Journal of Artificial Intelligence Research 55 (2016) 443-497 Submitted 11/14; published 02/16

Optimally Solving Dec-POMDPs as Continuous-State MDPs

Jilles Steeve Dibangoye JILLES-STEEVE.DIBANGOYE@INSA-LYON.FR
Univ de Lyon
INSA-Lyon, CITI-Inria, F-69621, France

Christopher Amato CAMATO@CS.UNH.EDU
University of New Hampshire
Durham, NH, USA

Olivier Buffet OLIVIER.BUFFET@INRIA.FR
François Charpillet FRANCOIS.CHARPILLET@INRIA.FR
Inria — Université de Lorraine — CNRS
Villers-lès-Nancy, F-54600, France

Abstract

Decentralized partially observable Markov decision processes (Dec-POMDPs) provide a general model for decision-making under uncertainty in decentralized settings, but are difficult to solve optimally (NEXP-Complete). As a new way of solving these problems, we introduce the idea of transforming a Dec-POMDP into a continuous-state deterministic MDP with a piecewise-linear and convex value function. This approach makes use of the fact that planning can be accomplished in a centralized offline manner, while execution can still be decentralized. This new Dec-POMDP formulation, which we call an *occupancy MDP*, allows powerful POMDP and continuous-state MDP methods to be used for the first time. To provide scalability, we refine this approach by combining heuristic search and compact representations that exploit the structure present in multi-agent domains, without losing the ability to converge to an optimal solution. In particular, we introduce a feature-based heuristic search value iteration (FB-HSVI) algorithm that relies on feature-based compact representations, point-based updates and efficient action selection. A theoretical analysis demonstrates that FB-HSVI terminates in finite time with an optimal solution. We include an extensive empirical analysis using well-known benchmarks, thereby demonstrating that our approach provides significant scalability improvements compared to the state of the art.

1. Introduction

Many significant real-world problems involve decision making in sequential multiagent environments. Examples include: exploration robots that must coordinate to perform experiments on an unknown planet (Zilberstein, Washington, Bernstein, & Mouaddib, 2002); rescue robots that, after a disaster, must safely find victims as quickly as possible (Paquet, Chaib-draa, Dallaire, & Bergeron, 2010); optimized distributed congestion control in a noisy computer network (Winstein & Balakrishnan, 2013); sensor networks where multiple sensors work jointly to perform a large-scale sensing task under strict power constraints (Jain, Taylor, Tambe, & Yokoo, 2009); or robot logistics problems with communication limitations and sensor uncertainty (Amato, Konidaris, Anders, Cruz, How, & Kaelbling, 2015). All these tasks require multiple decision makers, or agents, to coordinate their actions in order to achieve common long-term goals. Additionally, uncertainty is ubiquitous in these domains, both in the effects of actions and in the information received by the agents.

Markov decision processes (MDPs) address uncertainty in system dynamics, but assume centralization. Standard methods for solving MDPs, *e.g.,* linear and dynamic programming (Bellman, 1957; Puterman, 1994) and heuristic search (Barto, Bradtke, & Singh, 1995; Hansen & Zilberstein, 2001), are centralized during both the planning and the execution phases. Partially observable Markov decision processes (POMDPs) extend MDPs to situations in which there is uncertainty over the state of the system (Kaelbling, Littman, & Cassandra, 1998; Smith & Simmons, 2006; Shani, Pineau, & Kaplow, 2013), but similarly assume centralized planning and execution.

To use the MDP and POMDP models when multiple agents are present, a centralized coordinator agent must have a global view of the underlying state (or belief state in the case of a POMDP) of the entire system, and plan on behalf of its teammates. Every time step, this agent would transmit the appropriate action that each agent must perform and then observe the resulting state (or observations of each agent in a POMDP). These methods, collectively referred to as *centralized planning for centralized control*, assume agents communicate at each step with no delay or cost, either explicitly through messages or implicitly through observations. Unfortunately, in many practical applications, agents are not permitted to share their information with no delay or cost; rather, each agent possesses only local, unshared observations, and acts without full knowledge of what others observe or plan to do. These characteristics have led to the development of a rich body of research on decentralized decision-making under uncertainty.

The decentralized partially observable Markov decision process (Dec-POMDP) is a standard formulation for cooperative decision-making in these sequential settings without instantaneous, free and noiseless communication (Bernstein, Givan, Immerman, & Zilberstein, 2002). Over the past decade, there has been extensive research on solution methods for Dec-POMDPs, using methods such as *dynamic programming* (Hansen, Bernstein, & Zilberstein, 2004; Boularias & Chaib-draa, 2008; Amato, Dibangoye, & Zilberstein, 2009), *optimization* (Aras & Dutech, 2010; Amato, Bernstein, & Zilberstein, 2010) and *heuristic search* (Szer, Charpillet, & Zilberstein, 2005; Oliehoek, Spaan, & Vlassis, 2008; Oliehoek, Spaan, Amato, & Whiteson, 2013). These approaches directly search for an optimal solution in the space of possible solutions (or policies) but become intractable for larger problems. This is not unexpected given the worst-case NEXP complexity of finite-horizon Dec-POMDPs (Bernstein et al., 2002).

A key assumption in many Dec-POMDP algorithms is that planning can be centralized as long as execution remains decentralized (Hansen et al., 2004; Boularias & Chaib-draa, 2008; Amato et al., 2009; Szer et al., 2005; Oliehoek et al., 2008, 2013). That is, these methods represent *centralized planning for decentralized control* — a centralized planner generates a tuple of individual solutions, one individual solution for each agent. While the use of the Dec-POMDP model does not require centralized planning (e.g., Velagapudi, Varakantham, Sycara, & Scerri, 2011; Wu, Zilberstein, & Chen, 2011; Banerjee, Lyle, Kraemer, & Yellamraju, 2012), because Dec-POMDPs are a cooperative framework it is common to assume that centralized planning is possible. Unfortunately, current algorithms do not take full advantage of this centralized planning assumption.

This article extends a conference paper published at IJCAI'13 (Dibangoye, Amato, Buffet, & Charpillet, 2013), which includes the introduction of a centralized solution method that recasts a Dec-POMDP as a *continuous-state MDP* with more detailed background, new theorems and proofs, as well as more concise representations of policies and value functions. Our novel method is also able to produce decentralized solutions, but leverages work on centralized planning methods to significantly increase scalability. Furthermore, we show that the optimal value function of the aforementioned MDP is a piecewise-linear and convex function. In this form, theory from POMDPs

(Kaelbling et al., 1998) applies, allowing POMDP algorithms to produce optimal solutions for Dec-POMDPs. A wide range of POMDP algorithms, which have demonstrated significant scalability (Shani et al., 2013), can now be applied. We extend one such heuristic search algorithm (Smith & Simmons, 2004) to the Dec-POMDP case, but because the number of states and actions in this MDP grows exponentially with the planning horizon, scalability remains limited.

To increase scalability, we introduce a novel mechanism that refines this centralized solution methodology and present ways to combine classical heuristic search and compact representations, without losing the ability to converge to an optimal solution. To incorporate compact representations, we build on *feature-based dynamic programming* (Tsitsiklis & van Roy, 1996), which includes feature extraction and value prediction with approximation methods. We introduce the feature-based heuristic search value iteration (FB-HSVI) algorithm that relies on compact representations, point-based updates and efficient action selection. A theoretical analysis demonstrates that FB-HSVI terminates in finite time with an optimal solution. This combination of POMDP theory and compact representations can greatly reduce the problem size and solution efficiency while retaining optimal solutions for Dec-POMDPs.

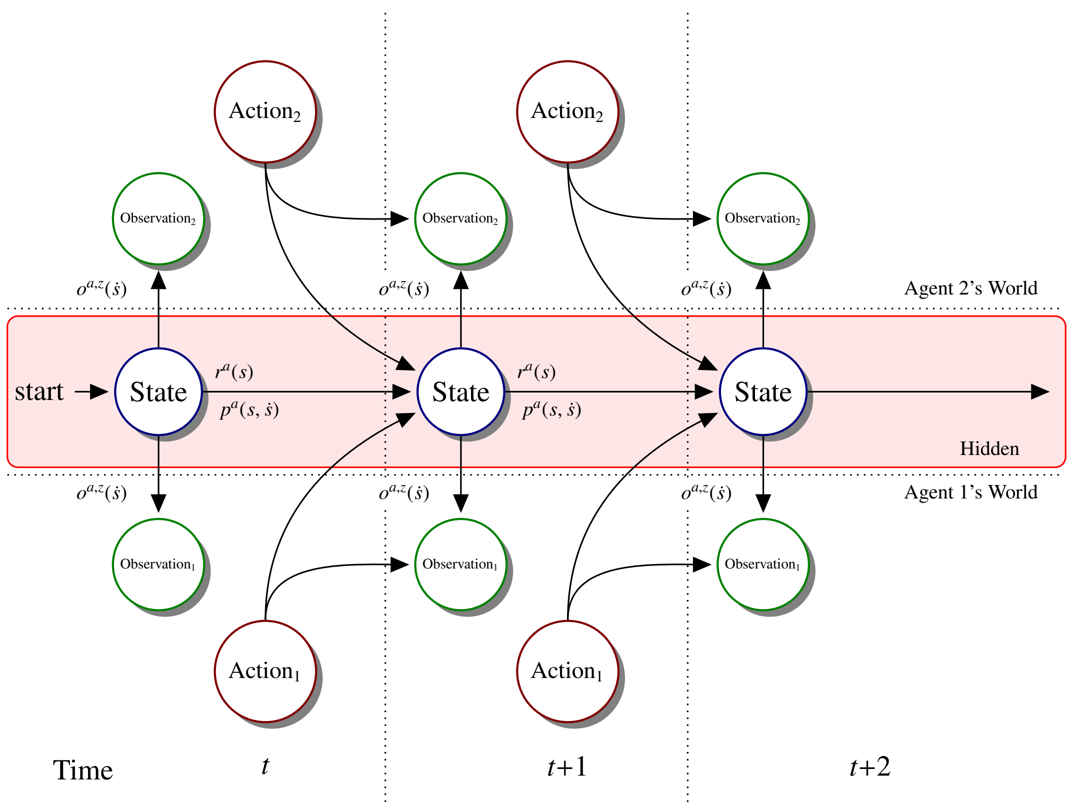

Figure 1: A graphical model of the two-agent Dec-POMDP model.

2. Background on Dec-POMDPs

This section introduces the basic components of decentralized partially observable Markov decision processes (Dec-POMDPs).

2.1 Problem Definition and Notations

Consider multiple agents that are faced with the task of influencing a stochastic system as it evolves over time (see the two agent case in Figure 1). At every time step, each agent receives a private observation that gives (possibly) incomplete and noisy information about the current state of the system. Since the states are not observable, an agent cannot choose its actions based on the states. Instead, it can consider a complete history of its past actions and observations to choose an action. Actions produce a common immediate reward, and the system evolves to a new state at the next time step according to a probability distribution conditioned on actions. At the next time step, agents face a similar problem again, but now the system may be in a different state. The goal of agents is to choose the actions based on these local action-observation sequences which cause the system to perform optimally with respect to a shared performance criterion (which we discuss below). The Dec-POMDP model formalizes these interactions between agents and the system. This paper formulates and solves this general decentralized stochastic control problem for a process that operates for a finite planning horizon.

