A Novel Point-based Algorithm for Multi-agent Control Using the Common Information Approach

Dengwang Tang, Ashutosh Nayyar, Rahul Jain

***Abstract*— The Common Information (CI) approach provides a systematic way to transform a multi-agent stochastic control problem to a single-agent partially observed Markov decision problem (POMDP) called the coordinator's POMDP. However, such a POMDP can be hard to solve due to its extraordinarily large action space. We propose a new algorithm for multi-agent stochastic control problems, called coordinator's heuristic search value iteration (CHSVI), that combines the CI approach and point-based POMDP algorithms for large action spaces. We demonstrate the algorithm through optimally solving several benchmark problems.**

I. Introduction

Multi-agent control problems arise in a number of applications including in teams of autonomous agents or robots carrying out a mission, in communication networks with multiple users and in distributed systems like the smart grid. The problem typically involves multiple agents with potentially different information taking actions and interacting with a dynamic system. In cooperative multi-agent problems, the agents' goal is to optimize a shared performance metric.

In this work, we assume that a control strategy can be determined offline collectively before interacting with the system (or *centralized planning and decentralized execution* as in Dec-POMDP literature). Even with centralized planning, multi-agent control problems can be significantly more difficult than their single-agent counterpart due to a variety of reasons: (1) due to the interdependent nature of all agents' information, actions, and the system evolution dynamics, we need to determine the strategies of all agents at the same time; (2) the decision making process of an agent involves not only the estimation of the underlying state but also the estimation of other agent's information, since it is crucial to predict other agents' actions. By the same token, an agent also needs to understand what other agents may believe of the information and action of herself and so on, resulting in the need to form a complicated hierarchy of beliefs.

There are several structural approaches to transform or decompose a multi-agent control problem into single-agent control problems. One of these approaches is the common information (CI) approach [1]. In this approach, a multi-agent control problem is transformed into a single-agent partially observed Markov decision problem (POMDP) by assuming the presence of a fictitious player, called *the coordinator*. At each time, the information available for each agent is partitioned into two parts, the *common information* and the *private information*. At each time, instead of letting each individual agent decide on their actions, the coordinator selects a *prescription* for each agent, which is a mapping from that agent's private information to its actions. The choice of prescription is based solely on the common information. In principle, the coordinator's problem can be solved using single-agent POMDP algorithms.

Single-agent POMDPs are nevertheless not easy to solve. Exact value iteration methods of solving POMDPs through updating a set of support vectors of the value function, namely α-*vectors*, were introduced in [2], [3]. However, these methods were practical only for problems with very few states [4]. A class of approximate solution methods, called *point-based* methods [5], [6], [7] have been quite successful in approximately solving POMDP problems with hundreds or thousands of states (e.g. PBVI [5] is the first algorithm to demonstrate great performance on the 870-state *Tag* problem. HSVI2 [8] is capable of tackling the $12,545$-state *RockSample*[7,8] problem). These methods are built on the following premise: Instead of computing all α-vectors, one can efficiently approximate the Bellman update through *point-based backup* procedures, i.e., computing a relatively small subset of α-vectors representing the gradients of the updated value function at certain belief points (see [4] for a comprehensive survey and a tutorial on such methods).

However, compared to a typical single-agent POMDP, a coordinator's POMDP has an *astronomically* large action space, since its actions are mappings. (Even a seemingly small problem like `DecTiger` (2,1,β) (see Appendix A.1) has millions of actions.) The huge action space creates major challenges in the use of state-of-the-art point-based methods to coordinator's POMDPs because: (1) The backup procedure requires solving discrete optimization problems over the action space. (2) The belief exploration processes of certain point-based methods [6], [7] also require solving discrete optimization problems over the action space. Therefore, even though the CI approach transforms a multi-agent problem into a single-agent one, there's still a need for a specialized algorithm to solve these specialized single-agent problems.

In this work, we present a new algorithm for multi-agent control that combines the CI approach with point-based POMDP solution methods for large action spaces. We demonstrate the performance of the new algorithm on several benchmark problems with huge action spaces.

Related Work: The CI approach [1] has been applied

Shorter version submitted to the 62nd IEEE Conference on Decision and Control (CDC2023).

This work is supported by ONR award N00014-20-1-2258 and NSF awards ECCS-2025732 and ECCS-1750041.

D. Tang, A. Nayyar, and R. Jain are with Ming Hsieh Department of Electrical and Computer Engineering, University of Southern California, Los Angeles, CA 90089-2560. Email: {dengwang, ashutosh.nayyar, rahul.jain}@usc.edu

in many settings. In [9], [10], the authors applied the approach to establish structural results for certain classes of multi-agent communication problems. In [11], the authors extended the approach to include the compression of private information. In [12], [13], [14], the authors proposed the idea of combining the CI approach with approximate planning and learning methods. While many theoretical results have been developed using the CI approach [1], [10], [11], [14], there is a lack of efficient planning algorithms based on this approach. As a result, empirical results have been established either in the case of very small private information spaces [13], or through machine learning techniques [15].

The model of Dec-POMDP is a multi-agent extension of POMDP that has been studied extensively (see [16] for a survey), where point-based methods have been applied (e.g. [17], [18], [19]). In many Dec-POMDP algorithms, the policy are represented as policy trees [19], whose size grows exponentially in time horizon. Therefore, such algorithms require a huge amount of memory and can only solve small finite horizon problems. There have also been Dec-POMDP algorithms designed for infinite horizon problems [20], [21]. However, those algorithms do not provide an optimality guarantee.

Point-based methods have been applied to multi-agent problems with partial history sharing. For example, in [22], the authors applied point-based methods to finite horizon problem with communication between agents. However, to the best of our knowledge, our work is the first to combine the CI approach with point-based POMDP methods on general infinite-horizon multi-agent problems.

Contributions: (1) In the context of works related to the CI approach, we present the first practical algorithm to solve a general coordinator's POMDP with large action spaces. (2) In the context of the Dec-POMDP literature, we provide a memory-efficient anytime algorithm for infinite horizon problems with an upper and lower bound for the value of the output policy.

II. Preliminaries

A. Problem Formulation

1) Multi-agent Control with Partial History Sharing: We consider an infinite-horizon multi-agent control model characterized by a tuple $\mathcal{E} = (\mathcal{I}, \mathcal{S}, \mathcal{A}, \mathcal{M}, \mathcal{O}, b_0, \mathbb{P}, r, \beta)$ where $\mathcal{I}$ is a finite set of agents;

- $\mathcal{S}$ is a finite set representing the state space;
- $\mathcal{A} = \prod_{i\in\mathcal{I}} \mathcal{A}^i$, where $\mathcal{A}^i$ is a finite set representing the action space of agent i;
- $\mathcal{M} = \prod_{i\in\mathcal{I}} \mathcal{M}^i$, where $\mathcal{M}^i$ is a finite set representing the domain of private information for agent i;
- $\mathcal{O}$ is a finite set representing the domain of common observation (A common observation at time t is allowed to be already observed in private by some agents before time t);
- $b_0 \in \Delta(\mathcal{S}\times\mathcal{M})$ is the initial joint distribution on the state and private information of all agents;
- $\mathbb{P} : \mathcal{S}\times\mathcal{M}\times\mathcal{A} \mapsto \Delta(\mathcal{S}\times\mathcal{M}\times\mathcal{O})$ is the state-information joint transition kernel;
- $r : \mathcal{S}\times\mathcal{A}\mapsto\mathbb{R}$ is the instantaneous reward;
- $\beta \in (0,1)$ is the discount factor.

Player i's information space at time t is $\mathcal{H}_t^i = (\mathcal{O})^t \times \mathcal{M}^i$ which represents the common observations from time 1 to t along with the private information at time t. Actions may or may not be commonly observed by all agents. If any a_t^i is commonly observed by all agents, then it is part of the common observation o_{t+1}. The goal in this problem is to choose a joint strategy $\pi = (\pi^i)_{i\in\mathcal{I}}, \pi^i = (\pi_t^i)_{t=0}^\infty, \pi_t^i : \mathcal{H}_t^i \mapsto \mathcal{A}^i$ to maximize the total discounted-reward $J(\pi)$, i.e.

$$\max_\pi \mathbb{E}^\pi\left[\sum_{t=0}^\infty \beta^t r(s_t, a_t)\right].$$

As shown in [1], this model can be used to model many decentralized decision and control problems with partial history sharing i.e. problems where certain subsets of agents' action and observation history are commonly known.

Remark 1. In [11], the authors introduced the concept of *sufficient private information*, which is a compression of private information that is sufficient for decision-making purpose. One can replace the space of private information $\mathcal{M}$ in this model by the space of sufficient private information and the proposed algorithm of this work will still apply.

