Point-Based Backup for Decentralized POMDPs: Complexity and New Algorithms

Akshat Kumar
Department of Computer Science
University of Massachusetts
Amherst, MA, 01003
akshat@cs.umass.edu

Shlomo Zilberstein
Department of Computer Science
University of Massachusetts
Amherst, MA, 01003
shlomo@cs.umass.edu

ABSTRACT

Decentralized POMDPs provide an expressive framework for sequential multi-agent decision making. Despite their high complexity, there has been significant progress in scaling up existing algorithms, largely due to the use of point-based methods. Performing point-based backup is a fundamental operation in state-of-the-art algorithms. We show that even a single backup step in the multi-agent setting is NP-Complete. Despite this negative worst-case result, we present an efficient and scalable optimal algorithm as well as a principled approximation scheme. The optimal algorithm exploits recent advances in the weighted CSP literature to overcome the complexity of the backup operation. The polytime approximation scheme provides a constant factor approximation guarantee based on the number of belief points. In experiments on standard domains, the optimal approach provides significant speedup (up to 2 orders of magnitude) over the previous best optimal algorithm and is able to increase the number of belief points by more than a factor of 3. The approximation scheme also works well in practice, providing near-optimal solutions to the backup problem.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: [Dynamic programming, Multiagent systems, Intelligent agents]

General Terms

Algorithms, Theory

Keywords

Multiagent planning, DEC-POMDPs

1. INTRODUCTION

Decentralized partially observable MDPs (DEC-POMDPs) have emerged in recent years as an important framework for modeling sequential decision making by a team of agents [3]. Their expressive power makes it possible to tackle coordination problems in which agents must act based on different partial information about the environment to maximize a global reward function. Applications of DEC-POMDPs include multi-robot control such as coordinating the operation of planetary exploration rovers [2], coordinating firefighting robots [14], broadcast channel protocols [3] and target tracking by a team of sensor agents [13]. However, the modeling advantage comes with a price – solving a finite horizon DEC-POMDP optimally is NEXP-Complete [3], essentially rendering optimal approaches intractable.

Consequently, research has focused on approximate algorithms, of which point-based approaches have shown great promise in scaling up and finding good quality policies [17]. As in single agent POMDP point-based approaches [18, 15], the key idea is to compute a set of reachable beliefs using a heuristic and compute the value function by performing a sequence of point-based backups. However, the backup technique differs fundamentally in the multi-agent setting. In POMDPs, backups can be performed efficiently. However, due to the decentralized nature of a DEC-POMDP policy, naively performing such backups requires *exponential* effort in the number of observations [17].

Recently, there have been substantial efforts to solve this problem more efficiently because point-based backups constitute the core of any point-based algorithm. For example, the IMBDP algorithm [16] avoids exponential blowup by limiting the full backup to a fixed number of observations which are most likely to occur at the given belief point. In the MBDP-OC algorithm [4], an information-theoretic criterion is used to merge observations into sets while minimizing the total loss in solution quality. Dibangoye *et al.* [8] take a different approach to solving this problem optimally by using a branch-and-bound search. While in the worst case, the complexity remains the same, their approach is significantly faster than previous approaches in practice. Amato *at al.* [1] further improve the scalability of the previous algorithm by limiting the possible next step sub-policies using state reachability analysis.

In our work, we elicit the key reason for the underlying difficulty of performing decentralized backups. We show that performing the backup step optimally is *NP-Complete* by reduction from the NP-Complete team decision problem in control theory [19, 5]. Despite this negative result, we present an optimal algorithm by reformulating the backup problem as an instance of the weighted constraint satisfaction problem (WCSP). The optimal algorithm uses the state-of-the-art constraint solver AND/OR branch-and-bound search (AOBB) [11]. However, for AOBB to perform well, it is crucial to select a good heuristic. We use the existential directional arc consistency (EDAC) heuristic [6],

Cite as: Point-Based Backup for Decentralized POMDPs: Complexity and New Algorithms, Akshat Kumar, Shlomo Zilberstein, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp 1315-1322

which enables us to solve the backup problem significantly faster (up to 2 orders of magnitude) than the previous best algorithm PBIP [8, 1]. This allows us to increase the number of belief point by a factor of 3 resulting in increased solution quality. Our detailed experimental results also show that performing backups with the new algorithm takes a very small fraction of the total runtime, essentially shifting the computational bottleneck to evaluating and storing joint-policies.

Based on recent results on approximating team decision problems [5], we also present a poly-time approximate algorithm for the decentralized backup operation, guaranteeing a solution within a factor of $1/\mathsf{MaxTree}$ of the optimal value, where $\mathsf{MaxTree}$ denotes the maximum number of policies retained at each step per agent while performing bottom-up dynamic programming. In the experiments, we show that this approximation scheme performs near-optimally, compared to the WCSP-based optimal algorithm.

2. THE DEC-POMDP MODEL

In this section, we introduce the DEC-POMDP model and develop the optimality condition for the decentralized backup. For simplicity, we present the model for two agents, but it can be easily generalized to any number of agents.

The set S denotes the set of environment states, with a given initial state distribution b_0. The action set of agent i is denoted by A_i. The state transition probability $P(s'|s, a_1, a_2)$ depends upon the actions of both the agents. Upon taking the joint action $\langle a_1, a_2\rangle$ in state s, agents receive the joint reward $R(s, a_1a_2)$. Z_i is the finite set of observations for agent i. $O(s, a_1a_2, z_1z_2)$ denotes the probability $P(z_1z_2|s, a_1a_2)$ of agent 1 observing $z_1 \in Z_1$ and agent 2 observing $z_2 \in Z_2$ when joint action $\langle a_1, a_2\rangle$ was taken and resulted in state s.

A local policy for agent i is defined as a mapping from local observation histories to actions. A joint policy is a tuple of local policies, one per agent. The goal in a DEC-POMDP is to find the joint policy that maximizes the total expected reward over some given finite horizon T.