Definition 1. *A **Dec-POMDP** is represented as a tuple $M \equiv (I, S, A, Z, p, o, r, b_0, T)$, where:*

- *I is a finite set of **agents** $i \in \{1, 2, \dots, |I|\}$;*
- *S is a finite set of n **states**;*
- *A^i is the finite set of agent i's **actions**; $A \equiv \times_i A^i$ is the finite set of joint actions;*
- *Z^i is the finite set of agent i's **observations**; $Z \equiv \times_i Z^i$ is the finite set of joint observations;*
- *$p = \{p^a \mid a \in A\}$ denotes the **transition model**. p^a is an $n \times n$ stochastic matrix, where $p^a(s, \dot{s})$ is the probability of transitioning to state $\dot{s}$ if the agents choose joint action a in state s;*
- *$o = \{o^{a,z} \mid a \in A, z \in Z\}$ is the **observation model**. $o^{a,z}$ is an $n \times 1$ vector[1], where $o^{a,z}(\dot{s})$ is the probability of observing z if joint action a is performed and the resulting state is $\dot{s}$;*
- *$r = \{r^a \mid a \in A\}$ is the **reward function**; r^a is a $1 \times n$ reward vector, where $r^a(s)$ is the bounded reward obtained by executing joint action a in state s;*
- *b_0 is the initial probability distribution over states; and*
- *T is the number of decision **steps** $t \in \{0, 1, \dots, T-1\}$ (the problem horizon).*

Remark 1. *We often use shorthand notation $p^{a,z}(s, \dot{s}) \overset{\text{def}}{=} o^{a,z}(\dot{s}) p^a(s, \dot{s})$, for all $s, \dot{s} \in S$, $a \in A$, and $z \in Z$, combining the transition and observation models. That is, the probability of transitioning to state $\dot{s}$ after observing z if joint action a is performed and the resulting state is s.*

1. The observation vector is not a stochastic vector.

To make the representation more concrete, we discuss a very simple Dec-POMDP, namely the multi-agent tiger problem (Nair, Tambe, Yokoo, Pynadath, & Marsella, 2003). We will revisit this problem in later sections to clarify the ideas presented in the paper.

Example 1 (Multi-agent tiger — problem description)**.** *In the multi-agent tiger problem, two agents stand in front of two closed doors. Behind one of the doors there is a hungry tiger, and behind the other door there is valuable treasure. The agents do not know the position of either. By listening, rather than simply opening one of the doors, the agents can gain information about the position of the tiger. But listening has a cost and is not entirely accurate (*i.e., *it only reveals the correct information about the location of the tiger with some probability). Moreover, agents cannot communicate their observations to each other. At each step, each agent can independently either listen or open one of the doors. If one of the agents opens the door with the treasure behind it (while the other agent listens or also opens the correct door), they both get the reward. If either agent opens the door with the tiger, a large penalty is incurred. However, if they both open the tiger door at the same time, they receive a smaller penalty. The agents must make decisions about listening and opening doors based on the local observations. After a door is opened and the agents receive a reward or penalty, the problem starts over again and the tiger is randomly repositioned.*

We refer to the state of the multi-agent tiger world when the tiger is on the left as s_{TL} (tiger left) and when it is on the right as s_{TR} (tiger right). The actions for each agent are a_{OL} (open left), a_{OR} (open right), and a_{L} (listen). There are only two possible observations for each agent (even after opening a door): to hear the tiger on the left z_{HL} (hear left) or to hear the tiger on the right z_{HR} (hear right). The reward function is defined as shown on Table 1.

actions of both agents	listens	opens good door	opens bad door
listens	-2	$+9$	-101
opens good door	$+9$	$+20$	-100
opens bad door	-101	-100	-50

Table 1: Reward function definition for the multi-agent tiger problem

The transition and observation models can be described in detail as follows. The joint action $(a_{\text{L}}, a_{\text{L}})$ does not change the state of the world. Any other joint action causes a transition to state s_{TL} with probability 0.5 *and to state s_{TR} with probability* 0.5 *— essentially resetting the problem. When the world is in state s_{TL}, the joint action $(a_{\text{L}}, a_{\text{L}})$ results in observation z_{HL} for either agent with probability* 0.85 *and observation z_{HR} with probability* 0.15*; conversely for state s_{TR}. No matter what state the world is in, the other joint actions result in either observation with probability* 0.5.

This example illustrates a small Dec-POMDP. Even in this small Dec-POMDP, coordination is difficult due to uncertainty about the location of the tiger and the actions of the other agent.

2.2 Preliminaries

Given a T-step Dec-POMDP M, we would like agents to act in such a way as to maximize some common measure of long-term return in M. The challenge in Dec-POMDPs is that each agent's strategy typically must take the other agents' strategies into account. To this end, we discuss agent and team decision rules and policies that allow the agents to act based on local information, while attempting to maximize a joint objective.

2.2.1 Private Decision Rules and Policies

At every time step, each agent chooses an action to be executed based on the actions the agent has previously executed and the observations that it has received. This is called a *policy*. To better understand this concept, we introduce the notions of private *histories* and *decision rules*.

Definition 2. *A step-t* ***private history*** *of agent $i \in I$ is a length-t sequence of actions and observations, $\theta^i_t \overset{\text{def}}{=} (a^i_0, z^i_1, \ldots, z^i_{t-1}, a^i_{t-1}, z^i_t)$, where a^i_t and z^i_t denote actions and observations of agent $i \in I$ at time step $t \in \{0, 1, \ldots, T-1\}$.*

Any step-t private history θ^i_t follows the recursion $\theta^i_t = (\theta^i_{t-1}, a^i_{t-1}, z^i_t)$ and the initial private history θ^i_0 is empty. Let Θ^i_t be the set of all step-t private histories of agent $i \in I$, namely the step-t private history set. A private policy specifies private decision rules an agent can use at all time steps, one private decision rule for each time step.

Definition 3. *A step-t* ***private decision rule*** *$d^i_t \colon \Theta^i_t \mapsto A^i$ prescribes agent $i \in I$ the private action to be executed in each private history $\theta^i_t \in \Theta^i_t$ at a specified time step $t \in \{0, 1, \ldots, T-1\}$.*

We further denote D^i_t to be the set of all step-t private decision rules for agent $i \in I$ at time step $t \in \{0, 1, \ldots, T-1\}$, namely the step-$t$ *private decision rule set* of agent $i \in I$. Decision rules may be *randomized* or *deterministic*. In Dec-POMDPs, as in MDPs, there always exists a deterministic decision rule that is as good as any randomized decision rule (Oliehoek et al., 2008; Puterman, 1994). For this reason, we focus on deterministic (private) decision rules. Hence, private policies provide each agent with a mapping for action selection for any possible private history.

Definition 4. *A $(t'+1)$-step* ***private policy*** *$\pi^i_{t:t+t'} \overset{\text{def}}{=} (d^i_t, \ldots, d^i_{t+t'})$ is a sequence of private decision rules for agent $i \in I$ from time step t to time step $t+t'$, where $t \in \{0, 1, \ldots, T-1\}$ and $t' \in \mathbb{N}$.*

Example 2 (Multi-agent tiger — private decision rule and policy descriptions)**.** *Figure 2 shows a pair of 2-step private policies $\pi^1_{0:1}$ and $\pi^2_{0:1}$ as trees, for agent 1 and 2, respectively.*

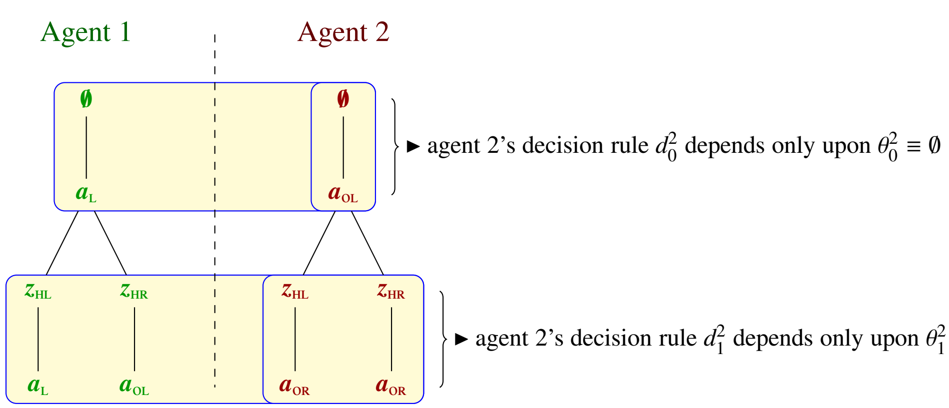

Figure 2: A pair of private policies in the form of trees and decomposed as decision rules for two steps of the multi-agent tiger problem.

For agent 2 *for example, step-*2 *private policy* $\pi^2_{0:1}$ *consists of two private decision rules* d^2_0 *and* d^2_1*, for time steps* $t = 0$ *and* $t = 1$*, respectively. The step-*0 *private decision rule* d^2_0 *maps empty private history* $\theta^2_0 = \emptyset$ *to private action* a_{L}*. In addition, the step-1 private decision rule* d^2_1 *maps private histories* $(a_{\text{OL}}, z_{\text{HL}})$ *and* $(a_{\text{OL}}, z_{\text{HR}})$ *to private action* a_{OR} *in both cases. It is worth noticing that private decision rules only maintain private histories that are reachable from past actions executed and observations received.*

So far, we focused on the information each agent has at the execution phase including: private histories, decision rules and policies. Nevertheless, the goal of Dec-POMDP planning is to find a separable joint policy. For this reason, we will next discuss joint histories, decision rules and policies.

2.2.2 Separable Joint Decision Rules and Policies

In this section, we extend private information available to a given agent, *e.g.,* private histories, decision rules and policies, to joint information that consists of a collection of private data. Let *joint histories* θ_t, *separable joint decision rules* d_t and *separable joint policies* $\pi_{t:t'}$ be $|I|$-tuples of private histories $(\theta^1_t, \theta^2_t, \ldots, \theta^{|I|}_t)$, decision rules $(d^1_t, d^2_t, \ldots, d^{|I|}_t)$ and policies $(\pi^1_{t:t'}, \pi^2_{t:t'}, \ldots, \pi^{|I|}_{t:t'})$, respectively for all time steps $t \in \{0, 1, \ldots, T-1\}$ and $t' \in \mathbb{N}$. Note that each of these concepts is *separable* in the sense that it is expressed as $|I|$-tuple using only private information, one private concept for each agent.

Example 3 (Multi-agent tiger — separable joint decision rule and joint policy descriptions)**.** *Figure 3 depicts a* 2*-step separable joint policy* $\pi_{0:1} \equiv (\pi^1_{0:1}, \pi^2_{0:1})$ *as a tuple of private policies, one for each agent* $i \in \{1, 2\}$*. If we group together private decision rules of agents at a given time step, then we have a separable joint decision rule. For example, at the initial time step* $t = 0$*, the tuple of private decision rules* $d_0 \equiv (d^1_0, d^2_0)$ *is a separable joint decision rule. In addition, separable joint decision rule* d_0 *prescribes to agents* 1 *and* 2 *actions* a_{L} *and* a_{OL}*, respectively.*

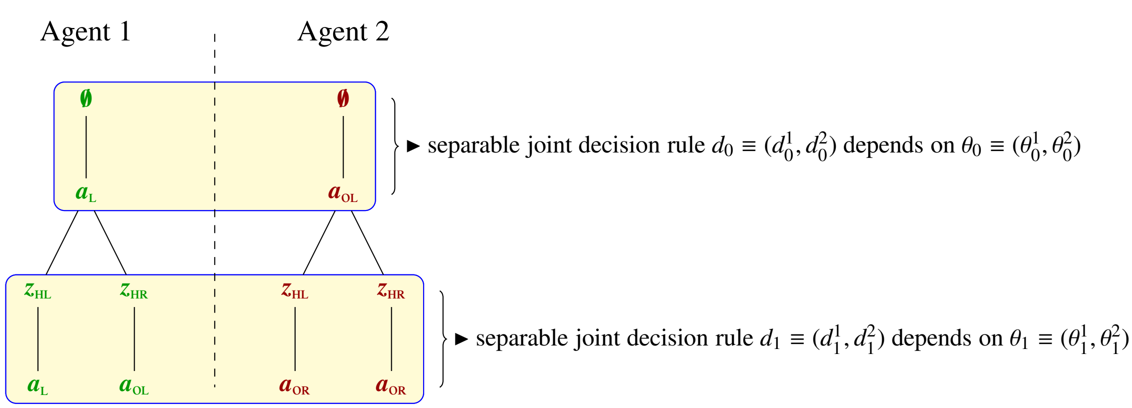

Figure 3: A 2-step separable joint policy for the multi-agent tiger problem.

