2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

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.

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 [2]. 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 [3], [4]. However, these methods were practical only for problems with very few states [5]. A class of approximate solution methods, called *point-based* methods [5] have been quite successful in approximately solving POMDP problems with hundreds or thousands of states. 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 [5] 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 definition in [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: While many theoretical results have been developed using the CI approach [2], [8], 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 [9], or through machine learning techniques [10].

The model of Dec-POMDP is a multi-agent extension of POMDP that has been studied extensively (see [11] for a survey), where point-based methods have been applied (e.g. [12]). In many Dec-POMDP algorithms, the policy are represented as policy trees [12], 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 [13], [14]. 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 [15], the authors applied point-based methods to finite horizon problem with communication between agents. However, to

Full paper is at [1].

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

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 performance bounds.

A quick word on notations: $\Delta(\mathcal{X})$ is the set of probability distributions on a set $\mathcal{X}$. For a function $f : \mathcal{X} \mapsto \Delta(\mathcal{Y})$, we write $f(y|x) = [f(x)](y)$. The letter $\mathbb{P}$ is reserved for transitions kernels. Any expression that involves a belief $b \in \Delta(\mathcal{S})$ in place of a state $s \in \mathcal{S}$ (e.g. $\mathbb{P}(o|b, a), r(b)$) is understood as a weighted average.

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; $b_0 \in \Delta(\mathcal{S} \times \mathcal{M})$ 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 $\mathcal{H}^i_t = (\mathcal{O})^t \times \mathcal{M}^i$ 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^i_t 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 $\pi = (\pi^i)_{i\in\mathcal{I}}, \pi^i = (\pi^i_t)^\infty_{t=0}, \pi^i_t : \mathcal{H}^i_t \mapsto \mathcal{A}^i$ to maximize the expected total discounted-reward $J(\pi) := \mathbb{E}^\pi \left[\sum_{t=0}^\infty \beta^t r(s_t, a_t)\right]$.

As shown in [2], 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.

2) Transformation into Coordinator's POMDP: Following the CI approach introduced in [2], 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 $m_s = (m^i_s)_{i\in\mathcal{I}} \in \mathcal{M}$ 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 1. The coordinator's POMDP can have a prohibitively large action space even for seemingly small problems like the DecTiger problem with $N \leq 3$ doors and 1-step delayed information sharing (described in [1]). For example, with $N = 2$, we have $|\bar{\mathcal{A}}| = \prod_{i=1}^2 |\mathcal{A}^i|^{|\mathcal{M}^i|} = 3^{14} \approx 4.78 \times 10^6$; with $N = 3$, we have $|\bar{\mathcal{A}}| = \prod_{i=1}^2 |\mathcal{A}^i|^{|\mathcal{M}^i|} = 4^{26} \approx 4.50 \times 10^{15}$. 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. A piecewise-linear convex function V 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. The Bellman update of such a function is given by $TV(b) = \max_{a\in\mathcal{A}} \max_{\mu\in\mathcal{V}^\mathcal{O}} (\alpha^{a,\mu})^T b$ 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 iteration algorithm can be achieved by updating the collection of α-vectors [3], [4], [5] 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 [4], the exact value iteration is still ill-equipped to handle large state and action spaces [5].

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

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

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

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

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.

2) Heuristic Search Value Iteration: The HSVI algorithm is a point-based algorithm first introduced in [6]. 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^* to guide exploration of beliefs. The algorithm repeatedly performs depth-first searches to select a few belief points and updates upper and lower bounds at those beliefs. HSVI is an anytime algorithm: the algorithm can be terminated either by user or after certain stopping criteria is met. When it terminates, it returns the direct control policy[1] associated with the α-vectors.

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 as described in (2) – (4). 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 $b \in \Delta(\mathcal{S})$,

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

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 computes

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

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.

[1]The direct control policy [16] can be described as follows: At belief b, find $\alpha^* \in \mathcal{V}$ 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.

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 $(s, \gamma^1(m^1_s)) \in \mathcal{S} \times \mathcal{A}^1$.

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 $(s, a^1, \gamma^2(m^2_s)) \in \mathcal{S} \times \mathcal{A}^1 \times \mathcal{A}^2$.

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 1), 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.