2.1 Point-based backup in DEC-POMDPs

The intractability of optimal POMDP algorithms can be attributed to planning for the complete belief space. DEC-POMDPs are further disadvantaged as the joint policy has to be represented explicitly (mapping from individual observation histories to actions) and cannot be extracted directly from the value function. The idea of planning for a finite set of belief points has lead to several successful point-based POMDP algorithms in recent years such as PBVI [15] and Perseus [18] among others. All these algorithms compute a set of reachable belief points using a heuristic such as the underlying MDP policy or using randomized trajectories [18]. The value function is computed over these beliefs by performing a sequence of point-based backups. A single backup step entails finding the best action for the given belief point and deciding which sub-policy (α-vectors for POMDPs) to execute upon receiving each of the possible observations. Although, the resulting value function may not be the optimal one, in practice it has been shown that these heuristics work reasonably well while keeping the solution technique tractable. For POMDPs, such point-based update can be performed efficiently using polynomial time in all problem parameters [15]. We develop below optimality conditions for point-based backup in DEC-POMDPs and show that performing this operation optimally is inherently more complex that in POMDPs.

In DEC-POMDPs, the joint value function cannot be described by a collection of α-vectors as in POMDPs. Instead, all existing point-based approaches use an explicit tree structured representation of the local policy for each agent, where a node denotes the action to be executed and edges correspond to the observations. Edges connect to subtrees, which are executed by the agent when the corresponding observation is received. For two agents, the joint-value of the trees p of agent 1 and q of agent 2 in the starting state s can be computed recursively as follows:

$$V(p,q,s) = R(s,a_p,a_q) \;+\; \sum_{z_1,z_2}\sum_{s'\in S} O(s',a_pa_q,z_1z_2) P(s'|s,a_p,a_q)V(p^{z_1},q^{z_2},s') \quad (1)$$

where a_p and a_q refer to the actions defined at the root of the trees p and q, edges corresponding to observations z_1 and z_2 connect to respective subtrees p^{z_1} and q^{z_2}. Consequently, the value for a belief point b can be defined as

$$V(p,q,b) = \sum_{s\in S} b(s)\cdot V(p,q,s) \quad (2)$$

As in POMDPs, a fixed number of policies are retained during each dynamic programming step–one per belief point. In the backup step, the best joint-policy $\langle p, q\rangle$ for a belief b is constructed by finding the joint action $\langle a_1, a_2\rangle$ to be performed at the root of $\langle p, q\rangle$ and determining for each agent, the sub-policy they will follow upon receiving the corresponding observations such that the policies p and q maximize the value $V(p, q, b)$. Since the value V depends on the joint policy, the best sub-policy for an agent cannot be determined in *isolation* of each other. Let us define decision rules $\delta_i^{a_i}$ for each agent i and every action a_i available to agent i. $\delta_i^{a_i}$ maps the observations of agent i to the set of available sub-policies for the next step when the immediate action executed is a_i. For example, $\delta_1^{a_1} : Z_1 \rightarrow P$, where P denotes the set of available sub-policies for agent 1. The dynamic programming equation can be written as follows:

$$V_t(b) = \max_{a_1,a_2,\delta_1^{a_1},\delta_2^{a_2}} \Big[\sum_{s\in S} R(s,a_1a_2)b(s) + \sum_{z_1,z_2} Pr(z_1z_2|a_1a_2,b) V(\delta_1^{a_1}(z_1),\delta_2^{a_2}(z_2),b')\Big] \quad (3)$$

where $b' = \tau(b, a_1a_2, z_1z_2)$ is the updated belief after agents take the joint action $\langle a_1, a_2\rangle$ and receive observations $\langle z_1, z_2\rangle$. It can be calculated in a straightforward manner using Bayes rule. Upon substituting its value, the final optimality condition is as follows:

$$V_t(b) = \max_{a_1,a_2,\delta_1^{a_1},\delta_2^{a_2}} \Big[\sum_{s\in S} R(s,a_1a_2)b(s) + \sum_{z_1,z_2}\sum_{s,s'} O(s',a_1a_2,z_1z_2) P(s'|s,a_1a_2)V(\delta_1^{a_1}(z_1),\delta_2^{a_2}(z_2),s')b(s)\Big] \quad (4)$$

The above equation does not admit efficient solutions and, as we show next, it is NP-Complete to find the optimal solution. Intuitively, the reason is that solving the above equation requires *optimization over functions* δ for each agent and the space of all possible mappings δ is exponential in the number of observations.

2.2 Complexity of decentralized backup

Before establishing the complexity, we introduce the NP-Complete team decision problem (TDP) [19, 5]. In TDP, there are two decision makers, each of which observes some local component of the system state. That is, if $(y_1, y_2) \in Y_1 \times Y_2$ is a system state, then agent 1 observes only y_1 and agent 2 observes y_2. The goal of the agents is to choose an action $(u_1, u_2) \in U_1 \times U_2$ which maximizes the joint reward $r(y_1, y_2, u_1, u_2)$ based only upon their local observation. Stated formally, the goal is to find decision rules $\gamma_1 : Y_1 \rightarrow U_1$ and $\gamma_2 : Y_2 \rightarrow U_2$ that maximize the expected reward, assuming each state is equally likely. That is,

$$\max_{\gamma_1,\gamma_2} \sum_{y_1,y_2} r(y_1, y_2, \gamma_1(y_1), \gamma_2(y_2))$$

THEOREM 1. *Solving the decentralized backup problem optimally is NP-Complete.*

PROOF. We reduce the TDP to a two agent DEC-POMDP $\mathcal{M}$ where taking one step backup is equivalent to solving the TDP. It is clear that the backup problem is in NP, because given a policy mapping δ_i, we can evaluate the R.H.S. of Eq. 4 in polynomial time.