2.3 Acting Optimally

In this section, we discuss the criterion used throughout the paper to compute an optimal separable joint policy starting at the initial belief-state. Before proceeding any further, we first cast Dec-POMDPs into an MDP, namely the information-state MDP.

2.3.1 INFORMATION-STATE MARKOV DECISION PROCESSES

As mentioned above, a common assumption in solving Dec-POMDPs is that planning takes place in a centralized (offline) manner even though agents execute actions in a decentralized fashion (online). In such a planning paradigm, a centralized algorithm maintains, at each step, the total available information about the process to be controlled. This centralized algorithm essentially performs a policy search in the space of separable joint policies. Thus, the separable joint decision rule choices are based only on the exhaustive information available to the centralized algorithm or on statistics derived from that information. This is illustrated in the influence diagram in Figure 4, where the separable joint decision rule at time step t depends only on previous separable joint decision rules and initial belief state, not on hidden states. The *statistics* summarizing the exhaustive information available to the centralized algorithm are called *information states* (Hauskrecht, 2000). As defined below, *complete information states* represent a trivial case, *i.e.,* the exhaustive data.

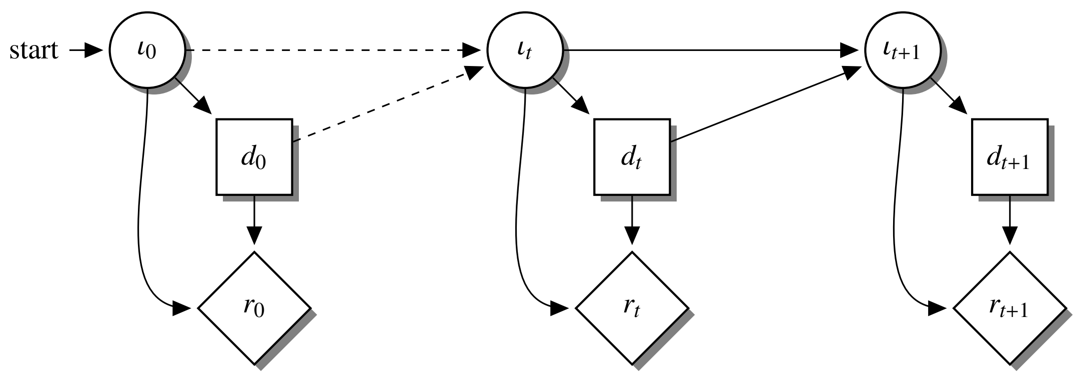

Figure 4: Influence diagram for an information-state MDP. Information states (ι_t and ι_{t+1}) are represented by cycles. Joint-decision-rule choices (d_t and d_{t+1}) are represented by rectangles, and depend only on the current information state, not on the underlying hidden states. Diamonds represent expected immediate rewards $r_0, \ldots, r_t$ and r_{t+1}. Dashed lines represent indirect influence over time.

Definition 5. *In Dec-POMDPs, a step-t* ***complete information state*** $\iota_t^C \overset{\text{def}}{=} (b_0, \pi_{0:t-1})$ *is a length-t sequence of separable joint decision rules starting with the initial belief state* b_0*, for all time steps* $t \in \{0, 1, \ldots, T-1\}$*. It satisfies the recursion:* $\iota_0^C = (b_0)$*, and* $\iota_{t+1}^C = (\iota_t^C, d_t)$*.*

Example 4 (Multi-agent tiger — complete information state description)**.** *Figure 5 depicts a step-2 complete information state as a sequence of separable joint decision rules* $\iota_2 \equiv (b_0, d_0, d_1)$ *starting at initial belief state* b_0*. Alternatively, the complete information state consists of a separable joint policy represented as a separable joint policy tree together with the initial distribution* b_0*. It is worth noting that, in a complete information state, each private history of each agent occurs in more than one joint history. It is this interdependence between joint histories that makes Dec-POMDPs significantly different from centralized problems (e.g., MDPs and POMDPs) since policies must remain decentralized. This interdependence also explains why joint histories, which are sufficient for optimally planning in MDPs and POMDPs, are no longer sufficient for optimally planning in Dec-POMDPs. Instead, we rely on complete information states.*

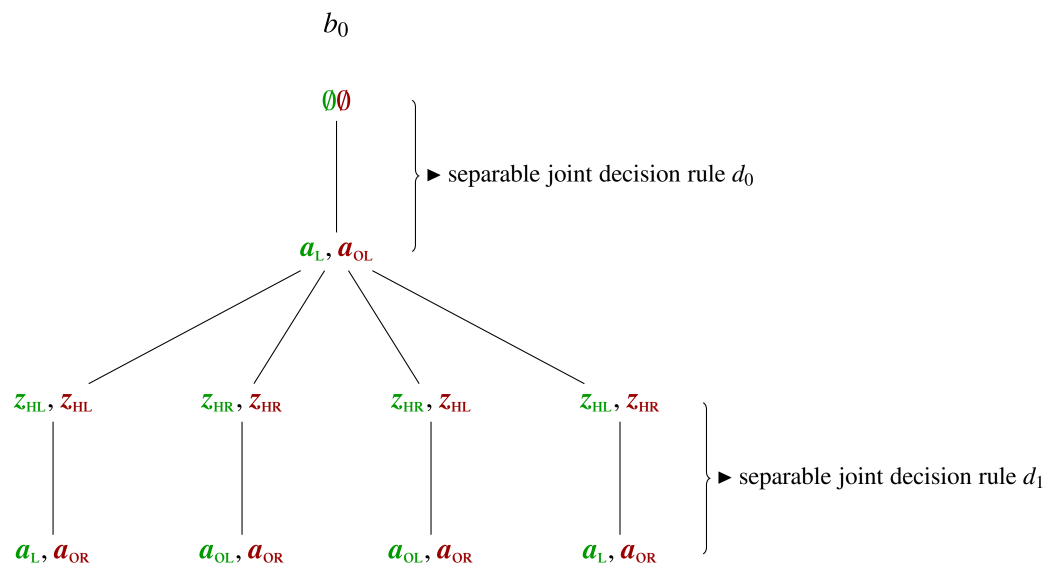

Figure 5: A step-2 complete information state $\iota_2^C \equiv (b_0, d_0, d_1)$ for the multi-agent tiger problem. Separable joint decision rules are groups of private decision rules depicted in Figure 3.

There is no need to retain the complete information states; instead, one can rely on more compact information states. Recall that information states are quantities summarizing the complete information states. The collection of random variables $\{\iota_t : t \in \{0, 1, \ldots, T\}\}$ taking values in the information state space $\boldsymbol{S}$ defines an *information-state* Markov decision process (ι-MDP). In such an MDP, "states" are information states and "actions" are separable joint decision rules as illustrated in the influence diagram in Figure 4. Our ι-MDP is deterministic since the next-step information state ι_{t+1} is a deterministic function of the previous information state ι_t and the joint-decision-rule choice d_t — *i.e.*, $\iota_{t+1} = \boldsymbol{P}(\iota_t, d_t)$. Furthermore, after taking a separable joint decision rule d_t at an information state ι_t, the expected reward is $\boldsymbol{R}(\iota_t, d_t)$.

Definition 6. *A ι-MDP $\hat{M} \equiv (\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{P}, \boldsymbol{R}, \iota_0, T)$ w.r.t. Dec-POMDP M is given by:*

- *$\boldsymbol{S}$ is the information-state set, which defines the set of all information states ι_t, at every time step $t \in \{0, 1, \ldots, T-1\}$;*
- *$\boldsymbol{A}$ is the set of separable joint decision rules, which defines the set of all separable joint decision rules d_t, at every time step $t \in \{0, 1, \ldots, T-1\}$;*
- *$\boldsymbol{P}$ specifies the next-step information state $\iota_{t+1} = \boldsymbol{P}(\iota_t, d_t)$ after taking separable joint decision rule d_t at information state ι_t, at every time step $t \in \{0, 1, \ldots, T-1\}$;*
- *$\boldsymbol{R}$ specifies the immediate expected reward $\boldsymbol{R}(\iota_t, d_t) = \sum_{s,\theta} Pr(s, \theta | \iota_t) \cdot r^{d_t(\theta)}(s)$ to be gained by executing a separable joint decision rule d_t at information state ι_t, at every time step $t \in \{0, 1, \ldots, T-1\}$;*
- *ι_0 is the initial information state; and T is the problem's temporal horizon.*

The information-state MDP $\hat{M}$ differs from the original Dec-POMDP M because the "state space" is implicit. That is, the information-state space $\boldsymbol{S}$ is much too large to be generated and stored in memory. Instead, information states are generated as they are explored during the state space search, and typically discarded thereafter. Generating the separable joint decision rules and their corresponding expected rewards are also sources of the complexity for current methods as discussed later. All these methods build upon the assumption that one can always convert the original Dec-POMDP into an information-state MDP by using *complete information states* without losing optimality (Szer et al., 2005; Oliehoek et al., 2008, 2013). Below, we provide a formal proof of this property for the sake of completeness.

Lemma 1. *Any optimal separable joint policy $\pi^*_{\hat{M}}$ for complete-information-state MDP $\hat{M}$ is an optimal separable joint policy π^*_M for the original Dec-POMDP M.*

Proof. In demonstrating the proof, we need to show that optimal joint policies $\pi^*_{\hat{M}}$ and π^*_M are separable joint policies with the same expected value. Throughout the proof, we will use notations $\boldsymbol{A}^T$ to denote T-steps separable joint policies and $P^{b_0,\pi}(\cdot)$ to denote a joint probability distribution with parameter b_0 and π. The proof starts with the definition of an optimal separable joint policy for information-state MDP $\hat{M}$:

$$\pi^*_{\hat{M}} \stackrel{\text{def}}{=} \arg\max_{\pi \in \boldsymbol{A}^T} \sum_{t=0}^{T-1} \boldsymbol{R}(\iota^C_t, d_t).$$

Next, we replace $\boldsymbol{R}(\iota^C_t, d_t)$ by the immediate expectation of rewards received after taking joint (separable) decision rule d_t at complete information state ι^C_t:

$$\pi^*_{\hat{M}} = \arg\max_{\pi \in \boldsymbol{A}^T} \sum_{t=0}^{T-1} \boldsymbol{E}_{(s_t,\theta_t)\sim P^{b_0,\pi_{0:t}}(\cdot)} \left\{ r^{d_t(\theta_t)}(s_t) \right\}, \qquad \text{(Def. of } \boldsymbol{R}(\iota^C_t, d_t)\text{)}.$$

The following holds because the sum of expectations is equal to expectation of sums:

$$\pi^*_{\hat{M}} = \arg\max_{\pi \in \boldsymbol{A}^T} \boldsymbol{E}_{(s_0,a_0,\ldots,s_{T-1},a_{T-1})\sim P^{b_0,\pi}(\cdot)} \left\{ \sum_{t=0}^{T-1} r^{a_t}(s_t) \right\} \stackrel{\text{def}}{=} \pi^*_M.$$

Hence, it is sufficient to search for an optimal separable joint policy using $\hat{M}$ to find an optimal separable joint policy for M (and vice versa). □

This lemma allows us to interchangeably use either complete-information-state MDPs $\hat{M}$ or the original Dec-POMDP counterpart M with no loss in optimality.

2.3.2 Optimality Criterion

In this paper, we consider the finite-horizon Dec-POMDP (and therefore the finite-horizon ι-MDP $\hat{M}$), where the optimality criterion is to find a separable joint policy that maximizes the expected sum of rewards over the planning horizon starting at a given belief state. To find an optimal separable joint policy, we first characterize the expected value to be gained from executing any arbitrary separable joint policy $\pi_{t:T-1}$ starting from any arbitrary step-t information state. This characterization represents the Dec-POMDP value function using the ι-MDP notation.