2) Transformation into Coordinator's POMDP: Following the CI approach introduced in [1], we transform the problem $\mathcal{E}$ into an equivalent POMDP problem, called the coordinator's POMDP, which can be characterized by a tuple $\bar{\mathcal{E}} = (\bar{\mathcal{S}}, \bar{\mathcal{A}}, \mathcal{O}, b_0, \bar{\mathbb{P}}, \bar{r}, \beta)$, where $\mathcal{O}, b_0, \beta$ are the same as in the original model; $\bar{\mathcal{S}} = \mathcal{S}\times\mathcal{M}$ is the (augmented) state space; $\bar{\mathcal{A}} = \prod_{i\in\mathcal{I}} \bar{\mathcal{A}}^i$ is the (new) action space, where $\bar{\mathcal{A}}^i$ is the set of mappings (called prescriptions) from $\mathcal{M}^i$ to $\mathcal{A}^i$; $\bar{\mathbb{P}} : \bar{\mathcal{S}}\times\bar{\mathcal{A}}\mapsto\Delta(\bar{\mathcal{S}}\times\mathcal{O})$ is the combined state transition and observation kernel; $\bar{r} : \bar{\mathcal{S}}\times\bar{\mathcal{A}}\mapsto\mathbb{R}$ is the instantaneous reward function. $\bar{\mathbb{P}}$ and $\bar{r}$ are respectively defined by

$$\bar{\mathbb{P}}(\bar{s}', o|\bar{s}, \gamma) = \mathbb{P}(\bar{s}', o|\bar{s}, a) \quad \forall \bar{s}, \bar{s}' \in \bar{\mathcal{S}}, o\in\mathcal{O}, \gamma\in\bar{\mathcal{A}}$$
$$\bar{r}(\bar{s}, \gamma) = r(s, a) \quad \forall \bar{s}\in\bar{\mathcal{S}}, \gamma\in\bar{\mathcal{A}}$$

where $\bar{s} = (s, m), a = (a^i)_{i\in\mathcal{I}}$ and $a^i = \gamma^i(m^i)$.

In this paper, we focus on the coordinator's POMDP and its variations. Without lost of generality, we remove the overline of $\bar{\mathcal{S}}$ and assume that the private information is a fixed function of the state. For $s\in\mathcal{S}$, we use $m_s = (m_s^i)_{i\in\mathcal{I}} \in \mathcal{M}$ to denote its corresponding private information. We also drop the overline of $\bar{\mathbb{P}}$ and $\bar{r}$ to simplify expressions. Note that we still use $\bar{\mathcal{A}}$ to represent the space of prescriptions to distinguish it from the space of actions of individual agents.

Remark 2. The coordinator's POMDP can have a prohibitively large action space even for seemingly small problems like the DecTiger problem with $N \le 3$ doors and 1-step delayed information sharing (described in Appendix A.1). For example, with $N = 2$, we have $|\bar{\mathcal{A}}| = \prod_{i=1}^2 |\mathcal{A}^i|^{|\mathcal{M}^i|} = 3^{14} \approx 4.78\times10^6$; with $N = 3$, we have $|\bar{\mathcal{A}}| = \prod_{i=1}^2 |\mathcal{A}^i|^{|\mathcal{M}^i|} = 4^{26} \approx 4.50\times10^{15}$. For POMDPs with such extraordinarily large action space, most computers wouldn't even have enough memory to initialize off-the-shelf POMDP solvers, let alone running them.

B. Point-based Algorithms for POMDPs

In this section, we provide an overview of point-based algorithms for POMDPs and their common ingredient: the point-based backup operation. We then discuss the heuristic search value iteration (HSVI) algorithm [6], which will be the basis of our proposed algorithm. To describe the algorithms in this section, we consider a standard single-agent infinite-horizon discounted reward POMDP defined through a tuple $\mathcal{E} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, b_0, \mathbb{P}, r, \beta)$.

1) Point-based Algorithms and the Backup Operation: For a POMDP, the value function is defined as a function of the belief state and the Bellman operator is defined as follows: For any function $V : \Delta(\mathcal{S}) \mapsto \mathbb{R}$,

$$TV(b) := \max_{a\in\mathcal{A}} \left[r(b,a) + \beta \sum_{o\in\mathcal{O}} \mathbb{P}(o|b,a) V(\tau(b,a,o)) \right]$$

where $b \in \Delta(\mathcal{S})$ and τ is the belief update function. Since the value function V is known to be piecewise-linear and convex, it can be written as $V(b) = \max_{\alpha\in\mathcal{V}} \alpha^T b$ where $\mathcal{V}$ is a finite collection of $|\mathcal{S}|-$dimensional vectors. Then we have

$$\begin{aligned}
&TV(b)\\
&= \max_{a\in\mathcal{A}} r(b,a) + \sum_{o\in\mathcal{O}} \mathbb{P}(o|b,a) \max_{\alpha\in\mathcal{V}} [\tau(b,a,o)]^T \alpha\\
&= \max_{a\in\mathcal{A}} r(b,a) + \sum_{o\in\mathcal{O}} \max_{\alpha\in\mathcal{V}} [\mathbb{P}(o|b,a)\tau(b,a,o)]^T \alpha\\
&= \max_{a\in\mathcal{A}} r(b,a) + \sum_{o\in\mathcal{O}} \max_{\alpha\in\mathcal{V}} \sum_{s'\in\mathcal{S}} \mathbb{P}(s',o|b,a)\alpha(s')\\
&= \max_{a\in\mathcal{A}} \max_{\mu\in\mathcal{V}^{\mathcal{O}}} r(b,a) + \sum_{o\in\mathcal{O}} \sum_{s'\in\mathcal{S}} \mathbb{P}(s',o|b,a)\mu_o(s')\\
&= \max_{a\in\mathcal{A}} \max_{\mu\in\mathcal{V}^{\mathcal{O}}} (\alpha^{a,\mu})^T b
\end{aligned}$$

where $\mathcal{V}^{\mathcal{O}}$ is the set of all mappings from $\mathcal{O}$ to $\mathcal{V}$ and

$$\alpha^{a,\mu}(s) := r(s,a) + \beta \sum_{o\in\mathcal{O}} \sum_{s'\in\mathcal{S}} \mathbb{P}(s',o|s,a)\mu_o(s').$$

Therefore, the value function can be achieved by updating the collection of α-vectors [2], [3], [4] through

$$\mathcal{V}^{(k+1)} := \{\alpha^{a,\mu} : a \in \mathcal{A}, \mu \in (\mathcal{V}^{(k)})^{\mathcal{O}}\}. \tag{1}$$

However, computing (1) is inefficient: We can see that $|\mathcal{V}^{(k+1)}| = |\mathcal{A}| \cdot |\mathcal{V}^{(k)}|^{|\mathcal{O}|}$, i.e. the growth of number of α-vectors is doubly exponential. Even with procedures to prune out dominated α-vectors [3], the exact value iteration is still ill-equipped to handle large state and action spaces [4].

While computing all the α-vectors in a Bellman update is expensive, computing an α-vector that supports the updated function at a particular belief point $b \in \Delta(\mathcal{S})$ (i.e. $TV(b) = \alpha^T b$ and $TV(\tilde{b}) \geq \alpha^T \tilde{b}$ for all $\tilde{b} \in \Delta(\mathcal{S})$) is not: one can compute this vector α^b through

$$\alpha^{b,a,o} := \arg\max_{\alpha\in\mathcal{V}} \sum_{s\in\mathcal{S}} b(s) \sum_{s'\in\mathcal{S}} \mathbb{P}(s',o|s,a)\alpha(s') \tag{2}$$
$$\forall a \in \mathcal{A}, o \in \mathcal{O}$$

$$\alpha^{b,a}(s) := r(s,a) + \beta \sum_{o\in\mathcal{O}} \sum_{s'\in\mathcal{S}} \mathbb{P}(s',o|s,a)\alpha^{b,a,o}(s') \tag{3}$$
$$\forall s \in \mathcal{S}, a \in \mathcal{A}$$

$$\alpha^b := \arg\max_{\alpha^{b,a}:a\in\mathcal{A}} \left(\alpha^{b,a}\right)^T b \tag{4}$$

The above procedure is referred to as the (point-based) *backup* procedure at belief point b. Point-based algorithms for POMDPs are based on the idea that by using backup procedures at a carefully selected subset of belief points, one can obtain a set of α-vectors that provide a good approximation of the optimal value function.

Function `GenericPointBased`**:**
- **Input:** A single-agent POMDP model $(\mathcal{S}, \mathcal{A}, \mathcal{O}, b_0, \mathbb{P}, r, \beta)$
- **Output:** A belief based policy π and a lower bound of $V^\pi(b_0)$
- Initialize $\mathcal{V}$;
- **while** *Stopping criterion not satisfied* **do**
  - Choose belief point set $\mathcal{B}$;
  - $\alpha^b =$ `Backup(`$\mathcal{V}, b$`)` for all $b \in \mathcal{B}$;
  - Add $\{\alpha^b\}_{b\in\mathcal{B}}$ to $\mathcal{V}$;
  - Prune dominated vectors in $\mathcal{V}$ once in a while;
- $\pi :=$ `DirectControlPolicy(`$\mathcal{V}$`)`;
- $\underline{v} := \max_{\alpha\in\mathcal{V}} \alpha^T b_0$;
- **return** $\pi, \underline{v}$

Algorithm 1: A generic point-based algorithm for single-agent POMDP

Moreover, if the initial set $\mathcal{V}$ represents a uniformly improvable (as defined in [23]) lower bound of the optimal value function, then $\mathcal{V}$ will remain a uniformly improvable lower bound after inserting new α-vectors from point-based updates. It can also be shown that [8] in this case, the value of the direct control policy[1] obtained from $\mathcal{V}$ is lower bounded by V, the piece-wise linear convex function with α-vectors $\mathcal{V}$. Therefore we have a theoretical guarantee of the resulting policy. A generic framework for many point-based algorithms (e.g. [25], [6], [7], [26]) is shown in Algorithm 1. These algorithms all share the same backup procedure. They differ mostly in how they determine the set of belief points to perform backups on.