There are $1 + |Y_1||Y_2|$ states in $\mathcal{M}$. s is the initial state s.t. $b_0(s) = 1$. The rest of the system states are denoted by $s_{y_1y_2} \forall (y_1, y_2) \in Y_1 \times Y_2$. The observation sets of the agents are: $Z_1 = Y_1$ and $Z_2 = Y_2$. The action space is defined as: $A_1 = U_1$ and $A_2 = U_2$. The state s transitions with equal probability to each state $s_{y_1y_2}$ regardless of the action taken and $P(s|s) = 0$. The observation probabilities are also independent of the action taken and observations identify the resulting state deterministically. That is, $O(s_{y_1y_2}, z_1z_2) = 1$ if $y_1 = z_1$ and $y_2 = z_2$, else it is zero. The reward for any joint action in the initial state s is 0 and the horizon for the problem is 2. The rest of the reward function will be defined shortly. First, note that the backup Eq. 4 can be written as

$$V_t(b_0) = \max_{\delta_1,\delta_2} \Bigg[\sum_{z_1,z_2} \sum_{s_{y_1y_2}} P(s_{y_1y_2}|s) O(s_{y_1y_2}, z_1z_2) \\ V(\delta_1(z_1), \delta_2(z_2), s_{y_1y_2}) \Bigg]$$

In the above equation, we used the fact that $R(s, \cdot) = 0$ by the problem construction and that any action can be taken at the initial belief b_0 as state transition and observation probabilities are independent of actions. As the problem horizon is 2, $V(\cdot, \cdot, \cdot)$ is simply the immediate reward R. Since observations identify the resulting state deterministically and $P(s_{y_1y_2}|s) = 1/|Z_1||Z_2|$, the equation can be further simplified as follows:

$$V_t(b_0) = \frac{1}{|Z_1||Z_2|} \max_{\delta_1,\delta_2} \left[\sum_{z_1,z_2} R(s_{z_1z_2}, \delta_1(z_1), \delta_2(z_2)) \right] \quad (5)$$

Now, we set the rest of the reward function such that

$$R(s_{z_1z_2}, a_1, a_2) = r(y_{z_1}, y_{z_2}, u_{a_1}, u_{a_2}) \forall s_{z_1z_2} \in Y_1 \times Y_2$$

where y_{z_i} and u_{a_i} are the counterparts of the DEC-POMDP in the given TDP instance. Clearly, the reduction from TDP to the DEC-POMDP has polynomial complexity.

Finally, we can show that there exists a solution to the TDP problem providing total reward W if and only if there exists a solution to the backup problem achieving a reward of $W/(|Z_1||Z_2|)$. This is straightforward to see as the functions γ_i from the TDP instance map directly to δ_i and vice-versa. The reward function for the DEC-POMDP is the same as the reward function for the TDP. $\square$

Next, we develop new algorithms for solving the decentralized backup problem efficiently and in a principled way. First, we detail the poly-time approximation scheme, which provides a constant factor quality guarantee. Then we describe the optimal approach based on WCSPs.

3. APPROXIMATE BACKUP

We have shown previously that $TDP \leq_p Backup$, therefore, $Backup \leq_p TDP$ by the property of NP completeness. We can further show that there exists an L-reduction (or the linear reduction) for the later, which preserves the approximability feature. That is, if there exists a α approximation algorithm for TDP, then there also exists an α approximation algorithm for the backup problem. We do not describe this reduction in detail, but directly describe the algorithm for the backup problem which provably gives a solution within $1/\mathsf{MaxTree}$ of optimal. For details of the original TDP approximation, we refer to [5]. The $\mathsf{MaxTree}$ parameter refers to the maximum number of available sub-policies (for either agent) for the next step while performing one step backup.

First, based on the optimality Eq. 4, we define the contribution of sub-policies $p \in P$ of agent 1, $q \in Q$ of agent 2 for a given joint action $\langle a_1, a_2 \rangle$ and observation $\langle z_1, z_2 \rangle$ as follows. We will omit mentioning the current belief b as long as it is unambiguous.

$$R^{a_1a_2}(z_1, z_2, p, q) = \sum_{s,s'} O(s', a_1a_2, z_1z_2) P(s'|s, a_1a_2) \\ V(p, q, s') b(s)$$

The key idea of the approximation scheme is as follows. It uses a heuristic to set the policies agent 1 will follow upon receiving each observation $z_1 \in Z_1$. Once the sub-policies for agent 1 are fixed, then a simple maximization rule allows us to find the best sub-policies for agent 2. Similar analysis can be done by considering agent 2 first and the best of the two becomes the solution to the backup problem providing a bounded approximation. The first step is the construction of a marginalized reward function for each observation z_1 and available sub-policy p of agent 1.

$$R_1^{a_1a_2}(z_1, p) = \sum_{z_2 \in Z_2, q \in Q} R^{a_1a_2}(z_1, z_2, p, q)$$

Another interpretation of the above function is that agent 2 follows every policy with equal probability upon receiving any observation, i.e., assuming a stochastic policy. Next, based on this reward function, we can fix the sub-policies of agent 1 for each observation z_1 as follows:

$$\delta_1^{a_1}(z_1) = \operatorname*{argmax}_{p \in P} R_1^{a_1a_2}(z_1, p)$$

Now, finding the best sub-policies for agent 2 is straightforward as follows:

$$\delta_2^{a_2}(z_2) = \operatorname*{argmax}_{q \in Q} \sum_{z_1} R^{a_1a_2}(z_1, z_2, \delta_1^{a_1}(z_1), q)$$

The above equation follows by realizing that once all the sub-policies for any one agent are fixed, then Eq. 4 allows us

Algorithm 1: *ApproxBackup*

```
1  $b \leftarrow$ selected belief for backup
2  $P \leftarrow$ available sub-policies for agent 1
3  $Q \leftarrow$ available sub-policies for agent 2
4  for $\langle a_1, a_2\rangle \in A_1 \times A_2, \langle z_1, z_2\rangle \in Z_1 \times Z_2, \langle p, q\rangle \in P \times Q$ do
5      $R^{a_1a_2}(z_1, z_2, p, q) = \sum_{s,s'} O(s', a_1a_2, z_1z_2)P(s'|s, a_1a_2)V(p, q, s')b(s)$
6  Calculate marginalized reward function for agent 1–
7  for $\langle a_1, a_2\rangle \in A_1 \times A_2$ do
8      for $z_1 \in Z_1$, $p \in P$ do
9          $R_1^{a_1a_2}(z_1, p) = \sum_{z_2\in Z_2,q\in Q} R^{a_1a_2}(z_1, z_2, p, q)$
10
11 $bestSol = -\infty$
12 for $\langle a_1, a_2\rangle \in A_1 \times A_2$ do
13     $\delta_1^{a_1}(z_1) = \text{argmax}_{p\in P} R_1^{a_1a_2}(z_1, p)\ \forall z_1 \in Z_1$
14     $\delta_2^{a_2}(z_2) = \text{argmax}_{q\in Q} \sum_{z_1} R^{a_1a_2}(z_1, z_2, \delta_1^{a_1}(z_1), q) \forall z_2 \in Z_2$
15     $currentVal$ = calculate $V(b)$ using Eq. 4
16     if $currentVal > bestSol$ then
17         $bestSol \leftarrow currentVal$
18         $a_i^\star \leftarrow a_i,\ \delta_i^{a_i^\star} \leftarrow \delta_i^{a_i}\ \forall i \in \{1, 2\}$
19
20 Repeat block 6-19 for agent 2
21 return best $a_i^\star, \delta_i^{a_i^\star}\ \forall i \in \{1, 2\}$
```