Definition 7. *Let $\pi_{t:T-1}$ be a separable joint policy with respect to $\hat{M}$. The* ***value function*** $V_{\hat{M},\pi_{t:T-1}}$ *denotes the expected cumulative reward obtained if the team of agents executes $\pi_{t:T-1}$ from time step t onward. For any arbitrary information state ι_t, $V_{\hat{M},\pi_{t:T-1}}(\iota_t) \overset{\text{def}}{=} \sum_{k=0}^{T-t-1} \boldsymbol{R}(\iota_{t+k}, d_{t+k})$, where $\iota_{t+k+1} = (\iota_t, d_t, \ldots, d_{t+k})$, $\forall t \in \{0, 1, \ldots, T-1\}$ and $\forall k \in \{0, 1, \ldots, T-t-1\}$.*

Example 5 (Multi-agent tiger — expected values given a separable joint policy and an information state)**.** *Figure 6 depicts a mapping from step-1 private histories to private policies $\pi^1_{1:T-1}$ and $\pi^2_{1:T-1}$ for agents 1 and 2, respectively. These private histories result from agents taking one action and receiving an observation, i.e., agent 1 took action a_{L} and received either observation z_{HL} or z_{HR}. The mapping ensures decentralized control since private histories map to private policies. For example, agent 1's private history $(a_{\text{L}}, z_{\text{HL}})$ maps to private policy $\bar{x}$. However, the expected value of one agent's private policy depends on the other agents' private policies. For this reason, we rely on joint histories induced by information state ι_1 as illustrated in Figure 7. Figure 7 depicts a mapping from joint histories to future separable joint policies. Each joint history is a pair of private histories from Figure 6, one for each agent. For example, joint history $\{(a_{\text{L}}, a_{\text{OL}}), (z_{\text{HL}}, z_{\text{HL}})\}$ maps to future separable joint policy (x, α) as a separable joint policy tree. The contribution of separable joint policy (x, α) to the expected value depends on the probability of joint history $\{(a_{\text{L}}, a_{\text{OL}}), (z_{\text{HL}}, z_{\text{HL}})\}$ and the initial belief.*

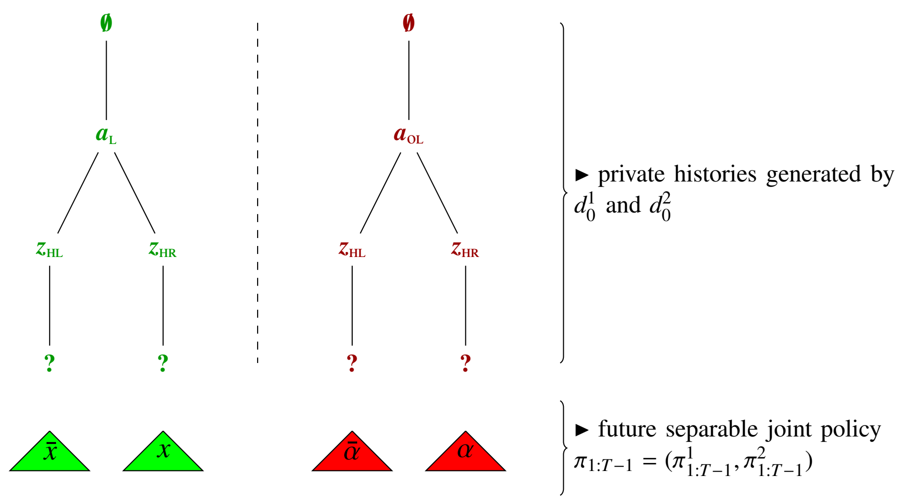

Figure 6: Mappings from private histories to future private policies for each agent.

Value function $V_{\hat{M},\pi_{t:T-1}}$ satisfies the following recursion: $V_{\hat{M},\pi_{t:T-1}}(\iota_t) = \boldsymbol{R}(\iota_t, d_t) + V_{\hat{M},\pi_{t+1:T-1}}(\boldsymbol{P}(\iota_t, d_t))$ where $V_{\hat{M},\pi_{t+1:T-1}}(\boldsymbol{P}(\iota_t, d_t))$ describes the future value of executing separable joint policy $\pi_{t+1:T-1}$ from time step $t+1$ onward starting at information state $\iota_{t+1} = \boldsymbol{P}(\iota_t, d_t)$.

2.3.3 Bellman's Optimality Equations

The standard definitions of optimality equations in a T-step ι-MDP follow. We first describe the optimal value at a given information state as the highest value of any separable joint policy for that information state. Let $\Pi_{t:T-1}$ be the set of all separable joint policies with respect to $\hat{M}$. For all $t \in \{0, 1, \ldots, T-1\}$, the optimal step-$t$ value function $V^*_{\hat{M},t}(\iota_t)$ at information state ι_t is $V^*_{\hat{M},t}(\iota_t) \overset{\text{def}}{=}$

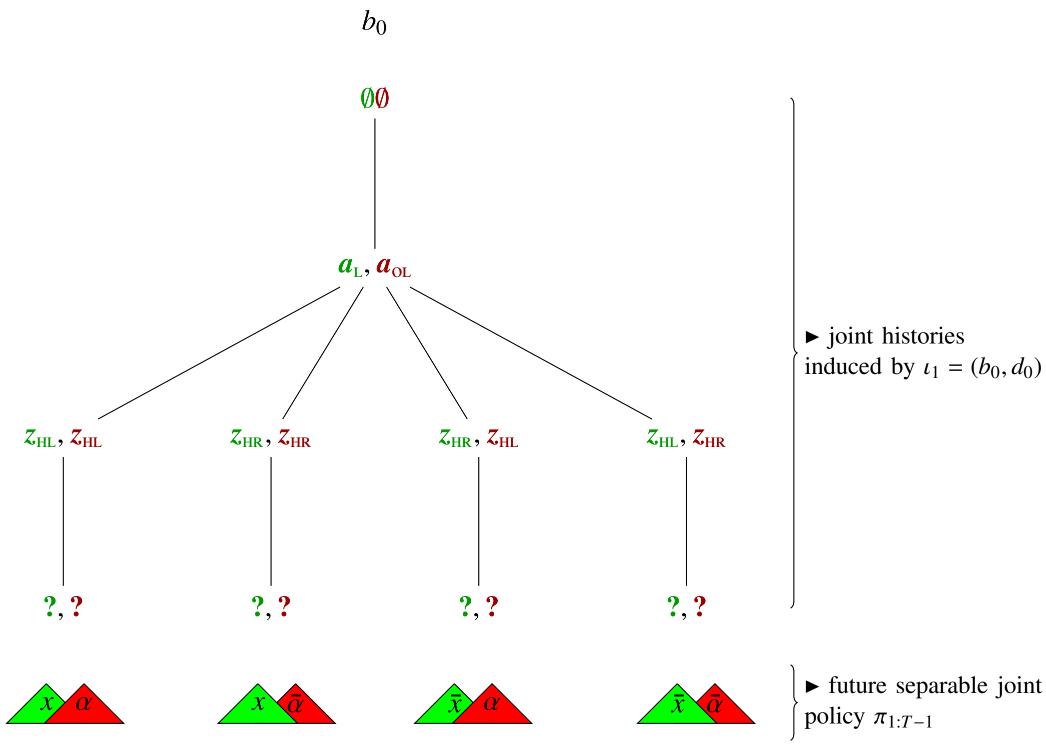

Figure 7: Mappings from joint histories to future separable joint policies.

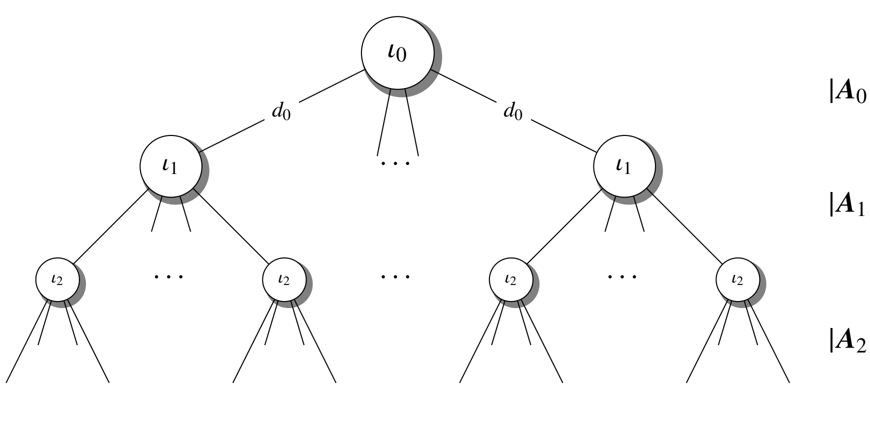

Figure 8: The information-state search tree where search nodes are information states and arcs of the search tree are labeled by separable joint decision rules.

$\max_{\pi \in \Pi_{t:T-1}} V_{\hat{M},\pi}(\iota_t)$. The **optimality equations** (or Bellman's optimality equations, see Bellman, 1957; Puterman, 1994) are:

$$V^*_{\hat{M},t}(\iota_t) \overset{\text{def}}{=} \max_{d_t \in \boldsymbol{A}_t} \Big(\boldsymbol{R}(\iota_t, d_t) + V^*_{\hat{M},t+1}(\boldsymbol{P}(\iota_t, d_t)) \Big), \qquad \forall \iota_t \in \boldsymbol{S}_t, \forall t \in \{0, 1, \ldots, T-1\}, \tag{1}$$

with an added boundary condition $V^*_{\hat{M},T}(\cdot) \overset{\text{def}}{=} 0$ for time step $t = T$.

Note that the optimal step-t value function is written in terms of the optimal step-$(t+1)$ value function. This recursion implies an efficient procedure for computing step-t value functions which we will discuss below. Moreover, an optimal separable joint policy can be directly extracted from the optimal value functions. Suppose $(V^*_{\hat{M},t})_{t \in \{0,1,\ldots,T-1\}}$ are solutions of the optimality equations (1) subject to the boundary condition, then it is clear that an **optimal separable joint policy** $\pi^*_{0:T-1} = (d^*_t)_{t \in \{0,1,\ldots,T-1\}}$ satisfies:

$$d^*_t \in \arg\max_{d_t \in \boldsymbol{A}_t} \Big(\boldsymbol{R}(\iota_t, d_t) + V^*_{\hat{M},t+1}(\boldsymbol{P}(\iota_t, d_t)) \Big), \qquad \forall t \in \{0, 1, \ldots, T-1\}. \tag{2}$$

This property implies that an optimal separable joint policy is found by first solving the optimality equations, and then for each time step choosing a separable joint decision rule that attains the maximum of the right hand side of (2) for $t \in \{0, 1, \ldots, T-1\}$.

2.4 Optimally Solving Dec-POMDPs

We provide an overview of dynamic programming and heuristic search principles exact methods build upon. For a thorough introduction to solution methods in Dec-POMDPs, the reader can refer to surveys (e.g., Oliehoek, 2012; Amato, Chowdhary, Geramifard, Ure, & Kochenderfer, 2013). Notice, however, that most dynamic programming methods do not explicitly consider information states (instead considering value functions over underlying system states). They construct separable joint policies from the last step in the horizon to the first by evaluating the possible separable joint policies at each step and pruning those that have provably lower value over the full state space (Hansen et al., 2004; Boularias & Chaib-draa, 2008; Amato et al., 2009). Heuristic search techniques implicitly use complete information states in developing Dec-POMDP solution methods (Szer et al., 2005; Oliehoek et al., 2008; Oliehoek, Whiteson, & Spaan, 2009; Spaan, Oliehoek, & Amato, 2011), but do not explicitly use the ι^C-MDP representation.