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 1: Coordinator's HSVI Algorithm

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

$$[T^\ell L^{\ell+1}](b) = \max_{\gamma^i \in \bar{\mathcal{A}}^i} L^{\ell+1}(\tau^\ell(b, \gamma^i))$$
$$= \max_{\gamma^i \in \bar{\mathcal{A}}^i} \max_{\alpha \in \mathcal{V}^i} \sum_{(s^\ell, a^i) \in \mathcal{S}^\ell \times \mathcal{A}^i} b(s^\ell)\gamma^i(a^i|m^i_{s^\ell})\alpha(s^\ell, a^i) \quad (6)$$

The optimization problem (6) can be solved via the following steps:

$$\gamma^{i,\alpha}(m^i) := \arg\max_{a^i \in \mathcal{A}^i} \sum_{s^\ell \in \mathcal{S}^\ell : m^i_{s^\ell} = m^i} b(s^\ell)\alpha(s^\ell, a^i) \quad (7)$$
$$\forall m^i \in \mathcal{M}^i, \alpha \in \mathcal{V}^i$$
$$J(\alpha) := \sum_{s^\ell \in \mathcal{S}^\ell} b(s^\ell)\alpha(s^\ell, \gamma^{i,\alpha}(m^i_{s^\ell})) \quad \forall \alpha \in \mathcal{V}^i \quad (8)$$
$$\alpha^* := \arg\max_{\alpha \in \mathcal{V}^i} J(\alpha) \quad (9)$$
$$\gamma^{i,*} := \gamma^{i,\alpha^*}, \quad (10)$$

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 $\mathcal{V}^\ell$:

$$\alpha^b(s^\ell) = \sum_{a^i \in \mathcal{A}^i} \gamma^{i,*}(a^i|m^i_{s^\ell})\alpha^*(s^\ell, a^i) \quad \forall s^\ell \in \mathcal{S}^\ell. \quad (11)$$

Remark 2. 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 (7) – (11) is $\Theta(|\mathcal{S}||\mathcal{V}^i||\mathcal{A}^i||\mathcal{M}^i|)$, which is polynomial in all parameters involved.

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

$$[T^2 L^0](b) = r(b) + \beta \sum_{o \in \mathcal{O}} \mathbb{P}(o|b) \max_{\alpha \in \mathcal{V}^0} [\tau^2(b,o)]^T \alpha$$
$$= r(b) + \beta \sum_{o \in \mathcal{O}} \max_{\alpha \in \mathcal{V}^0} \sum_{s^2 \in \mathcal{S}^2} b(s^2) \sum_{s' \in \mathcal{S}} \mathbb{P}(s', o|s^2)\alpha(s')$$

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

$$\alpha^{b,o} := \arg\max_{\alpha \in \mathcal{V}^0} \sum_{s^2 \in \mathcal{S}^2} b(s^2) \sum_{s' \in \mathcal{S}} \mathbb{P}(s', o|s^2)\alpha(s') \quad \forall o \in \mathcal{O} \quad (12)$$
$$\alpha^b(s^2) := r(s^2) + \beta \sum_{o \in \mathcal{O}} \sum_{s' \in \mathcal{S}} \mathbb{P}(s', o|s^2)\alpha^{b,o}(s') \quad \forall s^2 \in \mathcal{S}^2 \quad (13)$$

The lower bound update algorithm is given in Algo. 2.

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 (7) – (11);
  else
    Compute α^b with (12)(13);
  Add α^b to $\mathcal{V}^\ell$;
  Prune dominated vectors in $\mathcal{V}^\ell$ once in a while;

Algorithm 2: 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. The new upper bound is tighter than the ones used in the original HSVI. Due to limited space, we only provide a brief description of the new representation. Interested readers can refer to [1] for details.

To describe the new upper bound representation, we first note the dual form of (5) is

$$U(b) = \max\{b^T y : y \in \mathbb{R}^{\mathcal{S}}, \mathbf{B}^T y \leq \bar{v}\}, \tag{14}$$

which can be directly interpreted 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 (14) 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, (14) 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 (14), we have

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

to be a potentially tighter upper bound.

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

$$\begin{aligned}
\bar{v}^b &:= [T^\ell U^{\ell+1}](b) = \max_{\gamma^i \in \bar{\mathcal{A}}^i} U^{\ell+1}(\tau^\ell(b, \gamma^i)) \\
&= \max_{\gamma^i \in \bar{\mathcal{A}}^i} \max_{\substack{y \in \mathbb{R}^{\mathcal{S}^i} \\ \mathbf{M}^i y \leq w^i}} \sum_{(s^\ell, a^i) \in \mathcal{S}^\ell \times \mathcal{A}^i} b(s^\ell) \gamma^i(a^i | m^i_{s^\ell}) y(s^\ell, a^i) \\
&=: \max_{\gamma^i \in \bar{\mathcal{A}}^i} \max_{\substack{y \in \mathbb{R}^{\mathcal{S}^i} \\ \mathbf{M}^i y \leq w^i}} J^i(b, \gamma^i, y)
\end{aligned} \tag{15}$$

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, (15) is a *bilinear programming* (BP) problem, which has been studied extensively in the optimization as well as decentralized control literature. Notably, in [17], the author provided a method to convert bilinear programs to MILPs. Gurobi Optimization™ has also developed specialized solvers for bilinear programs [18].