to find the optimal sub-policy for the other agent for each observation *independently*. Also, note that this process can be performed recursively by iterating over agents until an equilibrium is reached. This is also referred to as *person-by-person* optimal policies in the context of control theory [5]. However, the approximation guarantee does not depend on this enhancement. One caveat to this approach is that the approximation guarantee requires the rewards or the sub-policy contributions to be positive. We note that locally optimal policies have also been explored in the context of DEC-POMDPs [12]. However, they optimize the complete policy whereas we focus on a single backup step.

Once, we have calculated such decision rules for each joint action, we can find the best joint action by plugging in the decision rules into Eq. 4. A similar analysis can be performed by considering agent 2 first and finding the best joint policy. The best of the two cases is used as the final solution of the backup problem.

Algorithm 1 shows how approximate backup is performed. The input to this algorithm is the belief point and the set of sub-policies for the next step for each agent. The remaining steps are self explanatory in light of the above discussion. Next, we analyze the complexity of this algorithm.

3.1 Complexity

Calculating the sub-policy contributions in Algorithm 1 takes $O(|A|^2|Z|^2|P||Q||S|^2)$ time, where $|A|$ and $|Z|$ denote the maximum size of the action and observation sets. This can be further simplified to $O(|A|^2|Z|^2\mathsf{MaxTree}^2|S|^2)$ by realizing that MaxTree bounds the size of policy sets P and Q. Calculating the marginalized reward function for any agent takes $O(|A|^2|Z|^2\mathsf{MaxTree}^2)$ time. Finally, finding out the best policy (block 12-18) takes $O(|A|^2|Z|^2\mathsf{MaxTree})$ time.

Therefore, the complexity of Algorithm 1 is quadratic in the problem parameters – $O(|A|^2|Z|^2(\mathsf{MaxTree}^2 + |S|^2))$. Most importantly, it is no longer exponential in the observation set size and can be implemented efficiently for large DEC-POMDP instances. Also, it is worth noticing that the approximation factor 1/MaxTree does not depend upon the number of observations, which is the root of the NP-Completeness of the problem. Tsitsiklis and Athans [19] show that the TDP problem remains NP-Hard even when the action set size is 2. Thus, when the MaxTree parameter is 2, this approach provides 1/2 approximation to this NP-Complete problem. In the next section, we present the optimal approach for solving the backup problem.

4. OPTIMAL BACKUP APPROACH

The optimal algorithm for the decentralized backup leverages recent advances in the weighted constraint satisfaction (WCSP) literature by reformulating the backup problem as finding the least cost solution of a WCSP instance. We briefly introduce the weighted constraint satisfaction problem below; further details can be found in [6].

A WCSP is a tuple $\langle \mathcal{X}, \mathcal{D}, \mathcal{C}, k\rangle$. $\mathcal{X} = \{X_1, \ldots, X_n\}$ is a set of variables. $\mathcal{D}$ is a set of domains D_i for each variable X_i. D_i is discrete and denotes all possible assignments to the variable x_i. $\mathcal{C}$ is a set of constraints. Each constraint C_S is defined over a subset of variables $S \subseteq \mathcal{X}$ and maps the tuples corresponding to assignments on S to a real valued cost. For example, $C_{ij} : D_i \times D_j \rightarrow [0 \ldots k]$, where C_{ij} is a constraint over variables X_i and X_j and k denotes the upper bound on the cost of any assignment tuple. When a constraint assigns a cost k to any assignment, it implies that the assignment is forbidden, else it is allowed with the corresponding cost.

The goal is to find the complete assignment X to variables such that the global cost is minimized. $X[S]$ represents the projection of tuple X over variables in S. The total cost is $\sum_{C_S\in\mathcal{C}} C_S(X[S])$.

4.1 Decentralized backup as a WCSP

In this section, we describe the reformulation of the backup problem as a WCSP instance. A constraint-based formulation has been recently used for speeding up DEC-POMDP algorithms [9]. However, that formulation is not applicable to the backup problem we target. First, we note that, for each joint action $\langle a_1, a_2\rangle$, the optimality equation (Eq. 4) can be written as follows:

$$V_t^{a_1a_2}(b) = \sum_{s\in S} R(s, a_1a_2)b(s) + \max_{\delta_1,\delta_2} \sum_{z_1,z_2} \sum_{s,s'} O(s', a_1a_2, z_1z_2) P(s'|s, a_1a_2)V(\delta_1(z_1), \delta_2(z_2), s')b(s)$$

If we solve the above equation optimally, then finding $V_t(b)$ is easy as it requires iterating over all the joint actions, which has polynomial complexity. In the above equation, only the second summation depends on the decision rule δ_i. Therefore, the optimization problem becomes

$$\max_{\delta_1,\delta_2} \sum_{z_1,z_2} \sum_{s,s'} O(s', a_1a_2, z_1z_2)P(s'|s, a_1a_2) V(\delta_1(z_1), \delta_2(z_2), s')b(s) \quad (6)$$

We seek to optimize the above equation by reformulating it as a WCSP for each joint action $\langle a_1, a_2\rangle$. The WCSP parameters are detailed below.