2) Heuristic Search Value Iteration: The HSVI algorithm in [6] is summarized in Algorithm 2. In addition to a set of α-vectors that represent a lower bound for the optimal value function V^*, the algorithm maintains an upper bound of V^* as well. At each time, the algorithm performs a depth-first search to select a few belief points. The algorithm updates both the upper bound and lower bound at those beliefs. In those updates, the algorithm computes exact Bellman updates of the upper and lower bounds at those beliefs,

[1]The direct control policy [24], [8] can be described as follows: At belief b, find an α-vector $\alpha^* \in \mathcal{V}$ that maximizes $\alpha^T b$, and take the action a^* associated with α^*, i.e. the maximizing action in (4) when α^* was initially computed.

and incorporate the new values back into the bounds. As a result, the lower bound increases and upper bound decreases over time. HSVI is an anytime algorithm that always tries to shrink the gap between the upper and the lower bound. Once the algorithm terminates once a certain stopping criteria is met, it returns the direct control policy associated with the α-vectors used for the lower bound function.

The HSVI algorithm distinguishes itself from previous algorithms through its use of an upper-bound based heuristic search procedure, as it is defined in the function `ChooseNext`. To determine the next belief to explore, the algorithm first optimistically chooses an action that maximizes the action-value function (or Q-function) associated with the upper bound. Then, the algorithm picks an observation that maximizes the *excess gap*. This procedure allows the algorithm to focus on the beliefs where the current gap between the upper bound and lower bound is unsatisfactory.

The lower bound function L in HSVI is represented as a set of α-vectors. Its update function $L.\texttt{Update}(b)$ adds a new α-vector to the set by performing the backup operation at b. The algorithm also periodically prunes certain dominated α-vectors in the set.

Unlike the lower bound, the upper bound function in HSVI is represented through the lower convex hull of a set of isolated points $(b, \bar{v}_b) \in \Delta(\mathcal{S}) \times \mathbb{R}$, i.e., let $\mathbf{B}$ be a matrix whose n column vectors represent n beliefs and $\bar{v} \in \mathbb{R}^n$ be a vector representing the upper bound values of those beliefs, then for any $b \in \Delta(\mathcal{S})$,

$$U(b) = \min\{\bar{v}^T \eta : \eta \in \mathbb{R}^n_+, \mathbf{B}\eta = b.\} \tag{5}$$

This representation is based on the fact that V^* is convex. Let T be the Bellman operator as defined in Section II-B.1. The $U.\texttt{Update}(b)$ procedure in Algorithm 2 computes

$$\bar{v}_b := TU(b) = \max_{a \in \mathcal{A}} r(b,a) + \beta \sum_{o \in \mathcal{O}} \mathbb{P}(o|b,a) U(\tau(b,a,o)) \tag{6}$$

and then adds $(b, \bar{v}_b)$ to the point set used for convex hull. The algorithm also removes redundant points in the set periodically. The procedure returns $a^* \in \mathcal{A}$, an optimizer of (6), for potential future use in the search heuristics.

Function `HSVI`:
- **Input:** A single-agent POMDP model $(\mathcal{S}, \mathcal{A}, \mathcal{O}, b_0, \mathbb{P}, r, \beta)$
- **Output:** A belief based policy π; an upper bound for $V^*(b_0)$ and lower bound for $V^\pi(b_0)$.

`//` $\zeta \in (0,1)$ `is a hyperparameter.`
Initialize U and L;
while *Stopping criterion not satisfied* **do**
  $\epsilon = \zeta[U(b_0) - L(b_0)]$;
  `Explore`(U, L, b_0, ϵ);
$\pi =$ `DirectControlPolicy`(L);
return $\pi, U(b_0), L(b_0)$

Function `Explore`(U, L, b, ϵ):
`//` `Modifies` U, L
$a^* = U.\texttt{Update}(b)$;
$L.\texttt{Update}(b)$;
if $U(b) - L(b) \leq \epsilon$ **then return**;
$(b', \epsilon') =$ `ChooseNext`(U, L, b, a^*, ϵ);
`Explore`(U, L, b', ϵ');
$U.\texttt{Update}(b)$;
$L.\texttt{Update}(b)$;

Function `ChooseNext`(U, L, b, a^*, ϵ):
$o^* = \arg\max_{o \in \mathcal{O}} \mathbb{P}(o|b,a^*)[U(\tau(b,a^*,o)) - L(\tau(b,a^*,o)) - \frac{\epsilon}{\beta})]$;
return $\tau(b, a^*, o^*), \frac{\epsilon}{\beta}$;

Algorithm 2: Heuristic Search Value Iteration

III. Coordinator's HSVI Algorithm

As noted earlier, compared to a regular single-agent POMDP, a coordinator's POMDP will have an exponentially large number of actions. This makes both the lower bound and upper bound update of HSVI infeasible. More specifically, in the backup operations (2)-(4), we need to first solve $|\bar{\mathcal{A}}| \times |\mathcal{O}|$ discrete optimization problems in (2) and then solve an optimization problem over $\bar{\mathcal{A}}$ in (4). Also, to perform a Bellman update for the upper bound at belief point b, the HSVI algorithm needs to compute $U(\tau(b, \gamma, o))$ for all pairs of $(\gamma, o) \in \bar{\mathcal{A}} \times \mathcal{O}$. This is infeasible also due to the large number of actions in a coordinator's POMDP.

In this section, we introduce the *Coordinator's Heuristic Search Value Iteration* (CHSVI) algorithm, which combines the CI approach with the HSVI algorithm. Instead of applying HSVI directly on the coordinator's POMDP, we apply HSVI on a multi-step extended form of coordinator's POMDP. Through the use of the extended form and the structure of prescription space, we are able to simplify both the upper bound and lower bound update operations, creating a more practical algorithm.

We first describe the extended form of coordinator's POMDP. For the ease of illustration, consider $\mathcal{I} = \{1, 2\}$ though the idea can naturally extend to more than two players. We derive an extended POMDP $\hat{\mathcal{E}}$ from the coordinator's POMDP $\bar{\mathcal{E}} = \{\mathcal{S}, \bar{\mathcal{A}}, \mathcal{O}, b_0, \mathbb{P}, r, \beta\}$ by extending each time t into three stages: $(t,0), (t,1), (t,2)$. The state space is $\mathcal{S}^0 := \mathcal{S}$ at $(t,0)$, $\mathcal{S}^1 := \mathcal{S} \times \mathcal{A}^1$ at $(t,1)$, and $\mathcal{S}^2 := \mathcal{S} \times \mathcal{A}$ at $(t,2)$. The common observation space is $\{\emptyset\}$ at both $(t,0)$ and $(t,1)$, and $\mathcal{O}$ at $(t,2)$. Only stage $(t,2)$ features an instantaneous reward, which is $r(s,a)$ for $(s,a) \in \mathcal{S}^2$. The system evolves as follows:

1) At $(t,0)$, the coordinator chooses prescription $\gamma^1 \in \bar{\mathcal{A}}^1$ for agent 1 only. The state deterministically transits from $s \in \mathcal{S}$ to $(s, \gamma^1(m_s^1)) \in \mathcal{S} \times \mathcal{A}^1$.
2) At $(t,1)$, the coordinator chooses prescription $\gamma^2 \in \bar{\mathcal{A}}^2$ for agent 2 only. The state deterministically transits from $(s, a^1) \in \mathcal{S} \times \mathcal{A}^1$ to $(s, a^1, \gamma^2(m_s^2)) \in \mathcal{S} \times \mathcal{A}^1 \times \mathcal{A}^2$.
3) At $(t,2)$, the coordinator has no action to take. The state-information joint transition kernel is simply the same $\mathbb{P}(s', o|s, a)$ as defined in the initial model $\mathcal{E}$.

The total reward for the new POMDP is given by

$\sum_{t=0}^{\infty} \beta^t r(s_{(t,2)}, a_{(t,2)})$. With some abuse of notation, for $\gamma^i \in \bar{\mathcal{A}}^i$, we define $\gamma^i(a^i|m^i) := \mathbf{1}_{\{\gamma^i(m^i)=a^i\}}$. The belief update functions for the three stages are given by

$$[\tau^0(b,\gamma^1)](s,a^1) = b(s)\gamma^1(a^1|m^1_s)$$
$$[\tau^1(b,\gamma^2)](s,a^1,a^2) = b(s,a^1)\gamma^2(a^2|m^2_s)$$
$$[\tau^2(b,o)](s') = \frac{\sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} \mathbb{P}(s',o|s,a)b(s,a)}{\sum_{\tilde{s}\in\mathcal{S}}\sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} \mathbb{P}(\tilde{s},o|s,a)b(s,a)}$$

The extended coordinator's POMDP $\hat{\mathcal{E}}$ differs from the coordinator's POMDP $\bar{\mathcal{E}}$ only in the ordering of events within a time instant t but not in the total reward at time t and the dynamics from t to $t+1$. Therefore, $\hat{\mathcal{E}}$ is equivalent to $\bar{\mathcal{E}}$: Any strategy in $\hat{\mathcal{E}}$ has a counterpart in $\bar{\mathcal{E}}$ with the same total reward and vice versa.