Remark 3. Even though bilinear programming problems are NP-hard [19] 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 (15) 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 [17] of BP).

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

$$\begin{aligned}
\bar{v}^b &= [T^2 U^0](b) \\
&= r(b) + \beta \sum_{o \in \mathcal{O}} \mathbb{P}(o|b) \max_{\substack{y \in \mathbb{R}^{\mathcal{S}} \\ \mathbf{M}^0 y \leq w^0}} [\tau^2(b, o)]^T y
\end{aligned} \tag{16}$$

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

The upper bound update algorithm is given in Algo. 3.

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 (15);
- **else**
  - Compute $\bar{v}^b$ with (16);
  - 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 3: 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 [16]. The α-constraints used in upper bounds are initialized to be the marginal belief based constraints 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 several `DecTiger` instances defined in [1]. 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 I and Figure 1. Additional experimental results are provided in [1]. In the first three instances, 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

TABLE I

EXPERIMENTAL RESULTS ON `DecTiger` (N, d, β).

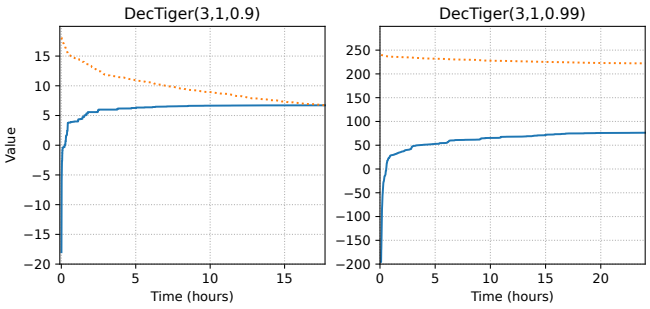

Fig. 1. Experimental Results on `DecTiger` $(3, 1, \beta)$. The dotted line represents the upper bound $U(b_0)$ and the solid line represents the lower bound $L(b_0)$.

It took the algorithm more than 18 hours to reduce the gap between upper and lower bound to 0.01 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). The remainder of the algorithm mostly reduces the upper bound to provide an optimality guarantee. This means that even if we terminate the algorithm early for `DecTiger`(3,1,0.9), we can still obtain a strategy with near optimal performance. 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 `DecTiger` is out of the question on our computer: We have tried to apply the state-of-the-art HSVI2 solver [16] on a coordinator's POMDP with 25 states, 4 observations, and 2^{10} actions/prescriptions. The program ended up using all of the memory, causing the computer to freeze. In comparison, coordinator's POMDP of `DecTiger`(2,1,β) has 74 states, 37 observations, and 3^{14} actions/prescriptions, while `DecTiger`(3,1,β) has 435 states, 145 observations, and 4^{26} actions/prescriptions.

V. CONCLUSIONS

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] D. Tang, A. Nayyar, and R. Jain, "A novel point-based algorithm for multi-agent control using the common information approach," *arXiv preprint arXiv:2304.04346*, 2023.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[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] A. Mahajan, "Optimal decentralized control of coupled subsystems with control sharing," *IEEE Trans. Automat. Contr.*, vol. 58, no. 9, pp. 2377–2382, 2013.

[9] K. Zhang, E. Miehling, and T. Başar, "Online planning for decentralized stochastic control with partial history sharing," in *Proc. ACC*, 2019, pp. 3544–3550.

[10] 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.

[11] 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.

[12] 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.

[13] J. Pajarinen and J. Peltonen, "Periodic finite state controllers for efficient POMDP and DEC-POMDP planning," *Proc. NIPS*, 2011.

[14] 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.

[15] S. Adhikari and P. Gmytrasiewicz, "Point based solution method for communicative IPOMDPs," in *Proc. Euro. Conf. Multi-Agent Syst. (EUMS)*, 2021, pp. 245–263.

[16] T. Smith and R. Simmons, "Point-based POMDP algorithms: Improved analysis and implementation," in *Proc. 21st Conf. UAI*, 2005.

[17] M. Petrik and S. Zilberstein, "Average-reward decentralized Markov decision processes." in *Proc. IJCAI*, 2007, pp. 1997–2002.

[18] "Nonconvex quadratic optimization," https://www.gurobi.com/events/non-convex-quadratic-optimization-2/, accessed: 03/21/2023.

[19] 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.