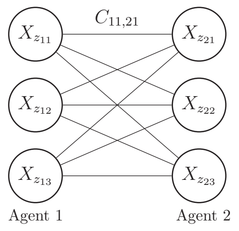


Figure 1: Primal graph of a WCSP for the backup problem. Each agent has three observations

- $\mathcal{X}$ **Variables:** We create one variable for each observation of each agent. For example, if $z_{1i} \in Z_1$ is a possible observation for agent 1, then variable $X_{z_{1i}}$ is created.
- $\mathcal{D}$ **Domain:** The domain of all the variables corresponding to an agent is the set of all next step sub-policies available for that agent. For example, if P denotes the set of sub-policies for agent 1, then $D_{z_{1i}} = \{p | p \in P\} \; \forall z_{1i} \in Z_1$.
- $\mathcal{C}$ **Constraints:** A constraint is created for every pair of observations from agent 1 and 2. That is, we create a binary constraint $C_{1i,2j}$ between variables $X_{z_{1i}}$ and $X_{z_{2j}}$ for every pair of observations $z_{1i} \in Z_1$ and $z_{2j} \in Z_2$.
- C **Valuations:** The valuation for each constraint $C_{1i,2j} \in \mathcal{C}$ is defined as in Eq. 6.

$$C_{1i,2j}(p,q) \;=\; \alpha - \sum_{s,s'} O(s', a_1a_2, z_{1i}z_{2j}) P(s'|s, a_1a_2) V(p,q,s')b(s)$$

where α is a large positive constant which is used to transform the maximization objective of Eq. 6 to minimizing the WCSP cost. Intuitively, the second part of the above equation (summation over states) represents the value accrued when agent 1, upon receiving observation z_{1i}, follows the sub-policy p and agent 2, upon receiving observation z_{2j}, follows the sub-policy q.

It is easy to see that minimizing the objective function of this WCSP, $\sum_{C_{1i,2j}} C_{1i,2j}$, is equivalent to maximizing Eq. 6. The complete assignment X represents the decision rules δ_i for each agent and the optimal assignment solves Eq. 6. The global cost of assignment X is given by

$$\sum_{C_{1i,2j}} C_{1i,2j} \;=\; |Z_1||Z_2|\alpha - \sum_{z_{1i},z_{2j}} \sum_{s,s'} O(s', a_1a_2, z_{1i}z_{2j}) P(s'|s,a_1a_2)V(X[X_{z_{1i}}], X[X_{z_{2j}}], s')b(s)$$

We can visualize this WCSP by using its primal graph. The primal graph of a WCSP with binary constraints is a graph whose nodes are the variables of the problem, and each edge connects a pair of variables that occur together in the scope of a constraint function. Fig. 1 shows the primal graph of the WCSP for a backup instance when each of the two agents has three observations. Each edge represents a constraint.

4.2 Solving the WCSP

Unfortunately, optimally solving a WCSP is NP-Hard. However, as constraint reasoning has numerous practical applications, many algorithms exist which can solve them efficiently either using dynamic programming or search techniques. It has been shown that if the primal graph of a WCSP has bounded tree-width, then the WCSP can be solved efficiently using dynamic programming with the bucket elimination algorithm [7]. However, the WCSP instance for the backup problem is a complete bipartite graph as shown in Fig. 1. For such graphs, the tree-width is $O(|Z|)$ regardless of the variable ordering used. Since the bucket elimination algorithm has complexity exponential in the tree-width, it cannot scale well with the number observations. Therefore, we used a search-based solver, AOBB [11], which uses a heuristic function to prune a large part of the search space. The heuristic function provides a lower bound for the WCSP at each step of the search. We used the state-of-the-art heuristic *existential directional arc consistency* (EDAC) [6]. We tried other heuristics too, such as the mini buckets [11], however EDAC outperformed the others by a significant margin, sometimes expanding an order of magnitude less nodes. Next, we explain key differences between our approach and the previous optimal approach PBIP [8], describe the EDAC heuristic and explain why it outperforms PBIP.

4.2.1 *Comparison with PBIP*

While PBIP is also a search algorithm [8], our approach is fundamentally different. The key differences lie in how the search process is structured and how the heuristic function is computed. In PBIP, a node represents a partial joint-policy tree, where the sub-policies that agents will follow are only specified for *some* joint observations and not for others. Expanding a fringe node requires selecting an unspecified joint observation and generating all possible successors by attaching all possible subtrees to this joint-observation, which number $\mathsf{MaxTree}^2$. This is one aspect where our formulation differs from PBIP. In our approach, the search *does not* take place over joint observations, but over individual observations of an agent (see Fig. 1). Consequently, the number of successors of a node is only $\mathsf{MaxTree}$. The depth of the search tree in our case is $2|Z|$, whereas in PBIP it is $|Z|$. However, the worst-case complexity remains the same because the branching factor in our case is $\mathsf{MaxTree}$ and in PBIP, it is $\mathsf{MaxTree}^2$. By searching over the space of joint observations, PBIP loses the structure present in the backup problem whereas our approach explicitly represents this structure using a constraint graph (Fig. 1). The heuristic function we use, EDAC, explicitly utilizes this constraint graph and produces much tighter bounds than PBIP.

4.2.2 *Details of the EDAC heuristic*

There is also a significant difference between the heuristic function used in PBIP and the EDAC heuristic. In PBIP, the upper bound of a backup problem is calculated by selecting the best joint sub-policy for each pair of joint observations where the sub-policy is unspecified. We show using an example, that this heuristic may lead to much worse bounds compared with EDAC. First, we introduce some basic concepts related to EDAC. The details can be found in [6].