In CHSVI algorithm (Algorithm 3), we apply the framework of HSVI to the extended coordinator's POMDP while utilizing the special structure of the prescription space to implement the update procedures more efficiently. In the following sections, we describe the upper bound U and lower bound L used in the CHSVI algorithm in detail.

Remark 3. In this section we have taken an agnostic approach to transform any coordinator's POMDP into an extended coordinator's POMDP, where the intermediate state spaces are $\mathcal{S}\times\mathcal{A}^1$ and $\mathcal{S}\times\mathcal{A}^1\times\mathcal{A}^2$, and only one agent (including nature) takes action at one stage. In fact, the idea of the CHSVI algorithm can be applied to any transformation of a coordinator's POMDP such that only one agent takes action at one stage.

A. Lower Bound Update

In CHSVI, we maintain three lower bound functions (sets of α-vectors) corresponding to the three stages of t. Let L^ℓ denote the lower bound function and $\mathcal{V}^\ell$ denote the corresponding set of α-vectors for stage ℓ (of some time t). Let T^ℓ be the Bellman operator for stage ℓ. We now describe the update steps in detail.

Stage $\ell = 0, 1$: In this stage, the coordinator picks a prescription for agent $i = \ell+1$. For $b \in \Delta(\mathcal{S}^\ell)$, the Bellman update of the lower bound at b is given by

$$[T^\ell L^{\ell+1}](b) = \max_{\gamma^i\in\bar{\mathcal{A}}^i} L^{\ell+1}(\tau^\ell(b,\gamma^i))$$
$$= \max_{\gamma^i\in\bar{\mathcal{A}}^i}\max_{\alpha\in\mathcal{V}^i} \sum_{(s^\ell,a^i)\in\mathcal{S}^\ell\times\mathcal{A}^i} b(s^\ell)\gamma^i(a^i|m^i_{s^\ell})\alpha(s^\ell,a^i) \quad (7)$$

The optimization problem (7) can be solved via the following steps:

$$\gamma^{i,\alpha}(m^i) := \arg\max_{a^i\in\mathcal{A}^i} \sum_{s^\ell\in\mathcal{S}^\ell : m^i_{s^\ell}=m^i} b(s^\ell)\alpha(s^\ell,a^i) \quad (8)$$
$$\forall m^i \in \mathcal{M}^i, \alpha\in\mathcal{V}^i$$
$$J(\alpha) := \sum_{s^\ell\in\mathcal{S}^\ell} b(s^\ell)\alpha(s^\ell,\gamma^{i,\alpha}(m^i_{s^\ell})) \quad \forall\alpha\in\mathcal{V}^i \quad (9)$$
$$\alpha^* := \arg\max_{\alpha\in\mathcal{V}^i} J(\alpha) \quad (10)$$
$$\gamma^{i,*} := \gamma^{i,\alpha^*}, \quad (11)$$

Function `CHSVI`:
- **Input:** A coordinator's POMDP model $(\mathcal{S},\bar{\mathcal{A}},\mathcal{O},b_0,\mathbb{P},r,\beta)$
- **Output:** A belief based coordination policy π; an upper bound for $V^*(b_0)$ and lower bound for $V^\pi(b_0)$.
- `//` $\zeta\in(0,1)$ `is a hyperparameter.`
- Initialize U and L;
- **while** *Stopping criterion not satisfied* **do**
  - $\epsilon = \zeta[U(b_0) - L(b_0)]$;
  - `Explore`(U,L,b_0,ϵ);
- $\pi =$ `DirectControlPolicy`(L);
- **return** $\pi, U(b_0), L(b_0)$

Function `Explore`(U,L,b,ϵ):
- $\gamma^* = U$.`Update`(b);
- L.`Update`(b);
- **if** $U(b) - L(b) \le \epsilon$ **then return**;
- $(b',\epsilon') =$ `ChooseNext`$(U,L,b,\gamma^*,\epsilon)$;
- `Explore`(U,L,b',ϵ');
- U.`Update`(b);
- L.`Update`(b);

Function `ChooseNext`$(U,L,b,\gamma^*,\epsilon)$:
- Set $\ell\in\{0,1,2\}$ to be such that $b\in\Delta(\mathcal{S}^\ell)$;
- **if** $\ell < 2$ **then**
  - **return** $\tau^\ell(b,\gamma^*),\epsilon$;
- **else**
  - $o^* = \arg\max_{o\in\mathcal{O}} \mathbb{P}(o|b)[U(\tau^2(b,o)) - L(\tau^2(b,o)) - \epsilon\beta^{-1}]$;
  - **return** $\tau^2(b,o^*),\epsilon\beta^{-1}$;

Algorithm 3: Coordinator's HSVI Algorithm

where we have used the fact that for each fixed $\alpha\in\mathcal{V}^i$, the optimization problem over $\gamma^i\in\bar{\mathcal{A}}^i$ can be separated into $|\mathcal{M}^i|$-optimization problems over $\mathcal{A}^i$. Then, we add the following new alpha vector $\alpha^b\in\mathbb{R}^{\mathcal{S}^\ell}$ to $\mathcal{V}^\ell$:

$$\alpha^b(s^\ell) = \sum_{a^i\in\mathcal{A}^i} \gamma^{i,*}(a^i|m^i_{s^\ell})\alpha^*(s^\ell,a^i) \quad \forall s^\ell\in\mathcal{S}^\ell. \quad (12)$$

Remark 4. If we apply the point-based backup procedure (2) – (4) directly at this stage, then the operation count would be $\Theta(|\mathcal{S}||\mathcal{V}||\bar{\mathcal{A}}^i|)$, which grows exponentially in $|\mathcal{M}^i|$ since $|\bar{\mathcal{A}}^i| = |\mathcal{A}^i|^{|\mathcal{M}^i|}$. In contrast, the operation count of the procedure listed in (8) – (12) is $\Theta(|\mathcal{S}||\mathcal{V}^i||\mathcal{A}^i||\mathcal{M}^i|)$, which is polynomial in all parameters involved.

Remark 5. We have used the separability of the optimization problem over the prescription space $\bar{\mathcal{A}}^i$ for a fixed α-vector to simplify the optimization problem (7). This can only be achieved if there's only one agent at one stage, which is part of the reason why we use the extended form of coordinator's POMDP.

Stage 2: For $b\in\Delta(\mathcal{S}^2)$ the Bellman update of the lower bound at b is given by

$$[T^2L^0](b) = r(b) + \beta\sum_{o\in\mathcal{O}} \mathbb{P}(o|b)\max_{\alpha\in\mathcal{V}^0}[\tau^2(b,o)]^T\alpha$$

$$= r(b) + \beta \sum_{o\in\mathcal{O}} \max_{\alpha\in\mathcal{V}^0} \sum_{s^2\in\mathcal{S}^2} b(s^2) \sum_{s'\in\mathcal{S}} \mathbb{P}(s', o|s^2)\alpha(s')$$

where $r(b) := \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} b(s,a)r(s,a)$.

Since we have no action in this stage, we can compute the new alpha vector $\alpha^b \in \mathbb{R}^{\mathcal{S}^2}$ in the same way as in (2)-(4) (except that there's no action), i.e. we compute

$$\alpha^{b,o} := \arg\max_{\alpha\in\mathcal{V}^0} \sum_{s^2\in\mathcal{S}^2} b(s^2) \sum_{s'\in\mathcal{S}} \mathbb{P}(s', o|s^2)\alpha(s') \quad \forall o \in \mathcal{O} \tag{13}$$

$$\alpha^b(s^2) := r(s^2) + \beta \sum_{o\in\mathcal{O}} \sum_{s'\in\mathcal{S}} \mathbb{P}(s', o|s^2)\alpha^{b,o}(s') \quad \forall s^2 \in \mathcal{S}^2 \tag{14}$$

The lower bound update algorithm is given in Algo. 4.

Function L.`Update`(b)**:**
  `//` L `stores three sets of` α`-vectors:` $\mathcal{V}^\ell, \ell = 0, 1, 2$
  Set $\ell \in \{0, 1, 2\}$ to be such that $b \in \Delta(\mathcal{S}^\ell)$;
  if $\ell < 2$ **then**
    Compute α^b with (8) – (12);
  else
    Compute α^b with (13)(14);
  Add α^b to $\mathcal{V}^\ell$;
  Prune dominated vectors in $\mathcal{V}^\ell$ once in a while;

Algorithm 4: Lower Bound Update

B. A New Upper Bound Representation

In this section, we propose a new upper bound representation for POMDPs, called α-constraints based upper bound. Our motivation for finding a new upper bound representation comes from a weakness of the convex hull-based upper bound in extended coordinator's POMDPs. We believe that our new upper bound can alleviate the weakness. In any case, the new upper bound is never worse (i.e. larger) than the original upper bound.