2.4.1 DYNAMIC PROGRAMMING METHODS

One class of Dec-POMDP solution methods is based on dynamic programming (Howard, 1960). Here, a set of T-step separable joint policies is generated from the bottom up (Hansen et al., 2004). At each step, all step-t separable joint policies are generated that build off separable joint policies from step $t+1$. Any separable joint policy that has lower value than some other separable joint policy for all states and possible separable joint policies of the other agents is then pruned (with linear programming). These generation and pruning steps continue until the desired horizon is reached and a separable joint policy with the highest value at the initial state is chosen. Given that the number of separable joint policies grows doubly exponentially every generation step, the importance of pruning away unnecessary separable joint policies is crucial. More efficient dynamic programming methods have been developed, reducing the number of separable joint policies generated (Amato et al., 2009) or compressing separable joint policy representations (Boularias & Chaib-draa, 2008).

2.4.2 Heuristic Search Methods

Another class of Dec-POMDP solution methods is based on heuristic search techniques. Unlike dynamic programming methods, heuristic search algorithms can take advantage of the initial complete information state. Separable joint policies can be built from the top down using centralized heuristic search methods over the search tree shown on Figure 8 (Szer et al., 2005).

In this case, a search node is a complete information state at a given horizon, t. These complete information states can be evaluated up to that horizon and then a heuristic value can be added. The resulting heuristic values are over-estimates of the true value, allowing an A*-style search through the space of possible complete information states, expanding promising search nodes to horizon $t + 1$ from horizon t. While in principle, A*-style search methods can find an optimal separable joint policy, in practice the doubly exponential growth of the search tree makes it difficult. Recent work has included clustering probabilistically equivalent complete information states (Boularias & Chaib-draa, 2008) and histories (Oliehoek et al., 2009), and incrementally expanding nodes in the search tree (Spaan et al., 2011; Oliehoek et al., 2013), greatly improving scalability of the original algorithms.

2.4.3 Limitations of Current Methods

While current methods attempt to reduce the number of separable joint policies or information states considered, they rely on explicit representations that consider all possible joint histories (even though many of them may be unreachable). Moreover, existing techniques fail to generalize value functions from one information state to other information states, which slows down the convergence to an optimal joint policy. Finally, even though most solution methods use an offline centralized planning phase, no concise representation of the information state has been identified (until now) that allows for greater scalability. Simultaneous to this work one exception developed concise representations based on observation histories, but did not show the value function over the resulting MDP was piecewise linear and convex (Oliehoek, 2013). We are able to show this piecewise linear and convex property and develop a novel algorithm to exploit the resulting structure. To do so, we draw inspiration from advances in MDP and POMDP algorithms as discussed below.

Significant progress has been made in solving large MDPs and POMDPs. One reason for progress in MDPs has been the use of approximate dynamic programming and function approximation (Tsitsiklis & van Roy, 1996; De Farias & Van Roy, 2003; Powell, 2007) to represent the state of the system and value functions more concisely. For POMDPs, efficient algorithms have been developed by recasting problems as belief MDPs that utilize probability distributions over states of the system, namely *belief states* (Smallwood & Sondik, 1973). This belief MDP is a continuous-state MDP with a piecewise linear and convex value function, allowing algorithms to scale to large problems while sometimes retaining performance bounds (Smith & Simmons, 2004; Shani et al., 2013). We will take advantage of such advances by recasting a Dec-POMDP as a continuous-state MDP with a piecewise linear and convex optimal value function. The resulting formulation opens the door for direct application of POMDP methods and opens research directions on utilizing Dec-POMDP structure in centralized planning representations. We discuss this formulation and some progress in using this structure in the remaining sections.

3. Solving Dec-POMDPs as Continuous-State MDPs

The contribution of this section is threefold. Section 3.1 introduces a statistic (*i.e.,* the occupancy state) which summarizes the information state. Section 3.1.4 demonstrates occupancy states are sufficient for optimally solving Dec-POMDPs, *i.e.,* occupancy states are *sufficient statistics*. Occupancy states further allow transforming information-state MDPs into occupancy-state MDPs. Section 3.1.6 establishes a fundamental property of the resulting MDP, namely the piecewise linearity and convexity of the optimal value function over the occupancy states. Remember that these methods assume centralized offline planning and actions as separable joint decision rules to ensure decentralized execution. These contributions enable the application of a vast collection of MDP and POMDP solution methods to Dec-POMDP problems. Finally, Section 3.2 introduces the occupancy-based heuristic search value iteration (OHSVI) algorithm for solving occupancy-state MDPs that builds upon the HSVI algorithm for POMDPs (Smith, 2007).

3.1 Summarizing Complete Information States

Before providing a formal definition of occupancy states, we start with a brief motivation. Then, we demonstrate that the occupancy state induces a deterministic process that is Markov, namely the occupancy-state Markov decision process. Finally, we prove that the occupancy state is a sufficient statistic for optimal decision-making in Dec-POMDPs.

3.1.1 OCCUPANCY STATE

As discussed in Section 2.4.2, standard heuristic search methods for solving Dec-POMDPs rely on complete information states (Szer et al., 2005; Oliehoek et al., 2008, 2009; Spaan et al., 2011). While complete information states preserve the ability to find an optimal separable joint policy (see Lemma 1), heuristic search methods using them can only solve small toy problems. One reason for this poor behavior is that complete information states result in redundant and useless computations. In particular, *every time* they estimate the immediate rewards $\boldsymbol{R}(\iota^C, d)$ the entire multivariate probability distribution $Pr(s, \theta|\iota^C)$ needs to be computed over all states and joint histories (see Definition 6). This operation is time-consuming because it involves exponentially many joint histories, including unreachable ones. Since this operation occurs at every time step, it is important to reduce the time required. To this end, we introduce a statistic called the *occupancy state* that we can maintain in place of the complete information state.

Definition 8. *The step-t* **occupancy state***, denoted ξ_t, is defined as the posterior probability distribution of state s_t and joint history θ_t given complete information state ι_t^C*, i.e., $\xi_t(s_t, \theta_t) \overset{\text{def}}{=} Pr(s_t, \theta_t|\iota_t^C)$, $\forall t \in \{0, 1, \ldots, T\}$. *We denote $\triangle_t$ the step-t occupancy simplex, that is, the set of all possible step-t occupancy states.*

Example 6 (Multi-agent tiger — from complete information states to occupancy states)**.** *Figure 9 depicts a complete information state (left-hand side) and a corresponding occupancy state (right-hand side) over joint histories and states of the system. We illustrate an occupancy state as a tree, where branches are joint histories of the complete information state and leaves are state-probability pairs. Note that the same initial belief is assumed in both state types.*

The occupancy state represents a predictive model of the state that the system may end up in and joint history the agents may experience given a complete information state. As such, in occupancy

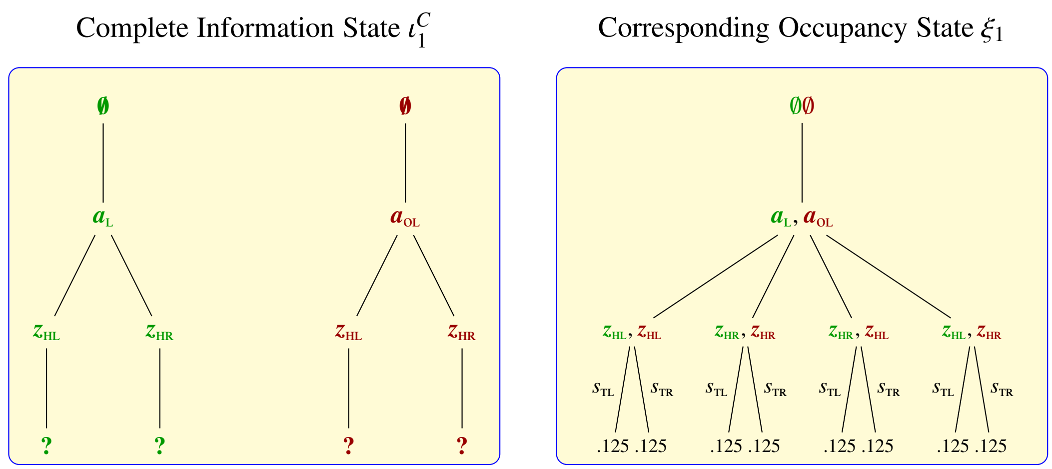

Figure 9: A step-1 occupancy state ξ_1 that corresponds to complete information state ι_1.

states, we need just to maintain state and joint history pairs that are reachable given the complete information state.

3.1.2 Markov Property

This section proves that occupancy states induce a process that is Markov. In other words, the future occupancy states of the process depend only upon the present occupancy state and the next-step separable joint decision rule.

Theorem 1. *Occupancy state ξ_{t+1} depends on current occupancy state ξ_t and separable joint decision rule d_t, i.e., for any arbitrary $\dot{s} \in S$, $a_t \in A$, $z_{t+1} \in Z$ and $\theta_t \in \Theta_t$,*

$$\xi_{t+1}(\dot{s}, (\theta_t, a_t, z_{t+1})) = \mathbf{1}_{\{a_t\}}(d_t(\theta_t)) \sum_{s \in S} \xi_t(s, \theta_t) \cdot p^{a_t, z_{t+1}}(s, \dot{s}), \tag{3}$$

where $\mathbf{1}_{\{\cdot\}}(\cdot)$ is an indicator function which returns 1 when the actions a_t are chosen by d_t, and returns 0 otherwise.

Proof. In demonstrating this theorem we also derive a procedure for updating the occupancy states. Let ι_t be our step-t information state, which we will decompose as $\iota_t = (\iota_{t-1}, d_{t-1})$ —i.e., the information state ι_{t-1} prior to time-step t plus the known separable joint decision rule d_{t-1}. By Definition 8, we can relate the occupancy state and the information state as follows: for any arbitrary state s_t and joint history θ_t,

$$\xi_t(s_t, \theta_t) \overset{\text{def}}{=} Pr(s_t, \theta_t | \iota_t). \tag{4}$$

The substitution of $\iota_t = (\iota_{t-1}, d_t)$ into (4) yields

$$\xi_t(s_t, \theta_t) = Pr(s_t, \theta_t | \iota_{t-1}, d_{t-1}). \tag{5}$$

The expansion of the right-hand side of (5) over all states of the system at the end of time-step $t-1$ produces

$$\xi_t(s_t, \theta_t) = \sum_{s_{t-1} \in S} Pr(s_{t-1}, s_t, \theta_t | \iota_{t-1}, d_{t-1}). \tag{6}$$

The expansion of the joint probability in (6) as the product of conditional probabilities results in

$$\xi_t(s_t, \theta_t) = \sum_{s_{t-1} \in S} Pr(a_{t-1}|\theta_{t-1}, d_{t-1}) \cdot Pr(s_t, z_t|s_{t-1}, \theta_{t-1}, \iota_{t-1}, d_{t-1}) \cdot Pr(s_{t-1}, \theta_{t-1}|\iota_{t-1}, d_{t-1}). \quad (7)$$

The first factor denotes the joint action a_{t-1} that separable joint decision rule d_{t-1} prescribes at θ_{t-1}. Since we assume that separable joint decision rules are deterministic, $Pr(a_{t-1}|\theta_{t-1}, d_{t-1}) \in \{0, 1\}$. In fact, $Pr(a_{t-1}|\theta_{t-1}, d_{t-1}) = 1$ if $d_{t-1}(\theta_{t-1}) = a_{t-1}$, otherwise $Pr(a_{t-1}|\theta_{t-1}, d_{t-1}) = 0$. So, $Pr(a_{t-1}|\theta_{t-1}, d_{t-1}) = \mathbf{1}_{\{a_{t-1}\}}(d_{t-1}(\theta_{t-1}))$, where $\mathbf{1}_F$ is an indicator function.