EDAC works by maintaining local consistency properties. Let C_ϕ denote the lower bound of a WCSP instance, initially $C_\phi = 0$. A variable X_i is node consistent (NC) if for all values $a \in D_i$, $C_\phi + C_i(a) < k$, and $\exists a \in D_i$ such that $C_i(a) = 0$, where C_i is the unary constraint on X_i, k is the upper bound of the WCSP. A WCSP is $NC^\star$ if every variable is node consistent. The rationale behind node consistency is that if for a variable X_i, $C_i(a) > 0 \; \forall a \in D_i$, then the lower bound C_ϕ of the WCSP can be incremented by α, where

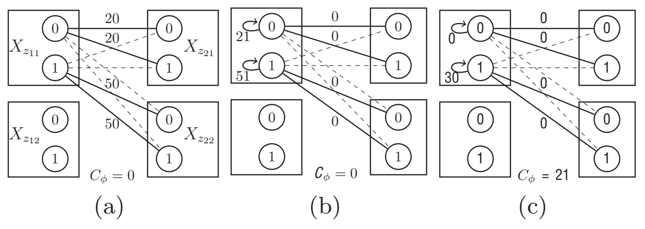


Figure 2: (a) Backup instance as a WCSP, (b) Arc consistency, and (c) Node consistency

$\alpha = \min_{a \in D_i} C_i(a)$, and the WCSP can be transformed into an equivalent problem. Another property is arc consistency ($AC^\star$). Given a binary constraint C_{ij}, $b \in D_j$ provides a simple support for $a \in D_i$ if $C_{ij}(a, b) = 0$. A variable x_i is arc consistent if each $a \in D_i$ has simple support in every constraint C_{ij}. A WCSP is $AC^\star$ if every variable is node and arc consistent. The intuition behind this property is that, if a variable is not arc consistent, then by enforcing the arc consistency, some costs from its binary constraints can be projected on its unary constraint, which may violate its node consistency. Then by enforcing node consistency again, the lower bound can be increased further. We illustrate these consistency techniques using Fig. 2.

Fig. 2(a) shows the backup problem for two agents as a WCSP with each agent having 2 observations. Each rectangle represents a variable corresponding to an observation. Domain values are shown in small circles implying each agent has two sub-polcies to choose from. Lines across variables denote valuations for that constraint. Each dashed line has valuation 1, the rest are shown alongside the solid lines. Variable $X_{Z_{12}}$ has zero valuations for each of its constraints, so they are not shown. Fig. 2(b) shows the equivalent WCSP after enforcing arc consistency. Notice that for constraint $C_{11,21}$ and variable value $X_{z_{11}} = 0$, every value of variable $X_{z_{21}}$ provides valuation 20. Therefore, this cost can be subtracted from this constraint and transferred to the unary constraint on $X_{z_{11}}$. Similarly, a cost of 1 is incurred for the constraints $C_{11,22}$ when $X_{z_{11}} = 0$. Hence, it is also added to the unary constraint on $X_{z_{11}} = 0$ resulting in total cost of 21. Similarly, the unary constraint on $X_{z_{11}} = 1$ has cost 51. The resulting WCSP (in Fig. 2(b)) is not node consistent as no domain value of variable $X_{z_{11}}$ has unary cost 0. Therefore, we add the minimum of unary constraint cost ($= 21$) to the WCSP lower bound C_ϕ and subtract it from the respective valuations. Fig. 2(c) shows the $NC^\star$ WCSP with a lower bound of 21. This bound also turns out to be the optimal cost.

Now consider for comparison how the PBIP-like heuristic performs. It will take the least cost for each joint observation and sum them up for calculating the lower bound of the instance. For observations $\langle z_{11}, z_{21} \rangle$, the least cost is 1 when $X_{z_{11}} = 1$, $X_{z_{21}} = 0$. Similarly, for observations $\langle z_{11}, z_{22} \rangle$, the least cost is also 1. Therefore, the heuristic will estimate the lower bound to be 2, which is far from the accurate bound produced by arc consistency. The main reason for the inaccuracy of the PBIP heuristic is that it ignores the dependencies among multiple observations by considering only a group of 2 at a time, whereas arc consistency propagates constraints to estimate a much tighter lower bound. This is also reflected in the computation cost of the heuristics – for PBIP, it is $O(|Z|^2 \mathsf{MaxTree}^2)$ and for EDAC, it is $O(|Z|^2 \mathsf{MaxTree}^2 \max\{|Z|\mathsf{MaxTree}, k\})$. Although, EDAC is computationally more expensive than PBIP, its accuracy pays off well by exploring very few nodes to find the optimal solution as shown in the experiments section.

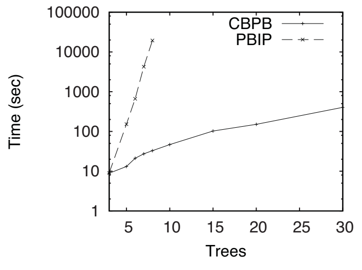
We conclude this section by highlighting the generality of the approximate and the optimal backup techniques we have developed. Both of these approaches are independent of the DEC-POMDP algorithm used and can be incorporated into any point-based solver. Furthermore, these approaches can be adapted with only minor modifications to other models of decentralized decision making such as Bayesian games, which have been used to solve DEC-POMDPs [10, 14]. The experiments in the next section confirm that by combining diverse concepts from control theory and constraint satisfaction, the scalability of existing approaches to decentralized sequential decision making can be improved dramatically.

5. EXPERIMENTS

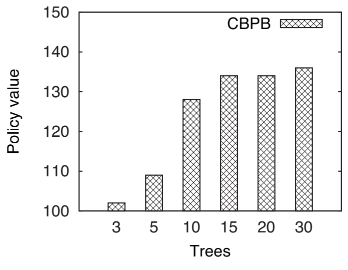
We incorporated our backup algorithms into the state-of-the-art point-based solver MBDP [17]. We compared both the WCSP-based optimal algorithm named *Constraint based point backup* (CBPB) and the approximate scheme *Team decision problem based policy iteration* (TDPI) with the best existing approach PBIP [8], which also solves the backup problem optimally and is built upon MBDP. We used the latest version of PBIP that uses incremental policy generation [1] to further enhance its performance. Experiments were performed on a Linux machine with 2GB RAM and 2.6GHz CPU. We used the largest DEC-POMDP benchmarks: the box pushing problem [16] and the stochastic Mars rover [1]. Each data point is an average over 10 runs.