In HSVI, the upper bound function plays a very important role in the search heuristics. It is represented with a collection of isolated points $(b, v^b) \in \Delta(\mathcal{S}) \times \mathbb{R}$. The upper bounds at other belief points are obtained via convex hull based interpolation. While convex hull based upper bounds and their approximations have achieved empirical success [8], [7] on many single-agent POMDP problems, such bounds can be insufficient for the extended coordinator's POMDP due to the following reason: For two probability distributions b and $\tilde{b}$ defined on the same space, we say that b is *inexpressible* with $\tilde{b}$ if in any convex combination that represents b, $\tilde{b}$ is not involved, i.e. its coefficient is 0. The inclusion of actions into the state space creates a exponentially large number of mutually inexpressible beliefs. As a result, the upper bound at those beliefs do not improve (i.e. become smaller) from their initial values unless it is explored by the algorithm. Due to the optimism-based exploration heuristics of these algorithms [8], [7], it is likely that the algorithm would end up exhaustively sampling those beliefs, resulting in a behavior similar to brute-force search.

To illustrate this point more clearly, consider a belief $b_0 \in \Delta(\mathcal{S})$ with full support. Then, consider two prescriptions $\gamma^1, \tilde{\gamma}^i \in \bar{\mathcal{A}}^1$. Suppose that $\gamma^1(m^1) = a^1 \neq \tilde{\gamma}^1(m^1) = \tilde{a}^1$ for some $m^1 \in \mathcal{M}^1$. Let $b_1 = \tau^0(b_0, \gamma^1)$ and $\tilde{b}_1 = \tau^0(b_1, \tilde{\gamma}^1)$. Let $s \in \mathcal{S}$ be some state such that $m^1_s = m^1$. We have $b_1(s, a^1) > 0, \tilde{b}_1(s, a^1) = 0$. This means that $\tilde{b}_1$ is inexpressible with b_1 and vice versa. As a result, updating the upper bound at b_1 yields no effect on the upper bound at $\tilde{b}_1$, even if b_1 is very close to $\tilde{b}_1$ (which is the case when γ^1 differs from $\tilde{\gamma}^1$ only at one $m^1 \in \mathcal{M}^1$).

To resolve this problem, we would like an upper bound representation that is not only tight, but also facilitates more *cross learning*, i.e. knowing the upper bound of a belief b helps us to reduce the upper bound another belief $\tilde{b}$, even if $\tilde{b}$ is inexpressible with b. Furthermore, if b and $\tilde{b}$ is close, then we would like their upper bound to be close as well.

To design a new upper bound representation, we start from the dual form of the convex hull representation (5) given by

$$U(b) = \max\{b^T y : y \in \mathbb{R}^{\mathcal{S}}, \mathbf{B}^T y \leq \bar{v}\}. \tag{15}$$

A direct interpretation of (15) is as follows: The optimal value function has an α-vector representation $V^*(b) = \max_{\alpha\in\mathcal{V}^*} \alpha^T b$, where $\mathcal{V}^*$ represents the limit of $\mathcal{V}^{(k)}$ defined in (1). The maximization problem (15) provides an upper bound for $V^*(b)$ since all α-vectors in $\mathcal{V}^*$ satisfy the constraints $\mathbf{B}^T\alpha \leq \bar{v}$ (due to the fact that $\bar{v}$ is an upper bound for V^* at beliefs in $\mathbf{B}$). In other words, (15) can be seen as an *α-constraint* based upper bound, where the upper bound function is constructed from a group of linear constraints the set $\mathcal{V}^*$ should satisfy.

Building upon this idea, we make use of other types of linear constraints other than value function upper bounds, e.g., $\alpha(s) \geq v_{\min}$ where $v_{\min} := \min_{s\in\mathcal{S},a\in\mathcal{A}} r(s,a)/(1-\beta)$. Adding this to (15), we have

$$U(b) = \max\{b^T y : y \in \mathbb{R}^{\mathcal{S}}, \mathbf{B}^T y \leq \bar{v}, y \geq v_{\min}\mathbf{1}\}. \tag{16}$$

to be an upper bound no worse than the original one used in HSVI. In contrast to (15), the upper bound (16) can facilitate more cross learning: By bounding the L_∞ diameter of the constraint set, it can be shown that (16) is L_1-Lipschitz continuous with constant $v_{\max} - v_{\min}$ (where $v_{\max} := \max_{s\in\mathcal{S},a\in\mathcal{A}} r(s,a)/(1-\beta)$), hence ensuring that if b is close to $\tilde{b}$, then their upper bounds are relatively close as well, no matter whether $\tilde{b}$ is expressible with b or not. See Appendix B for more types of α-constraints.

In summary, any valid linear inequalities that are satisfied by all vectors in $\mathcal{V}^*$ can be added to the set of α-constraints. This provides us a lot of flexibility compared to the convex hull-based bounds. We next describe the update of α-constraint based upper bound in extended coordinator's POMDPs. The update method described in the next section is independent of the specific choice of α-constraints.

C. Upper Bound Update

We maintain three upper bound functions corresponding to the three stages. Each upper bound function is represented through a set of α-constraints as described in the previous section. Let $\mathbf{M}^\ell y \leq w^\ell$ represent the α-constraints (on $y \in \mathcal{S}^\ell$) associated with stage ℓ. Let T^ℓ be the Bellman operator for stage ℓ. We now describe the update steps in detail:

Stage $\ell = 0, 1$: In this stage, the coordinator picks a prescription for agent $i = \ell + 1$. For $b \in \Delta(\mathcal{S}^\ell)$, the Bellman updated upper bound at b is given by

$$\begin{aligned}\bar{v}^b &:= [T^\ell U^{\ell+1}](b) = \max_{\gamma^i \in \bar{\mathcal{A}}^i} U^{\ell+1}(\tau^\ell(b, \gamma^i)) \\ &= \max_{\gamma^i \in \bar{\mathcal{A}}^i} \max_{\substack{y \in \mathbb{R}^{\mathcal{S}^i} \\ \mathbf{M}^i y \leq w^i}} \sum_{(s^\ell, a^i) \in \mathcal{S}^\ell \times \mathcal{A}^i} b(s^\ell)\gamma^i(a^i | m^i_{s^\ell}) y(s^\ell, a^i) \\ &=: \max_{\gamma^i \in \bar{\mathcal{A}}^i} \max_{\substack{y \in \mathbb{R}^{\mathcal{S}^i} \\ \mathbf{M}^i y \leq w^i}} J^i(b, \gamma^i, y) \qquad (17)\end{aligned}$$

Now, notice that if we treat each γ^i as a 0-1 indicator vector in $[0,1]^{\mathcal{A}^i \times \mathcal{M}^i}$, then $J^i(b, \gamma^i, y)$ is *bilinear* in γ^i and y: It is linear in γ^i for each fixed y and linear in y for each fixed γ^i. Therefore, (17) is a *bilinear programming* (BP) problem, which has been studied extensively in the optimization literature [27], [28], [29] as well as decentralized control literature. [30], [31]. Notably, in [30], the author provided a method to convert bilinear programs to MILPs[2]. Gurobi Optimization™ has also developed specialized solvers for bilinear programs [32].

Remark 6. Even though bilinear programming problems are NP-hard [33] in general, modeling the upper bound update through BP still offers several advantages over the original update method in HSVI (i.e. separately solving the inner linear program in (17) for each $\gamma^i \in \bar{\mathcal{A}}^i$) for the following reasons: (i) It allow us to apply systematic methods for BP to avoid brute-force enumeration of prescriptions. (ii) Not all BPs fall into the NP-hard category. (iii) It opens up the door for approximate upper bound methods (e.g. certain relaxation of the MILP reformulation [30] of BP).

Stage 2: For $b \in \Delta(\mathcal{S}^2)$, the Bellman update of the upper bound at belief b is given by

$$\begin{aligned}\bar{v}^b &= [T^2 U^0](b) \\ &= r(b) + \beta \sum_{o \in \mathcal{O}} \mathbb{P}(o|b) \max_{\substack{y \in \mathbb{R}^{\mathcal{S}} \\ \mathbf{M}^0 y \leq w^0}} [\tau^2(b, o)]^T y \qquad (18)\end{aligned}$$

which can be computed by solving $|\mathcal{O}|$ linear programs.

The upper bound update algorithm is given in Algo. 5.

Function U.`Update`(b)**:**
  `// U stores three sets of α-constraints:` $\mathcal{C}^\ell, \ell = 0, 1, 2$
  Set $\ell \in \{0, 1, 2\}$ to be such that $b \in \Delta(\mathcal{S}^\ell)$;
  if $\ell < 2$ **then**
    Compute $\bar{v}^b, \gamma^{i,*}$, the optimal value and optimizer of the bilinear program (17);
  else
    Compute $\bar{v}^b$ with (18);
    Set $\gamma^{i,*}$ to represent the null prescription;
  Add $b^T y \leq \bar{v}^b$ to $\mathcal{C}^\ell$;
  Prune redundant α-constraints once in a while;
  return $\gamma^{i,*}$;

Algorithm 5: Upper Bound Update

IV. Experimental Results

We implemented the CHSVI algorithm in Python (`https://github.com/dwtang/chsvi`). All BP and LP involved in the algorithm are solved with Gurobi Optimization Studio™. The hyperparameter ζ is set to 0.85. We terminate the algorithm when the gap between the upper and lower bounds is less than 0.01 or if the run time has exceeded 24 hours. The lower bounds are initialized through the fixed action bound in the same way as in [8]. The α-constraints used in upper bounds are initialized to be the marginal belief based constraints (see Appendix B.2) obtained from a relaxed POMDP problem where all private information are assumed to be common. We adapt the same pruning strategy as HSVI: For lower bounds, we only prune α-vectors that are pointwise dominated by another vector. For upper bounds, we remove redundant α-constraints through solving linear programs.