The second factor on the right-hand side of (7) is the transition probability

$$\xi_t(s_t, \theta_t) = \mathbf{1}_{\{a_{t-1}\}}(d_{t-1}(\theta_{t-1})) \sum_{s_{t-1} \in S} p^{a_{t-1}, z_t}(s_{t-1}, s_t) \cdot Pr(s_{t-1}, \theta_{t-1}|\iota_{t-1}, d_{t-1}). \quad (8)$$

The last factor defines the prior occupancy state ξ_{t-1} at state s_{t-1} and joint history θ_{t-1}, which does not depend on the current separable joint decision rule d_{t-1}. Overall (6) becomes

$$\xi_t(s_t, \theta_t) = \mathbf{1}_{\{a_{t-1}\}}(d_{t-1}(\theta_{t-1})) \sum_{s_{t-1} \in S} p^{a_{t-1}, z_t}(s_{t-1}, s_t) \cdot \xi_{t-1}(s_{t-1}, \theta_{t-1}).$$

Therefore, the calculation of the occupancy state after time-step t requires only the occupancy state of the previous time-step $t - 1$ and the current separable joint decision rule. □

Equation (3) describes the transitions of a continuous-state MDP in which states are occupancy states and actions are separable joint decision rules. For this process, the transitions are deterministic but the state space is continuous. Next, we formally define the process occupancy states induce.

3.1.3 Occupancy-State Markov Decision Processes

We consider the MDP described by the occupancy states; we call it an *occupancy-state Markov decision process*.

Definition 9. *Let $\check{M} \equiv (\triangle, \boldsymbol{A}, \boldsymbol{R}, \boldsymbol{P}, b_0, T)$ be the* ***occupancy-state Markov decision process*** *with respect to Dec-POMDP M, where:*

- $\triangle = \cup_{t \in \{0,1,\ldots,T\}} \triangle_t$ *is the occupancy simplex, where $\xi_0 = b_0$ is the initial occupancy state;*
- $\boldsymbol{A} = \cup_{t \in \{0,1,\ldots,T\}} \boldsymbol{A}_t$ *is the separable joint decision rule set;*
- $\boldsymbol{R} \colon \triangle \times \boldsymbol{A} \mapsto \mathbb{R}$ *is a reward function: the reward at (ξ_t, d_t) is $\boldsymbol{R}(\xi_t, d_t) \stackrel{\text{def}}{=} \sum_{s,\theta} \xi_t(s, \theta) \cdot r^{d_t(\theta)}(s)$;*
- $\boldsymbol{P} \colon \triangle \times \boldsymbol{A} \mapsto \triangle$ *is a transition function: next occupancy state $\xi_{t+1} \stackrel{\text{def}}{=} \boldsymbol{P}(\xi_t, d_t)$ as described in Equation (3) given (ξ_t, d_t);*
- *b_0 is the initial belief state; and*
- *T denotes the planning horizon.*

Here, the states of the system represent the centralized knowledge of the planner while the actions represent separable joint decision rules to ensure decentralized execution. The (occupancy) state can be updated by using the known transition and observation functions of the Dec-POMDP given the current (occupancy) state and the chosen separable joint decision rule. The rewards are also calculated (as an expectation) using the known reward model of the Dec-POMDP. The optimal value functions of $\check{M}$ are solutions of the optimality equations:

$$V^*_{\check{M},t}(\xi_t) \overset{\text{def}}{=} \max_{d_t \in \boldsymbol{A}} \Big(\boldsymbol{R}(\xi_t, d_t) + V^*_{\check{M},t+1}(\boldsymbol{P}(\xi_t, d_t)) \Big), \quad \forall t \in \{0, 1, \cdots, T-1\}, \tag{9}$$

with an added boundary condition $V^*_{\check{M},T}(\cdot) \overset{\text{def}}{=} 0$ for $t = T$.

Notice that once a solution $(V^*_{\check{M},t})_{t\in\{0,1,\ldots,T-1\}}$ of the optimality equations Eq. (9) has been found, one can always retrieve an optimal separable joint policy starting at the initial occupancy state. This is achieved by iteratively retrieving optimal separable joint decision rules for decision steps $t \in \{0, 1, \ldots, T-1\}$. At each decision step t, the procedure selects the current occupancy state ξ_t (starting with initial occupancy state ξ_0). It uses the max operator to retrieve an optimal separable joint decision rule d^*_t at current occupancy state ξ_t:

$$d^*_t \overset{\text{def}}{=} \arg\max_{d_t \in \boldsymbol{A}} \Big(\boldsymbol{R}(\xi_t, d_t) + V^*_{\check{M},t+1}(\boldsymbol{P}(\xi_t, d_t)) \Big). \tag{10}$$

Thereafter, it moves to the next occupancy state $\xi_{t+1} = \boldsymbol{P}(\xi_t, d^*_t)$ and makes it the current one. The procedure then repeats until final decision epoch T has been reached. The sequence of optimal separable joint decision rules $(d^*_0, d^*_1, \ldots, d^*_{T-1})$ defines an optimal separable joint policy π^* for an occupancy-state MDP $\check{M}$. Furthermore, Theorem 2 proves that an optimal separable joint policy for an occupancy-state MDP $\check{M}$ is an optimal separable joint policy for the original Dec-POMDP M.

Optimally solving a continuous-state MDP, such as the occupancy-state MDP, is a nontrivial task. In general, there is no exact solution method for solving general continuous-state MDPs. Methods often rely on structural assumptions about the shape of the optimal value function (Tsitsiklis & van Roy, 1996; De Farias & Van Roy, 2003; Powell, 2007). We next demonstrate that a useful structure does indeed exist for occupancy MDPs in the form of optimal value functions that are piecewise linear and convex functions over the occupancy states.

3.1.4 SUFFICIENCY OF OCCUPANCY STATES

We first show that the occupancy state is a sufficient statistic for optimal decision-making in Dec-POMDPs. Throughout the remainder of this paper, we call a statistic a *sufficient statistic* when the statistic of the information state is sufficient for optimal decision making in occupancy-state MDPs.

Theorem 2. *Occupancy state $\xi_t = Pr(s, \theta_t | \iota^C_t)_{s\in S, \theta_t \in \Theta_t}$ is a **sufficient statistic** of complete information state ι^C_t, i.e., it is sufficient for optimally solving occupancy-state MDPs. Furthermore, an optimal joint policy for the occupancy-state MDP $\check{M}$ together with the correct estimation of the occupancy states, is also optimal for information-state MDP $\hat{M}$ (respectively Dec-POMDP M).*

Proof. In demonstrating the sufficiency of the occupancy state with respect to its corresponding information state, we need to demonstrate that (a) the optimal value function at an occupancy state is identical to that of its corresponding information state and (b) the future occupancy states depend only upon the current occupancy states (and next-step separable joint decision rule). We proved (b) in Theorem 1, so it only remains to prove statement (a). We show this by induction.

The sufficiency of the occupancy state with respect to its corresponding information state trivially holds at the last time step of the problem. In fact, $V^*_{\hat{M},T}(\iota^C_T) = V^*_{\check{M},T}(\xi_T) = 0$ for any arbitrary complete information state ι^C_T and its corresponding occupancy state ξ_T (since the horizon has been reached).

If we assume that statement (a) holds for time-step $t+1$, we can now show it holds for time-step t. For any arbitrary step-t information state, Bellman's optimality criterion prescribes the following:

$$V^*_{\hat{M},t}(\iota^C_t) = \max_{d_t \in \boldsymbol{A}} \boldsymbol{R}(\iota_t, d_t) + V^*_{\hat{M},t+1}(\iota^C_{t+1}), \tag{11}$$

where $\iota^C_{t+1} = (\iota^C_t, d_t)$. By the induction hypothesis, we have $V^*_{\hat{M},t+1}(\iota^C_{t+1}) = V^*_{\check{M},t+1}(\xi_{t+1})$, if ξ_{t+1} corresponds to the occupancy state associated to ι^C_{t+1}. Hence, Equation (11) becomes

$$V^*_{\hat{M},t}(\iota^C_t) = \max_{d_t \in \boldsymbol{A}} \boldsymbol{R}(\iota^C_t, d_t) + V^*_{\check{M},t+1}(\xi_{t+1}). \tag{12}$$

Moreover, given that $\boldsymbol{R}(\iota^C_t, d_t) = \sum_{s,\theta} r^{d_t(\theta)}(s) \cdot Pr(s, \theta | \iota^C_t) = \boldsymbol{R}(\xi_t, d_t)$, the following expression holds since $\xi_t = (Pr(s, \theta | \iota^C_t))_{s,\theta}$:

$$V^*_{\hat{M},t}(\iota^C_t) = \max_{d_t \in \boldsymbol{A}} \boldsymbol{R}(\xi_t, d_t) + V^*_{\check{M},t+1}(\xi_{t+1}), \tag{13}$$

which ends the proof of statement (a) at time-step t, since $V^*_{\check{M},t}(\xi_t) = \max_{d_t \in \boldsymbol{A}} \boldsymbol{R}(\xi_t, d_t) + V^*_{\check{M},t+1}(\xi_{t+1})$. As a consequence, statement (a) holds for any arbitrary complete information state $\iota^C_t \in \boldsymbol{S}_t$ and at any arbitrary time-step $t \in \{0, 1, \ldots, T-1\}$. Combining statements ($a$) and ($b$), we are guaranteed to find the optimal value function for $\hat{M}$ by using occupancy states instead of information states.

As such, an optimal joint policy for the occupancy-state MDP $\check{M}$, together with the correct estimation of the occupancy states, is also optimal for information-state MDP $\hat{M}$ (or the original Dec-POMDP M). Given the optimal value function of $\check{M}$, an optimal joint policy $\pi^* = (d^*_t)_{t \in \{0,1,\ldots,T-1\}}$ is given by:

$$d^*_t = \arg\max_{d_t \in \boldsymbol{A}} \boldsymbol{R}(\xi_t, d_t) + V^*_{\check{M},t+1}(\xi_{t+1}), \tag{14}$$

for any arbitrary information state ι_t, and all $t \in \{0, 1, \ldots, T-1\}$, where ξ_t and ξ_{t+1} correspond to ι^C_t and $\iota^C_{t+1} = (\iota^C_t, d_t)$, respectively. □

This theorem demonstrates that by optimally solving any of $M, \hat{M}$ or $\check{M}$, we are guaranteed to find an optimal separable joint policy to the others.

3.1.5 Belief States versus Occupancy States

Note that there is a similarity between the occupancy state in Dec-POMDPs and the belief state in POMDPs. Formally, a step-t belief state $b_t = (P(s|\theta_t, b_0))_{s \in S}$ is a probability distribution over states conditioned on step-t history θ_t. It is also a sufficient statistic for the total data available to the centralized agent (*i.e.,* the action-observation history) an algorithm can rely on to find an optimal solution in POMDPs. Similarly, an occupancy state is a sufficient statistic for the total data available to a centralized planner (*i.e.,* the history of separable joint decision rules) an algorithm can rely on to find an optimal separable joint policy in Dec-POMDPs. However, the occupancy state remains fundamentally different from the belief state. First, the belief state is not sufficient for optimal decision making in Dec-POMDPs (because it is not geared to ensure the separability of the joint policy). Second, a belief state defines a time-invariant statistic, *i.e.,* the dimension of belief states is bounded by the number of states. In contrast, the dimension of the occupancy states grows exponentially with the horizon. Also, unlike the belief state, the occupancy state is only a *plan time* sufficient statistic which is not used during execution time. Instead, agents still condition their actions on local action-observation histories in Dec-POMDPs. These differences make algorithmic and theoretic transfers from belief-state MDPs to occupancy-state MDPs nontrivial.

3.1.6 Piecewise-Linearity and Convexity Property

We now present one of the main results of this paper — the piecewise-linearity and convexity of the optimal value function of the occupancy-state MDP.