The aim of our experiments is threefold. First, we demonstrate the computational advantages of CBPB over the PBIP algorithm in terms of execution time and the ability to increase the number of belief points – CBPB provides more than 2 orders of magnitude of speedup for some settings and is about an order of magnitude faster on average, and can increase the number of belief points (the $\mathsf{MaxTree}$ parameter) by more than a factor of 3. Second, we show the scalability of CBPB by explicitly comparing the actual time required for performing all the backups versus the total execution time. We show that as we increase the $\mathsf{MaxTree}$ parameter, the bottleneck becomes evaluating the joint-policies. CBPB can still solve the larger search problem without considerable overhead expanding sub-hundred nodes on average. Finally, we show the accuracy of the TDPI approximation scheme, which produces near-optimal solution to the backup problem and has polynomial complexity.

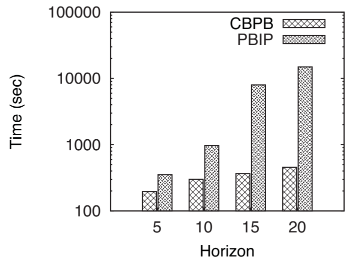
Fig. 3(a) shows a comparison of CBPB with PBIP on the Box pushing domain ($|S| = 100$, $|A_i| = 4$, $|Z_i| = 5$) with varying $\mathsf{MaxTree}$ values for horizon 10. Clearly, CBPB provides significant savings over PBIP, whose time requirements become excessive very quickly upon increasing the number of belief points. For $\mathsf{MaxTree} = 8$, CBPB is about 500 times faster than PBIP, which takes over $19,000$ sec, whereas CBPB takes only 32 sec. Furthermore, CBPB can scale up the number of belief points to 30 whereas PBIP cannot scale above 8 belief points. Fig. 3(b) shows how solution quality is impacted by increasing the number of belief points. Overall, when $\mathsf{MaxTree}$ is increased from 3 to 30, solution quality does increase significantly from 102 to 135. However, the rate of increase with belief points is slow, which indicates that the number of beliefs has to be increased further to gain higher solution quality. This observation also



(a) Box pushing: Time comparisons



(b) Box pushing: Solution quality



(c) Mars rover: Time comparisons

Figure 3: Comparisons with PBIP

favors CBPB, which can potentially increase this parameter further as highlighted in the next part of the experiments.

Fig. 3(c) shows time comparisons with PBIP on one of the largest DEC-POMDP domains – stochastic Mars rover ($|S| = 256$, $|A_i| = 6$, $|Z_i| = 8$) with varying horizon and MaxTree = 3. Again, we observe that as the horizon is increased from 5 to 20, the speedup provided by CBPB increases significantly. At horizon 20, CBPB is about 33 times faster than PBIP. For Mars rover too, CBPB can increase the belief points from 3 to 10 as shown in Table 1. The runtime of CBPB does increases, but we will show later that this increase is largely due to the overhead of evaluating joint-policies in MBDP. The actual search time remains a tiny fraction of the total time. Solution quality increases as well by increasing MaxTree. For horizon 20, with MaxTree = 3 the best policy value is 37.8; with MaxTree = 10, it increases to 43.6. For this domain, we could not increase MaxTree further, as simply storing and evaluating all joint policies would exceed the system RAM of 2GB.

For the next set of experiments, we emphasize the scalability of CBPB by explicitly comparing the actual time required for performing all the backups versus the total execution time (all time units in sec). Table 2 shows the total search time (time required by the WCSP solver) over all point-based backups and the total execution time of CBPB for box pushing for horizon 10. Clearly, even with increased MaxTree, the total search time remains a small fraction of the total time. This is because the average number of nodes expanded by CBPB for each backup (shown in the last column) remains below 100 for each setting. Table 3 shows similar results for the Mars rover domain with MaxTree=10. The overall execution time for CBPB is relatively high compared to box pushing as the rover domain's state and observation space is much larger than box pushing and evaluating joint-policies is much more expensive. But, notice that performing backups requires only a tiny fraction of the total time. Nodes expanded per backup instance also remain low. These results are a further testimony to the accuracy of the EDAC heuristic and show that if the memory limitations of MBDP are overcome (using more efficient data structures), then CBPB can scale to large numbers of belief points.

For the last set of experiments, we measure the accuracy of the approximate scheme TDPI. TDPI works by setting the policies of one agent using a heuristic approach and then sequentially optimizing the policy of the other agent until no improvement is possible. A natural yardstick to gauge the accuracy of the TDP heuristic is to set the policy of one agent randomly and then sequentially optimize the policy of the other agent similar to TDPI. Fig. 4(a) shows the comparison among random, TDPI and CBPB for the box pushing domain with horizon 10 and varying MaxTree values. For random and TDPI, policies of each agent were optimized recursively until convergence. As expected, CBPB achieved higher solution quality, but surprisingly the random heuristic did not perform much worse than TDPI. We found the reason for this was the sequential optimization of agents' policies until convergence to the local optima. Since the quality guarantee of the TDPI heuristic does not depend on this sequential or recursive optimization, a better criterion would be to use just one step of policy optimization, where the policy of one agent is fixed either randomly or using TDPI and the policies of other agent are set using simple maximization.

Fig. 4(b) shows the result of this comparison. The difference between random and TDPI is much more pronounced in this graph. The quality of the random heuristic drops sharply compared with the previous results, whereas TDPI provides more or less similar solution quality as with the sequential optimization. This is good news as it shows that the TDPI heuristic can find near-optimal policies using just one step maximization for all MaxTree settings. Moreover, the random heuristic relies on sequential optimization for local convergence and the quality of this local optima can be arbitrarily bad. Since TDPI is not so sensitive to this process, theoretically, TDPI represents a better heuristic than

MaxTree \ Horizon	5	10	15	20
3	197.2	301.8	363.4	457
10	276.6	1250.3	2000.6	2729

Table 1: Mars Rover: Execution time

MaxTree	Search Time	Total Time	Nodes Exp.
5	2.4 (18.3%)	13.1	27.6
10	6.1 (13.1%)	46.7	34.8
15	15.9 (15.5%)	102.5	39.5
20	24.6 (16.3%)	151.1	80.5
30	60 (14.8%)	404.2	82.1

Table 2: Box pushing: Search time Vs. Total time

Horizon	Search Time	Total Time	Nodes Exp.
5	8.1 (2.9%)	276.6	37.4
10	22 (1.8%)	1250.3	55.3
15	46.2 (2.3%)	2000.6	91.4
20	68.6(2.5%)	2729	94.5