We run the algorithm on `DecTiger` and `MultiCast` instances defined in Appendix A.1 and A.2 respectively. The experiments are conducted on a computer with Intel Xeon® E3-1231 v3 CPU (4 cores, 3.4Ghz) and 16GB RAM. The results are shown in Table III and Figure 1, with additional plots in Appendix C. In the all instances except `DecTiger`(3,1,0.99), the algorithm is able to close the gap between the upper and lower bound and find a near optimal strategy. To the best of our knowledge, except when analytical solutions are available, these are the first provably optimal solutions for infinite-horizon multi-agent control problems.

	$L(b_0)$	$U(b_0)$	Time (s)
`DecTiger` (2,1,0.9)	32.7704	32.7792	56
`DecTiger` (2,1,0.99)	388.4035	388.4134	2081
`DecTiger` (3,1,0.9)	6.7139	6.7236	63781
`DecTiger` (3,1,0.99)	76.2553	222.2532	86483
`MultiCast` (8,8,0.1,0.2,0.9)	-4.8550	-4.8450	346
`MultiCast` (16,16,0.2,0.4,0.9)	-10.0653	-10.0400	86423

TABLE I
Experimental Results.

Even though the algorithm took hours to close the gap between upper and lower bound to 0.01 for `MultiCast` instances, we would like to note that in hindsight, the lower bound arrives at a near optimal value very quickly as shown in Figure 2. The remainder of the algorithm mostly reduces the upper bound to provide an optimality guarantee. Recall that CHSVI is an anytime algorithm. This means that even

[2]The method only applies to BP where the constraints on two groups of variables are separate. The optimization problem of (17) does lie in this category.

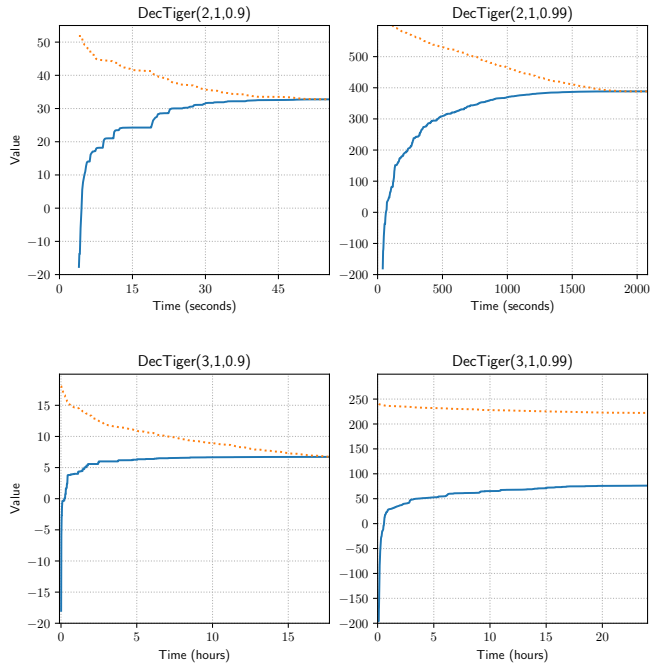

Fig. 1. Experimental Results on `DecTiger` (N, d, β). The dotted line represents the upper bound $U(b_0)$ and the solid line represents the lower bound $L(b_0)$.

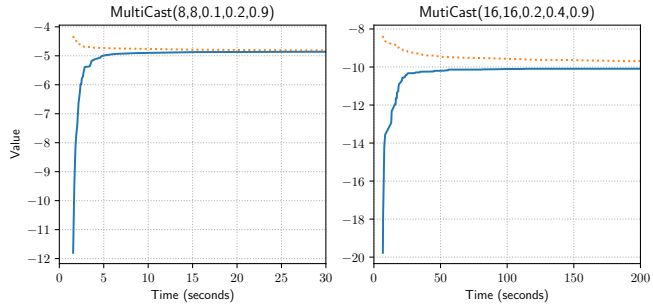

Fig. 2. Experimental Results on `MultiCast` $(C^1, C^2, p^1, p^2, \beta)$. The dotted line represents the upper bound $U(b_0)$ and the solid line represents the lower bound $L(b_0)$. Only the begining of the algorithm is shown. The lower bound quickly arrives at a near optimal value. For the rest of the run time, the upper bound and lower bound both moves very slowly.

if we terminate the algorithm early, we can still obtain a strategy with near optimal performance.

It took the algorithm more than 18 hours to reduce the gap for `DecTiger`(3,1,0.9). However, in hindsight, one can observe that the lower bound arrives at a near optimal value relatively quickly (At the 3 hour mark, the lower bound is already at 5.9919, not too far from the final value of 6.7236). In `DecTiger`(3,1,0.99), the gap is still large at the 24 hours mark. However, we conjecture that the solution reported at 24 hours is already close to optimal, and the gap will continue to decrease with run time.

We would like to note that directly applying off-the-shelf POMDP solvers to any of the instances we used is out of the question on our computer: We have tried to apply the state-of-the-art HSVI2 solver [8] on `MultiCast`(5,5,0.1,0.2,0.9), which has 36 states, 4 observations, and 2^{12} actions/prescriptions. The program took more than 1 hour just to initialize the solver (in this stage, the algorithm does not output any solution if one terminates the algorithm).[3] In comparison, the instances we solved are much bigger: The number of states, observations, and actions/prescriptions for the coordinator's POMDP of our instances is listed in Table II. Given that the complexity of the initialization stage of HSVI2 algorithm (which computes an upper bound with the Fast Informed Bound method [8]) is linear in the number of actions/prescriptions, we project that it would take at least 2.7 days for `MultiCast` (8,8,0.1,0.2,0.9) and 194 days for any other instances just to initialize the HSVI2 solver.

	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{O}\rvert$	$\lvert\bar{\mathcal{A}}\rvert$
`DecTiger` (2,1,β)	74	37	3^{14}
`DecTiger` (3,1,β)	435	145	4^{26}
`MultiCast` (8,8,0.1,0.2,0.9)	64	4	2^{18}
`MultiCast` (16,16,0.2,0.4,0.9)	256	4	2^{34}

TABLE II
DIMENSIONS OF THE COORDINATOR'S POMDP.

V. CONCLUSIONS

Decentralized stochastic control problems can be significantly more difficult than their centralized counterparts due to information asymmetry. Even though some decentralized control problem can be transformed into an equivalent centralized problem with the CI approach, these centralized problems are still difficult to solve due to the extremely large number of actions. In this work, we proposed the Coordinator's Heuristic Search Value Iteration (CHSVI) algorithm, which combines the CI approach and point-based POMDP methods to solve multi-agent stochastic control problems. Our algorithm allows us to solve multi-agent control problems much more efficiently than directly using point-based algorithms for coordinator's POMDP (see Remarks 1 and 2). Further, our approach suggests several immediate future directions for further improving scalability such as: (1) approximate upper bound update methods for CHSVI; (2) combination of the CI approach with other point-based algorithms such as SARSOP[7]; (3) efficient methods to initialize α-constraints for the upper bound representation.

REFERENCES

[1] A. Nayyar, A. Mahajan, and D. Teneketzis, "Decentralized stochastic control with partial history sharing: A common information approach," *IEEE Trans. Automat. Contr.*, vol. 58, no. 7, pp. 1644–1658, 2013.
[2] E. J. Sondik, "The optimal control of partially observable Markov processes over the infinite horizon: Discounted costs," *Oper. Res.*, vol. 26, no. 2, pp. 282–304, 1978.
[3] A. Cassandra, M. L. Littman, and N. L. Zhang, "Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes," in *Proc. 13th Conf. UAI*, 1997, pp. 54–61.

[3] In the CDC submission, we stated that we tried to solve an instance with 25 states, 4 observations, and 2^{10} actions/prescriptions using the HSVI2 solver and it caused the computer to freeze. The instance is different from `MultiCast` with buffer size 4. It has much more non-zero entries in its transition kernel than `MultiCast`.