For this discussion, we use vector (resp. matrix) representation for operators $\boldsymbol{R}(\cdot, d_t)$ and $\boldsymbol{P}(\cdot, d_t)$. Vector $r^{d_t} = (r^{d_t(\theta)}(s))_{s,\theta}$ denotes the immediate reward of executing d_t starting from any state and joint history (*i.e.,* $\boldsymbol{R}(\xi_t, d_t)$ is the inner product of ξ_t and r^{d_t} for any occupancy state $\xi_t \in \triangle_t$). Moreover, operator $\boldsymbol{P}(\cdot, d_t)$ transforms any step-t occupancy state to a step-$(t+1)$ occupancy state, that is, $\boldsymbol{P}(\cdot, d_t)$ describes a transition matrix p^{d_t} such that $\boldsymbol{P}(\xi_t, d_t) = \xi_t p^{d_t}$ for every step-t occupancy state $\xi_t \in \triangle_t$. With these linear transformations as a background, the following holds.

Theorem 3. *The optimal value functions* $(V^*_{\check{M},t})_{t\in\{0,1,\cdots,T\}}$ *(solutions to Equations in (9)) are piecewise-linear and convex functions of the occupancy states. Hence, for all* $t \in \{0, 1, \cdots, T-1\}$*, there exists a finite set of length-*$n|\Theta_t|$ *vector values* Λ_t *such that, for any arbitrary occupancy state* $\xi_t \in \triangle_t$*, we have:*

$$V^*_{\check{M},t}(\xi_t) = \max_{\beta_t \in \Lambda_t} \langle \xi_t, \beta_t \rangle, \tag{15}$$

where $\langle \xi_t, \beta_t \rangle$ *denotes the inner product* $\sum_s \sum_\theta \xi_t(s,\theta) \cdot \beta_t(s,\theta)$.

Proof. We show that (15) holds by induction. Since $V^*_{\check{M},T}(\xi_T) = 0$ for all $\xi_T \in \triangle$ (since the horizon has been reached), we have that $V^*_{\check{M},T}(\xi_T) = \max_{\beta_T \in \Lambda_T} \langle \xi_T, \beta_T \rangle$, where $\beta_T(\cdot) = 0$ and $\Lambda_T = \{\beta_T\}$. Hence, the property holds for $k = T$. Assume that the property holds for $k \geq t+1$, that is, $V^*_{\check{M},k}(\xi_k) = \max_{\beta_k \in \Lambda_k} \langle \xi_k, \beta_k \rangle$ for $k \geq t+1$. Now we want to prove the property for $k = t$ —*i.e.,*

$$V^*_{\check{M},t}(\xi_t) = \max_{d_t \in \boldsymbol{A}} \left(\boldsymbol{R}(\xi_t, d_t) + V_{\check{M},t+1}(\boldsymbol{P}(\xi_t, d_t)) \right), \qquad \forall \xi_t \in \triangle_t.$$

Using linear transformations r^{d_t} and p^{d_t}, the following holds:

$$\begin{aligned}
V^*_{\check{M},t}(\xi_t) &\overset{\text{def}}{=} \max_{d_t \in \boldsymbol{A}} \left(\boldsymbol{R}(\xi_t, d_t) + V_{\check{M},t+1}(\boldsymbol{P}(\xi_t, d_t)) \right), \\
&= \max_{d_t \in \boldsymbol{A}} \left(\langle \xi_t, r^{d_t} \rangle + \max_{\beta_{t+1} \in \Lambda_{t+1}} \langle \boldsymbol{P}(\xi_t, d_t), \beta_{t+1} \rangle \right), && \text{(Inductive Hypothesis)} \\
&= \max_{d_t \in \boldsymbol{A}} \max_{\beta_{t+1} \in \Lambda_{t+1}} \left(\langle \xi_t, r^{d_t} \rangle + \left\langle \xi_t \cdot p^{d_t}, \beta_{t+1} \right\rangle \right), && \text{(Rearranging Terms)} \\
&= \max_{d_t \in \boldsymbol{A}} \max_{\beta_{t+1} \in \Lambda_{t+1}} \left(\langle \xi_t, r^{d_t} + \beta_{t+1} \cdot (p^{d_t})^\top \rangle \right).
\end{aligned}$$

Finally, if we let Λ_t be the set of all length-$n|\Theta_t|$ vectors $\beta_t \overset{\text{def}}{=} r^{d_t} + \beta_{t+1} \cdot (p^{d_t})^\top$ for all separable joint decision rules $d_t \in \boldsymbol{A}$ and all vectors $\beta_{t+1} \in \Lambda_{t+1}$, then $V^*_{\check{M},t}(\xi_t) = \max_{\beta_t \in \Lambda_t} \langle \xi_t, \beta_t \rangle$. As a consequence, the proof holds for every time step $t \in \{0, 1, \ldots, T-1\}$. ◻

We demonstrated that the information states and value functions can be represented in a vector space, the occupancy-state space, without loosing optimality. Next, we provide an approach for extending MDP and POMDP solution methods to occupancy-state Markov decision processes.

3.2 Heuristic Search Solution Methods

This section presents the occupancy-based heuristic search value iteration (OHSVI) algorithm for solving occupancy-state MDPs. This algorithm extends the heuristic search value iteration (HSVI) algorithm for POMDPs (Smith & Simmons, 2004) as well as other heuristic search algorithms such as A* (Hart, Nilsson, & Raphael, 1968) and LRTA* (Korf, 1990).

3.2.1 Heuristic Search Value Iteration for Occupancy-State MDPs

HSVI is a state-of-the-art algorithm for solving POMDPs (Smith & Simmons, 2004). It produces solutions by maintaining two-sided bounds on the optimal value function and updating them over a number of sample trajectories. The upper bounds, $(U_{\check{M},t})_{t\in\{0,1,\cdots,T\}}$, are represented as (belief) state-value mappings, and the lower bounds, $(L_{\check{M},t})_{t\in\{0,1,\cdots,T\}}$, are represented by vector sets. Each trajectory begins at the initial belief state and continues until the time horizon is reached. Once a trajectory is finished, the upper and lower bounds are updated at each belief state, in the reverse order of visit. The trajectories can also be interrupted once they have reached a belief state where the upper and lower bounds are equal, since there is no reason to expand a belief state whose optimal value is provably known. Finally, it is often useful to prune lower and upper bounds to maintain concise representations, by removing either dominated vectors or points, respectively (Pineau, Gordon, & Thrun, 2006; Smith, 2007).

Algorithm 1: The OHSVI algorithm

```
function OHSVI((L_{M̌,t})_{t∈{0,1,···,T}}, (U_{M̌,t})_{t∈{0,1,···,T}})
    while ¬STOP(ξ_0) do EXPLORE (ξ_0)

function STOP(ξ_t)
    return U_{M̌,t}(ξ_t) = L_{M̌,t}(ξ_t)

function EXPLORE (ξ_t)
    if ¬STOP(ξ_t) then
        d*_t ← arg max_{d_t} R(ξ_t, d_t) + U_{M̌,t+1}(P(ξ_t, d_t))
        EXPLORE(P(ξ_t, d*_t))
        update U_{M̌,t} and L_{M̌,t} at ξ_t
```

OHSVI (outlined in Algorithm 1) operates in a similar manner as described above, but remains fundamentally different from HSVI. HSVI generates trajectories of belief states while OHSVI generates trajectories of occupancy states. Also, HSVI generates trajectories by (i) picking a greedy action with respect to the upper bound (optimistic exploration), and (ii) performing the transition corresponding to the largest gap in error. Due to the deterministic nature of occupancy-state MDPs, OHSVI does not need HSVI's gap-based heuristic to guide the state exploration (Smith, 2007). Instead, OHSVI always executes a greedy separable joint decision rule with respect to the upper bounds, and then selects the next occupancy state based on this greedy separable joint decision rule.

OHSVI can be also thought of as an extension of *learning real-time A** (LRTA*) (Korf, 1990) that takes advantage of the piecewise-linearity and convexity of the optimal value function.

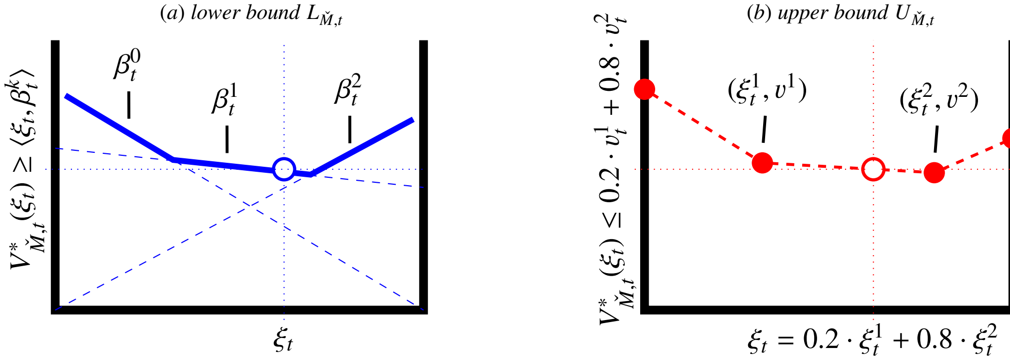

Figure 10: (a) Lower bounds are represented using sets of vectors, where vectors are dashed lines, solid lines represent the upper surface of these vectors (lower-bound value function), and the circle is the projection of the target occupancy onto the lower-bound value function. (b) Upper bounds are represented using occupancy-value mappings, where dashed lines denotes the convex hull formed by these points.

OHSVI relies on standard approaches to represent lower and upper bounds for piece-wise linear and convex value functions: *vector sets* and *occupancy-value mappings*, which we detail in the next section.

3.2.2 Vector Sets : Lower Bounds

As in HSVI or other algorithms (and depicted in Figure 10(a)), the lower bound $L_{\check{M},t}$ can be represented as a finite collection Λ_t of $n|\Theta_t|$-dimensional vectors, for every time-step $t \in \{0, 1, \ldots, T\}$ (Smith, 2007; Kaelbling et al., 1998; Hauskrecht, 2000; Smallwood & Sondik, 1973). Lower bounds can be iteratively updated using point-based backup steps as follows: $\forall \xi_t \in \triangle_t$,

$$\Lambda'_t = \Lambda_t \bigcup \{backup(\Lambda_{t+1}, \xi_t)\}, \text{ where} \tag{16}$$

$$backup(\Lambda_{t+1}, \xi_t) = \arg\max_{g^{\xi_t}_{d_t}:\, d_t \in \boldsymbol{A}} \langle \xi_t, g^{\xi_t}_{d_t} \rangle, \tag{17}$$

$$g^{\xi_t}_{d_t} = r^{d_t} + \arg\max_{g^{\beta_{t+1}}_{d_t}:\, \beta_{t+1} \in \Lambda_{t+1}} \langle \xi_t, g^{\beta_{t+1}}_{d_t} \rangle, \qquad \text{(sum vectors in } \mathbb{R}^{n|\Theta_t|}) \tag{18}$$

$$g^{\beta_{t+1}}_{d_t} = \beta_{t+1} \cdot (p^{d_t})^\top, \qquad \text{(projection } \mathbb{R}^{n|\Theta_{t+1}|} \mapsto \mathbb{R}^{n|\Theta_t|}) \tag{19}$$

Λ_t being the vector set prior to backup and Λ'_t the vector set after the backup. Lower bounds $(L_{\check{M},t})_{t\in\{0,1,\ldots,T-1\}}$ initializes $(\Lambda_t)_{t\in\{0,1,\ldots,T-1\}}$ with a single vector $\beta_t(\cdot) = \min_{s\in S, a\in A} (T-t) \cdot r^a(s)$, for all $t \in \{0, 1, \ldots, T\}$. Notice that the vector representation is suitable only for lower bounds.