Table 3: Mars Rover: Search time Vs. Total time

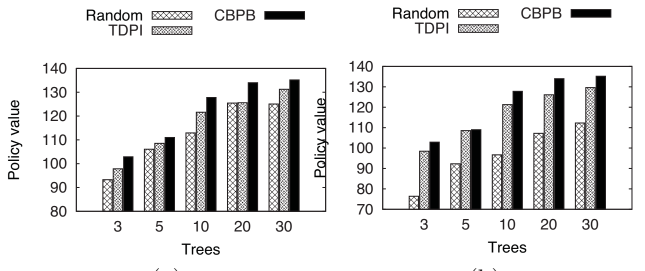


Figure 4: Box pushing solution quality: (a) Recursive, and (b) Single-step

random. For the Mars rover domain too, TDPI shows similar performance achieving solution quality close to CBPB. For MaxTree = 3 and horizon 10, CBPB achieves a quality of 22.01 and TDPI achieved 16.9. Simiarly, for horizon 20, CBPB achieved a quality of 37.8 and TDPI followed closely at 32.4. These results show that when the optimal solution is hard to attain, TDPI can be adopted as a principled heuristic approach for solving the backup problem.

6. CONCLUSION

Point-based backup constitutes the core of state-of-the-art DEC-POMDP solvers. We have shown that performing backups in the decentralized setting is NP-Hard, highlighting another stark challenge that multi-agent sequential decision making poses. Despite this negative worst-case result, we presented an efficient and scalable optimal algorithm as well as a principled approximation scheme. Our optimal algorithm reformulates the backup problem as a weighted constraint satisfaction problem and uses state-of-the-art constraint optimization techniques to solve it efficiently. In the experiments, we show that our optimal approach is highly scalable as it expands very few nodes in order to find the optimal solution even for large backup problems. Our approach provides orders of magnitude of speedup in the policy computation over the previous best algorithm, and it increases the number of belief points by a significant factor. These improvements are crucial in order to obtain better solution quality in larger DEC-POMDPs. The approximate scheme provides near-optimal solutions to the backup problem as well as a worst-case quality guarantee.

Overall, our results show that a significant bottleneck in DEC-POMDP approximation methods can be addressed using diverse techniques from constraint satisfaction and control theory. Finally, we note that the algorithms presented in this paper are general in the sense that they can be incorporated into any point-based DEC-POMDP solver with minor modifications, and have significant potential to scale up other models of multi-agent decision making such as Bayesian games.

Acknowledgments

Support for this work was provided in part by the National Science Foundation Grant IIS-0812149 and by the Air Force Office of Scientific Research Grant FA9550-08-1-0181.

7. REFERENCES

[1] C. Amato, J. Dibangoye, and S. Zilberstein. Incremental policy generation for finite-horizon DEC-POMDPs. In *Proc. of the Nineteenth International Conf. on Automated Planning and Scheduling*, pages 2–9, 2009.

[2] R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman. Solving transition independent decentralized markov decision processes. *Journal of Artificial Intelligence Research*, 22:423–455, 2004.

[3] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27:819–840, 2002.

[4] A. Carlin and S. Zilberstein. Value-based observation compression for DEC-POMDPs. In *Proc. of the Seventh International Joint Conf. on Autonomous Agents and Multiagent Systems*, pages 501–508, 2008.

[5] R. Cogill and S. Lall. An approximation algorithm for the discrete team decision problem. *SIAM Journal on Control and Optimization*, 45(4):1359–1368, 2006.

[6] S. de Givry, F. Heras, M. Zytnicki, and J. Larrosa. Existential arc consistency: Getting closer to full arc consistency in weighted CSPs. In *Proc. of the International Joint Conf. on Artificial Intelligence*, pages 84–89, 2005.

[7] R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113(1-2):41–85, 1999.

[8] J. S. Dibangoye, A.-I. Mouaddib, and B. Chaib-draa. Point-based incremental pruning heuristic for solving finite-horizon DEC-POMDPs. In *Proc. of the Eighth International Joint Conf. on Autonomous Agents and Multiagent Systems*, pages 569–576, 2009.

[9] A. Kumar, S. Zilberstein. Constraint-based dynamic programming for decentralized POMDPs with structured interactions. In *Proc. of the Eighth International Joint Conf. on Autonomous Agents and Multiagent Systems*, pages 561–568, 2009.

[10] R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun. Approximate solutions for partially observable stochastic games with common payoffs. In *Proc. of the Third International Joint Conf. on Autonomous Agents and Multiagent Systems*, pages 136–143, 2004.

[11] R. Marinescu and R. Dechter. AND/OR branch-and-bound search for graphical models. In *Proc. of the International Joint Conf. on Artificial Intelligence*, pages 224–229, 2005.

[12] R. Nair, M. Tambe, M. Yokoo, D. Pynadath, and S. Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proc. of the International JointConf. on Artificial Intelligence*, pages 705–711, 2003.

[13] R. Nair, P. Varakantham, M. Tambe, and M. Yokoo. Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. In *Proc. of the Twentieth National Conf. on Artificial Intelligence*, pages 133–139, 2005.

[14] F. A. Oliehoek, M. T. J. Spaan, and N. A. Vlassis. Optimal and approximate q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research*, 32:289–353, 2008.

[15] J. Pineau, G. Gordon, and S. Thrun. Anytime point-based approximations for large POMDPs. *Journal of Artificial Intelligence Research*, 27:335–380, 2006.

[16] S. Seuken and S. Zilberstein. Improved memory-bounded dynamic programming for decentralized POMDPs. In *Proc. of the Twenty-Third Conf. on Uncertainty in Artificial Intelligence*, 2007.

[17] S. Seuken and S. Zilberstein. Memory-bounded dynamic programming for DEC-POMDPs. In *Proc. of the Twentieth International Joint Conf. on Artificial Intelligences*, pages 2009–2015, 2007.

[18] M. Spaan and N. Vlassis. Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195–220, 2005.

[19] J. N. Tsitsiklis and M. Athans. On the complexity of decentralized decision making and detection problems. *IEEE Trans. on Automatic Control*, 30(2):440–446, 1985.