[4] G. Shani, J. Pineau, and R. Kaplow, "A survey of point-based POMDP solvers," *Auton. Agent Multi-Agent Syst.*, vol. 27, pp. 1–51, 2013.
[5] J. Pineau, G. Gordon, and S. Thrun, "Anytime point-based approximations for large POMDPs," *J. Artif. Intell. Res.*, vol. 27, pp. 335–380, 2006.
[6] T. Smith and R. Simmons, "Heuristic search value iteration for POMDPs," in *Proc. 20th Conf. UAI*, 2004.
[7] H. Kurniawati, D. Hsu, and W. S. Lee, "SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces." in *Proc. Robot. Sci. Sys.*, 2008.
[8] T. Smith and R. Simmons, "Point-based POMDP algorithms: Improved analysis and implementation," in *Proc. 21st Conf. UAI*, 2005.
[9] A. Nayyar, T. Başar, D. Teneketzis, and V. V. Veeravalli, "Optimal strategies for communication and remote estimation with an energy harvesting sensor," *IEEE Trans. Automat. Contr.*, vol. 58, no. 9, pp. 2246–2260, 2013.
[10] A. Mahajan, "Optimal decentralized control of coupled subsystems with control sharing," *IEEE Trans. Automat. Contr.*, vol. 58, no. 9, pp. 2377–2382, 2013.
[11] H. Tavafoghi, Y. Ouyang, and D. Teneketzis, "A unified approach to dynamic decision problems with asymmetric information: Nonstrategic agents," *IEEE Trans. Automat. Contr.*, vol. 67, no. 3, pp. 1105–1119, 2022.
[12] J. Subramanian, A. Sinha, R. Seraj, and A. Mahajan, "Approximate information state for approximate planning and reinforcement learning in partially observed systems." *J. Mach. Learn. Res.*, vol. 23, pp. 12–1, 2022.
[13] K. Zhang, E. Miehling, and T. Başar, "Online planning for decentralized stochastic control with partial history sharing," in *Proc. ACC*, 2019, pp. 3544–3550.
[14] H. Kao and V. Subramanian, "Common information based approximate state representations in multi-agent reinforcement learning," in *Proc. AISTATS*. PMLR, 2022, pp. 6947–6967.
[15] J. Foerster, F. Song, E. Hughes, N. Burch, I. Dunning, S. Whiteson, M. Botvinick, and M. Bowling, "Bayesian action decoder for deep multi-agent reinforcement learning," in *Proc. ICML*, 2019.
[16] C. Amato, G. Chowdhary, A. Geramifard, N. K. Üre, and M. J. Kochenderfer, "Decentralized control of partially observable Markov decision processes," in *Proc. IEEE CDC*, 2013, pp. 2398–2405.
[17] A. Kumar and S. Zilberstein, "Point-based backup for decentralized POMDPs: Complexity and new algorithms," in *9th Int. Conf. Auton. Agents and Multiagent Syst. (AAMAS)*, 2010.
[18] L. C. MacDermed and C. L. Isbell, "Point based value iteration with optimal belief compression for Dec-POMDPs," *Adv. Neural. Inf. Process. Syst.*, vol. 26, 2013.
[19] J. S. Dibangoye, C. Amato, O. Buffet, and F. Charpillet, "Optimally solving Dec-POMDPs as continuous-state MDPs," *J. Artif. Intell. Res.*, vol. 55, pp. 443–497, 2016.
[20] J. Pajarinen and J. Peltonen, "Periodic finite state controllers for efficient POMDP and DEC-POMDP planning," *Proc. NIPS*, 2011.
[21] J. S. Dibangoye, O. Buffet, and F. Charpillet, "Error-bounded approximations for infinite-horizon discounted decentralized POMDPs," in *Proc. ECML PKDD*, 2014, pp. 338–353.
[22] S. Adhikari and P. Gmytrasiewicz, "Point based solution method for communicative IPOMDPs," in *Proc. Euro. Conf. Multi-Agent Syst. (EUMS)*, 2021, pp. 245–263.
[23] N. L. Zhang and W. Zhang, "Speeding up the convergence of value iteration in partially observable Markov decision processes," *J. Artif. Intell. Res.*, vol. 14, pp. 29–51, 2001.
[24] M. Hauskrecht, "Value-function approximations for partially observable Markov decision processes," *J. Artif. Intell. Res.*, vol. 13, pp. 33–94, 2000.
[25] M. T. Spaan and N. Vlassis, "Perseus: Randomized point-based value iteration for POMDPs," *J. Artif. Intell. Res.*, vol. 24, pp. 195–220, 2005.
[26] P. Poupart, K.-E. Kim, and D. Kim, "Closing the gap: Improved bounds on optimal POMDP solutions," in *Proc. ICAPS*, vol. 21, 2011, pp. 194–201.
[27] G. Gallo and A. Ülkücü, "Bilinear programming: An exact algorithm," *Math. Program.*, vol. 12, no. 1, pp. 173–194, 1977.
[28] H. Konno, "A cutting plane algorithm for solving bilinear programs," *Math. Program.*, vol. 11, no. 1, pp. 14–27, 1976.
[29] R. Cogill and S. Lall, "An approximation algorithm for the discrete team decision problem," *SIAM J. Contr. Optim.*, vol. 45, no. 4, pp. 1359–1368, 2006.
[30] M. Petrik and S. Zilberstein, "Average-reward decentralized Markov decision processes." in *Proc. IJCAI*, 2007, pp. 1997–2002.
[31] ——, "A bilinear programming approach for multiagent planning," *J. Artif. Intell. Res.*, vol. 35, pp. 235–274, 2009.
[32] "Nonconvex quadratic optimization," https://www.gurobi.com/events/non-convex-quadratic-optimization-2/, accessed: 03/21/2023.
[33] J. Tsitsiklis and M. Athans, "On the complexity of decentralized decision making and detection problems," *IEEE Trans. Automat. Contr.*, vol. 30, no. 5, pp. 440–446, 1985.
[34] R. Nair, M. Tambe, M. Yokoo, D. Pynadath, and S. Marsella, "Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings," in *Proc. IJCAI*, 2003.

APPENDIX

A. Experimental Environments

In this section, we introduce the environments we use for our experimental results in Section IV.

1) DecTiger: We consider a parameterized extension of the DecTiger [34], [19] problem with three parameters (N, d, β): Two agents face N closed doors. Behind one of the doors is a ferocious tiger, while behind all other doors are boxes of tiger-shaped cookies. At each time, each agent has $N + 1$ choices: open one of the doors, or listen for the roar of the tiger. However, listening has a cost and is prone to error: an agent hears the roar from the correct door with probability $p = 0.85/(0.7 + 0.15N)$ and any other door with probability $q = 0.15/(0.7 + 0.15N)$. The instantaneous reward for the agents are listed in Table III. The discount factor is β. Initially, the tiger is placed behind a uniform random door. The state of the world does not change if both agents choose to listen. If any door is opened by either agent, then the tiger resets itself to a uniform random location. If both agents choose to listen, then at the next time step, they will obtain conditionally independent observations each following the distribution described above. If either of the agents opens any door, then all observations will be independently uniform random and independent of the tiger's actual position, regardless of the other agent's action.

$a^1 \backslash a^2$	listens	open tiger door	open other door
listens	-2	-101	$\frac{20}{N} - 1$
open tiger door	-101	-50	-100
open other door	$\frac{20}{N} - 1$	-100	$\frac{40}{N}$

TABLE III
INSTANTANEOUS REWARDS FOR DECTIGER(N, d, β)

In the original setup [34], [19], the agents' actions and observations are private information, as it is the tradition for the Dec-POMDP model. In this work, we consider a delayed information sharing model of this problem: The agents share their actions and observations with each other but with a delay of $d > 0$. More specifically, at time t, each agent has access to the other agent's actions taken at anytime before (not including) $t-d$ and the observations yielded from these actions. For agent i, the actions taken after (including) time $t - d$ and their resulting observations are private at time t.

`DecTiger` (N, d, β) can be modeled with our multi-agent control model introduced in Section II-A.1 as follows: $\mathcal{S} = [N]$ represents the N possible locations of the tiger. $\mathcal{A}^i =$

$[N+1]$ represent the choices of agent i at any given time. Let $\mathcal{Z}^i = [N]$ represent the space of observations of agent i and $\mathcal{Z} = \mathcal{Z}^1 \times \mathcal{Z}^2$. Then $\mathcal{M}^i = \{\emptyset\} \cup \bigcup_{k=1}^{d} (\mathcal{Z}^i \times \mathcal{A}^i)^k$ is the space of actions and observations of agent i that have not been known to the other agent yet. (The symbol $\emptyset$ represents the nonexistence of such action and observations at the beginning.) The common observation space $\mathcal{O} = \{\emptyset\} \cup (\mathcal{Z} \times \mathcal{A})$ is the space of newly shared actions and observations, which are the actions and observations d-steps ago. (The symbol $\emptyset$ represents that the first piece of information shared by the agents has not arrive yet.) The rest of the tuple making up the model $\mathcal{E}$ (i.e. $b_0, \mathbb{P}, r$) can be defined naturally.

2) MultiCast: We consider a two-agent multi-access broadcast system model inspired by [10] and [20]. The model is parameterized by $(C^1, C^2, p^1, p^2, \beta)$. Each agent i has a buffer with size C^i. Initially, the buffers are empty. At each time t, a packet arrives at agent i's buffer with probability p^i, independent of all history and the packet arrival event of the other agent. If the buffer is full, then the packet is dropped and the agents pay a penalty of $c_D = 2$. Otherwise, the packet is placed into the buffer. Both users may attempt to transmit a packet over a shared broadcast medium. If only one agent attempt to transmit, then the packet is successfully transmitted and removed from the buffer. If both agent attempt to transmit at the same time, then transmissions of both agents fail due to collision. An attempt to transmit, whether successful or not, costs $c_T = 0.5$. In addition to the drop penalty and transmission cost, the agents pay a penalty of 1 for each unit of time each packet stays in the buffer. The discount factor for the penalties and costs is β. The agents know their own packet arrival histories but not the other agents. An agent can also observe the other agent's action after it is taken.

