840, 2002.

[Dibangoye *et al.*, 2016] Jilles S. Dibangoye, Christopher Amato, Olivier Buffet, and François Charpillet. Optimally solving Dec-POMDPs as continuous-state MDPs. *Journal of Artificial Intelligence Research*, 55:443–497, 2016.

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

[Kumar and Zilberstein, 2010] Akshat Kumar and Shlomo Zilberstein. Point-based backup for decentralized POMDPs: Complexity and new algorithms. In *Proc. of AAMAS*, pages 1315–1322, 2010.

[Kumar *et al.*, 2016] Akshat Kumar, Hala Mostafa, and Shlomo Zilberstein. Dual formulations for optimizing Dec-POMDP controllers. In *Proc. of ICAPS*, pages 202–210, 2016.

[Nair *et al.*, 2003] Ranjit Nair, Milind Tambe, Makoto Yokoo, David Pynadath, and Stacy Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proc. of IJCAI*, pages 705–711, 2003.

[Oliehoek *et al.*, 2013] Frans A. Oliehoek, Matthijs T. J. Spaan, Christopher Amato, and Shimon Whiteson. Incremental clustering and expansion for faster optimal planning in Dec-POMDPs. *Journal of Artificial Intelligence Research*, 46:449–509, 2013.

[Oliehoek, 2013] Frans A. Oliehoek. Sufficient plan-time statistics for decentralized POMDPs. In *Proc. of IJCAI*, pages 302–308, 2013.

[Pajarinen and Peltonen, 2011] Joni K. Pajarinen and Jaakko Peltonen. Periodic finite state controllers for efficient POMDP and DEC-POMDP planning. In *Proc. of NIPS*, pages 2636–2644, 2011.

[Petrik *et al.*, 2016] Marek Petrik, Yinlam Chow, and Mohammad Ghavamzadeh. Safe policy improvement by minimizing robust baseline regret. In *Proc. of NIPS*, pages 2298–2306, 2016.

[Rabinovich *et al.*, 2003] Zinovi Rabinovich, Claudia V. Goldman, and Jeffrey S. Rosenschein. The complexity of multiagent systems: The price of silence. In *Proc. of AAMAS*, pages 1102–1103, 2003.

[Regan and Boutilier, 2010] Kevin Regan and Craig Boutilier. Robust policy computation in reward-uncertain MDPs using nondominated policies. In *Proc. of AAAI*, pages 1127–1133, 2010.

[Seuken and Zilberstein, 2007] Sven Seuken and Shlomo Zilberstein. Memory-bounded dynamic programming for DEC-POMDPs. In *Proc. of IJCAI*, pages 2009–2015, 2007.

[Szer *et al.*, 2005] Daniel Szer, François Charpillet, and Shlomo Zilberstein. MAA*: A heuristic search algorithm for solving decentralized POMDPs. In *Proc. of UAI*, pages 576–583, 2005.

[Wu and Jennings, 2014] Feng Wu and Nicholas R. Jennings. Regret-based multi-agent coordination with uncertain task rewards. In *Proc. of AAAI*, pages 1492–1499, 2014.

[Wu *et al.*, 2010a] Feng Wu, Shlomo Zilberstein, and Xiaoping Chen. Point-based policy generation for decentralized POMDPs. In *Proc. of AAMAS*, pages 1307–1314, 2010.

[Wu *et al.*, 2010b] Feng Wu, Shlomo Zilberstein, and Xiaoping Chen. Rollout sampling policy iteration for decentralized POMDPs. In *Proc of UAI*, pages 666–673, 2010.

[Wu *et al.*, 2010c] Feng Wu, Shlomo Zilberstein, and Xiaoping Chen. Trial-based dynamic programming for multi-agent planning. In *Proc. of AAAI*, pages 908–914, 2010.

[Wu *et al.*, 2011] Feng Wu, Shlomo Zilberstein, and Xiaoping Chen. Online planning for multi-agent systems with bounded communication. *Artificial Intelligence*, 175(2):487–511, 2011.

[Wu *et al.*, 2012] Feng Wu, Nicholas R. Jennings, and Xiaoping Chen. Sample-based policy iteration for constrained DEC-POMDPs. In *Proc. of ECAI*, pages 858–863, 2012.

[Wu *et al.*, 2013] Feng Wu, Shlomo Zilberstein, and Nicholas R. Jennings. Monte-carlo expectation maximization for decentralized POMDPs. In *Proc. of IJCAI*, pages 397–403, 2013.

[Xu and Mannor, 2009] Huan Xu and Shie Mannor. Parametric regret in uncertain Markov decision processes. In *Proc. of CDC*, pages 3606–3613, 2009.

[Zinkevich *et al.*, 2007] Martin Zinkevich, Michael Johanson, Michael H. Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In *Proc. of NIPS*, pages 1729–1736, 2007.