It can be shown that [10], [11] in this model, the number of packets in the buffer for each agent i is a *sufficient private information* for decision making. Therefore, per Remark 1, `MultiCast` $(C^1, C^2, p^1, p^2, \beta)$ can be modeled with our multi-agent control model in Section II-A.1 as follows: $\mathcal{S}^i = \{0, 1, \cdots, C^i\}$ is the set of private states for agent i indicating the number of packets in its buffer. $\mathcal{S} = \mathcal{S}^1 \times \mathcal{S}^2$ is the state space. $\mathcal{A}^i = \{\mathrm{NT}, \mathrm{T}\}$ represent the two choices of agent i at any given time. $\mathcal{M}^i = \mathcal{S}^i$ is the space of sufficient private information. $\mathcal{O} = \mathcal{A}$ is the space of common observations. The instantaneous reward r is defined by

$$r(s,a) = \sum_{i=1}^{2} r^i(s^i, a^i)$$
$$r^i(s^i, a^i) = -s^i - c_D \cdot p^i \mathbf{1}_{\{s^i = C^i\}} - c_T \cdot \mathbf{1}_{\{a^i = \mathrm{T}\}}$$

The rest of the tuple making up the model $\mathcal{E}$ (i.e. $b_0, \mathbb{P}$) can be defined naturally.

B. More on the New Upper Bound Representation

In this section we describe more types of α-constraints for the upper bound representation described in Section III-B.

1) Lower Bound Based Constraints: For any belief b, we can derive a *lower bound* $\underline{v}^b$ on $\min_{\alpha \in \mathcal{V}^*} \alpha^T b$ through considering a reward *minimization* POMDP. Then, we can add the inequality $b^T y \geq \underline{v}^b$ to the set of α-constraints. A crude lower bound of this kind can be efficiently computed through the Fast Informed Bound method [24].

2) Marginal Belief Based Constraints: Oftentimes, we can derive an upper bound of $V^*(b)$ as a function of some marginal distribution of b, i.e. suppose that $\mathcal{S} = \mathcal{K}^1 \times \mathcal{K}^2$, we have $V^*(b) \leq \bar{v}(\varphi(b))$ for all $b \in \Delta(\mathcal{S})$ where $\varphi(b) \in \Delta(\mathcal{K}^1), [\varphi(b)](k^1) = \sum_{k^2 \in \mathcal{K}^2} b(k^1, k^2)$. An example is the upper bound for a coordinator's POMDP derived by solving a relaxed problem where part or all of the private information is assumed to be commonly known. (The relaxed problem can be much easier to solve due to the fact that it features a smaller private information space and a smaller augmented state space.) This kind of upper bound can be easily translated into α-constraints: Suppose that we have an upper bound $\bar{v}^{b^1}$ at a fixed $b^1 \in \Delta(\mathcal{K}^1)$, then we know that all α-vectors in $\mathcal{V}^*$ satisfy the constraint

$$\max_{b:\varphi(b)=b^1} b^T y \leq \bar{v}^{b^1}. \tag{19}$$

We can transform (19) into a group of finitely many linear constraints through the following steps: First note that

$$\max_{b:\varphi(b)=b^1} b^T y = \sum_{k^1 \in \mathcal{K}^1} b^1(k^1) \max_{k^2 \in \mathcal{K}^2} y(k^1, k^2).$$

Therefore, by adding an auxiliary vector $\bar{y} \in \mathbb{R}^{\mathcal{K}^1}$, we obtain the α-constraints:

$$\begin{aligned} &y(k^1, k^2) \leq \bar{y}(k^1) \quad \forall k^1 \in \mathcal{K}^1, k^2 \in \mathcal{K}^2, \\ &(b^1)^T \bar{y} \leq \bar{v}^{b^1}. \end{aligned} \tag{20}$$

In the case that a lower bound for the reward minimization problem can be expressed as a function of the marginal belief, one can use a similar way to obtain the corresponding α-constraints as well. Suppose that $\underline{v}^{b^1}$ is a lower bound of the minimum total reward at belief b for all b such that $\varphi(b) = b^1$, then we can use an auxiliary vector $\underline{y} \in \mathbb{R}^{\mathcal{K}^1}$ to formulate the α-constraints:

$$\begin{aligned} &y(k^1, k^2) \geq \underline{y}(k^1) \quad \forall k^1 \in \mathcal{K}^1, k^2 \in \mathcal{K}^2, \\ &(b^1)^T \underline{y} \geq \underline{v}^{b^1}. \end{aligned} \tag{21}$$

C. Additional Figures for Experimental Results

In the CHSVI algorithm, the memory use is mainly from two sources: (1) α-vectors for the lower bound representation, and (2) α-constraints for the upper bound representation. We record the number of α-vectors and α-constraints[4] during each run of the algorithm. The results are shown in Figure 3 and 4. The zigzag pattern is due to periodic pruning of α-vectors and α-constraints.

As is the case for HSVI on single-agent POMDPs, the pruning procedure plays a crucial role in reducing the

[4] α-constraints that relate auxiliary variables to main variables, i.e. (20)(21), are not included.

and memory complexity of the algorithm. It also play a significant role in reducing the time complexity since stage optimization problems with fewer α-vectors or fewer α-constraints are easier to solve. From the figures, we observe that in all cases, the majority of α-vectors and α-constraints are those for stage 2. This is to be expected since stage 2 has the largest state space of the three stages, resulting in a large dimension of the belief space. Since a larger state space also means larger memory use for each α-vector or α-constraint, the stage 2 α-vectors and α-constraints dominate the memory use. In `DecTiger`(2,1,0.99) and `DecTiger`(3,1,0.9), we observe that the number of α-vectors and α-constraints grows with a decreasing rate until it stabilizes well before the termination of the algorithm. We conjecture that the same behaviour holds for `DecTiger`(3,1,0.99) if it were given enough run time.

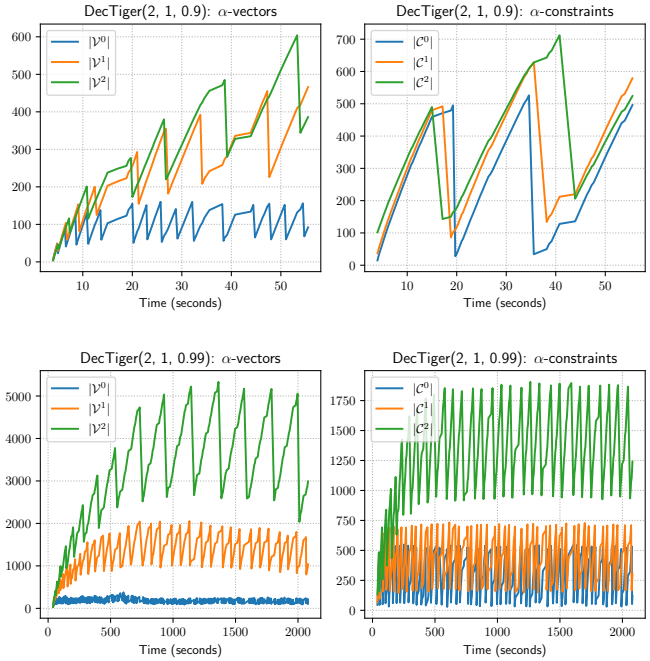

Fig. 3. Number of α-vectors and α-constraints for `DecTiger` $(2, 1, \beta)$. $\mathcal{V}^\ell$ is the set of α-vector for stage ℓ defined in Algorithm 4. $\mathcal{C}^\ell$ is the set of α-constraints for stage ℓ defined in Algorithm 5.

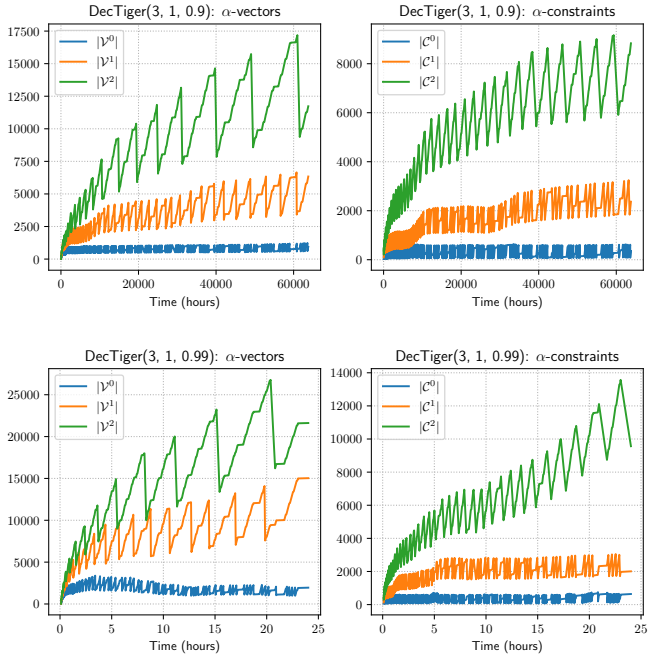

Fig. 4. Number of α-vectors and α-constraints for `DecTiger` $(3, 1, \beta)$. $\mathcal{V}^\ell$ is the set of α-vector for stage ℓ defined in Algorithm 4. $\mathcal{C}^\ell$ is the set of α-constraints for stage ℓ defined in Algorithm 5.