3.2.3 Occupancy-Value Mappings : Upper Bounds

Upper bounds $(U_{\check{M},t})_{t\in\{0,1,\ldots,T-1\}}$ can be represented using mappings from occupancy states to reals, see e.g., Figure 10(b). The upper bound is then the convex hull of the current point set. It is possible to interpolate the value for occupancy states whose mapping is not currently maintained or is outdated. This can be achieved using linear approximation methods (e.g., Hauskrecht, 2000; Smith, 2007). Of this family, the sawtooth linear interpolation maps every occupancy state $\xi' \in \triangle_t$ and point set Ψ_t to upper-bound value

$$U_{\check{M},t}(\xi') = \min \{v^*_{\xi'}, v^{\xi}_{\xi'} \mid (\xi \mapsto v^{\xi}) \in \Psi_t\}, \text{ where} \tag{20}$$

$$v^{\xi}_{\xi'} = v^*_{\xi'} + (v^{\xi} - v^*_{\xi}) \cdot D(\xi, \xi'), \text{ and} \tag{21}$$

$$v^*_{\xi} = \sum_{s,\theta} \xi(s,\theta) \cdot v^{s,*}. \tag{22}$$

We refer to $D(\xi, \xi') = \min_{s,\theta:\, \xi(s,\theta)>0}\ \xi'(s,\theta)/\xi(s,\theta)$ as the sawtooth measure. To update the upper bound $U_{\check{M},t}$ at a specific occupancy state ξ using sawtooth, we need to compute a new value for ξ, and add it to occupancy-value mapping $U_{\check{M},t}$, as follows:

$$\Psi'_t = \Psi_t \bigcup \left\{(\xi \mapsto v^{\xi})\right\}, \tag{23}$$

$$v^{\xi} = \max_{d \in \boldsymbol{A}_t} \boldsymbol{R}(\xi, d) + U_{\check{M},t+1}(\boldsymbol{P}(\xi, d)), \tag{24}$$

where Ψ_t is the point set prior update and Ψ'_t is the point set after the update. The upper bound $U_{\check{M},t}$ initializes $\Psi_t = \{\xi^{s,*} \mapsto v^{s,*}_t \mid s \in S\}$ using the optimal value of the underlying MDP for corner points, for every $t \in \{0, 1, \ldots, T\}$.

Clearly, one can eventually find an optimal solution to the occupancy-state MDP using OHSVI with the full lower and upper bounds. However, it quickly becomes intractable to maintain these bounds in the full occupancy-state space since the number of state and joint-history pairs grows as the horizon increases. This highlights the necessity for compact representations of occupancy states, decision rules and vector values.

4. Solving Dec-POMDPs as Lossless Compact Occupancy-State MDPs

The previous sections show that every Dec-POMDP can be represented as an occupancy-state MDP without losing optimality. The difficulty with using this representation is that common algorithms for solving such MDPs with piecewise linear convex value functions quickly run out of time and/or memory since state and action spaces become intractably large for most real-world problems. This is not surprising given the NEXP-Complete worst-case complexity of general Dec-POMDPs, but realistic Dec-POMDP applications often have significant structure.

In this section, we discuss optimally solving occupancy-state MDPs while potentially reducing the dimensionality of the occupancy states, decision rules and value functions. In Subsection 4.1, we reduce the dimensionality of occupancy states and decision rules by constructing clusters of equivalent histories. Next, we define compact representations for occupancy states and decision rules based upon clusters of histories rather than single histories. While the resulting compact MDP may have exponentially fewer states and actions than the original model, the optimal value function of the compact model may no longer be piecewise linear and convex. In Subsection 4.2, we overcome this limitation, allowing values from one compact occupancy state to generalize to another one using parametric value functions. Finally, Subsection 4.3 presents the *feature-based heuristic search value iteration* algorithm and theoretical guarantees.

4.1 Lossless Compact Occupancy-State MDPs

The dimension of occupancy states and decision rules typically grows exponentially with the horizon. Because of this, it is often impractical to compute and store every component of occupancy state and decision rule representations. We overcome this limitation by using compact representations of occupancy states and decision rules based on notions of equivalence between histories. These notions of equivalence are fundamental to designing and analyzing algorithms for reducing the dimensionality of the occupancy-state MDP, thereby improving scalability. Equivalence relations permit us to aggregate histories that convey the same information about the process. We target equivalence relations that, upon replacing each group of aggregated histories by one element of the group, allow us to produce compact representations for all occupancy states and decision rules while still preserving the ability to find an optimal solution.

4.1.1 Probabilistic Equivalence for History Space Aggregation

We first present the equivalence notions that we build upon before defining compact representations and proving that they preserve optimality. The definitions here are inspired by work on concise information states (Boularias & Chaib-draa, 2008; Oliehoek et al., 2013). Here, we connect this research to occupancy-state MDPs, and later provide natural algorithms for constructing and effectively solving them.

Definition 10. *Private histories $\theta^i, \theta'^i \in \Theta^i_t$ of agent $i \in I$ are* ***locally probabilistically equivalent*** *with respect to occupancy state $\xi \in \triangle_t$ —denoted ξ-LPE— if, and only if, for any state $s \in S$ and history $\theta^{\neg i} \in \Theta^{\neg i}_t$ of the other agents $I \backslash \{i\}$: $Pr(s, \theta^{\neg i} | \theta^i, \xi) = Pr(s, \theta^{\neg i} | \theta'^i, \xi)$.*

It is worth noticing that ξ-LPE can be used to partition private history set Θ^i_t of any agent $i \in I$, for all time steps $t \in \{0, 1, \ldots, T\}$. This partition is a set of nonempty subsets (called clusters) $B^i_1, B^i_2, \ldots, B^i_k$ of private histories, such that $B^i_1 \cup B^i_2 \cup \ldots \cup B^i_k = \Theta^i_t$. We distinguish between two sets of private histories for each agent $i \in I$ and any occupancy state ξ. The first set, denoted $\Theta^i_t(\xi)$, consists of private histories with non-zero probability with respect to ξ. The second set, denoted $\Theta^i_t \backslash \Theta^i_t(\xi)$, consists of private histories with zero probability with respect to ξ. This difference is particularly important, as we will show later. In fact, only nonzero private histories play a part in demonstrating that ξ-LPE preserves optimality. In addition, it is useful to note that, for each agent, ξ-LPE groups together all zero private histories w.r.t. ξ into the same cluster.

Given that all private histories are clustered that convey the same information, representations of compact occupancy states, decision rules and value functions should depend only upon these clusters. Unfortunately, maintaining clusters still requires a large amount of memory, which explains the impetus for *labeled clusters*. A labeled cluster is a cluster along with a label; all private histories in a cluster match the corresponding label. Throughout the paper, we use the following convention: *each cluster of private histories maps to the minimum private history (of that cluster) using lexicographical ordering*. Specifically, the label of a cluster is chosen among private histories in the cluster, which have non-zero probability w.r.t. the occupancy state. Therefore, the label of a cluster is also a private history in that cluster, which leads to a clear relationship between a private history, a cluster and its corresponding label. Thus, the representation of compact occupancy states, decision rules and value functions should depend only upon labels instead of clusters.

Nonetheless, these compact representations make it hard to generalize the value function from one compact occupancy state to another. As we will see later, this generalization requires the ability to *quickly* check whether a given private history belongs to a specified cluster. If we group histories using ξ-LPE, checking whether private history θ^i belongs to cluster B^i whose label is another private history $\vartheta^i \in B^i$ can now be replaced by checking whether private histories θ^i and ϑ^i are ξ-LPE, for any arbitrary agent $i \in I$. Unfortunately, checking whether two private histories of an agent are ξ-LPE requires enumerating all states and other agents' nonzero histories w.r.t. ξ, which results in a subroutine with complexity $O(n|\Theta_t^{-i}(\xi)|)$ — see Algorithm 3 in Appendix B. Given that we call this procedure for exponentially many private histories, the importance of replacing local probabilistic equivalence by a cheaper equivalence relation is clear. To this end, we introduce *truncation probabilistic equivalence* w.r.t. ξ (denoted ξ-TPE). Before providing the formal definition of ξ-TPE, we start with a motivation.

In many practical situations, all important information about the process to be controlled lies in a history window (of a fixed length) of actions and observations the agents have experienced. Based on this insight, we want truncation probabilistic equivalence to group private histories of each agent that share: (*i*) the same model about states and other agent histories (*i.e.,* be ξ-LPE) and (*ii*) a common suffix of a fixed length m. In such a setting, once the private histories have been clustered, checking if any particular private history belongs to a cluster can now be replaced by checking whether both that private history and the label (another private history) for the cluster share the same suffix of a specified length m — which is significantly faster than checking whether two private histories are ξ-LPE: $O(m)$ versus $O(n|\Theta_t^{-i}(\xi)|)$. That is, once the private histories are clustered, any two private histories with the same suffixes of length m are already known to be ξ-LPE though the clustering process and choice of m. It turns out that in defining the probabilistic truncation equivalence w.r.t. ξ, we need to determine the suffix length (*i.e.,* history window) m_ξ that is sufficient — called the *local truncation parameter* w.r.t. ξ.

The **local truncation parameter** m_ξ with respect to occupancy state $\xi \in \triangle_t$ has to be: (*i*) large enough to ensure all nonzero private histories w.r.t. ξ that share the same suffix of length m_ξ are also ξ-LPE; and (*ii*) small enough to group the maximum number of private histories. A straightforward method (Algorithm 4 in Appendix B) to compute m_ξ starts with parameter $m = 0$ and proceeds as follows: (step 1) If for any agent, nonzero private histories w.r.t. ξ that share a common suffix of length m are also ξ-LPE with one another, then set $m_\xi = m$ and terminate; (step 2) Otherwise, increment $m = m + 1$ and go back to step 1. This algorithm is guaranteed to terminate after at most t (the current time step) iterations, *i.e.,* $m_\xi = t$ in the worst case. The later case corresponds to the boundary case where no clustering is done — *i.e.,* each cluster corresponds to a single joint history. We are now ready to formally define the truncation probabilistic equivalence relation.

Definition 11. *Private histories $\theta^i, \theta'^i \in \Theta_t^i$ of agent $i \in I$ are* ***truncation probabilistically equivalent*** *with respect to occupancy state $\xi \in \triangle_t$ — we denote ξ-TPE — if, and only if, they hold the same last m_ξ private actions and observations given m_ξ is found for the set of histories as discussed above.*

Not surprisingly, ξ-TPE also partitions the private history sets. However, it often produces more clusters than ξ-LPE since it further constrains ξ-LPE non-zero private histories w.r.t. ξ (since it also requires the suffixes to be the same). In contrast to ξ-LPE, it spreads zero private histories w.r.t. ξ over different clusters.

Recall that the advantage of ξ-TPE over ξ-LPE is that finding the appropriate cluster for a private history (*i.e.,* checking whether two private histories are equivalent) is cheaper using ξ-TPE than using ξ-LPE. The former requires the comparison of their suffix for a fixed length, whereas the latter needs the comparison of large distributions over states and joint histories.

It is worth noticing that, to the best of our knowledge, TPE and LPE are the weakest assumptions to date that can reduce the dimensionality of full occupancy states. Nevertheless, not all Dec-POMDPs will benefit from these data reduction approaches. More precisely, it is unlikely that these assumptions provide concise occupancy states in totally unstructured domains. Fortunately, real-world domains are often very structured, as demonstrated in Section 5. Next, we address the problem of using equivalence relations between private histories to find compact representations for all occupancy states, decision rules and real vectors.

4.1.2 Compact States, Actions, Vectors and MDPs

We define the compact representations for occupancy states, decision rules and real vectors based upon the aforementioned equivalence relations. Notice that the following definitions depend on either *local* or *truncation probabilistic equivalence* relations only through the labeled clusters they generate. Intuitively, compact occupancy states are distributions over states and joint labels, compact decision rules are mappings from labels to private actions, and compact real vectors are mappings from pairs of states and joint labels to reals. Notationally, let L^i_ξ be the label set of agent $i \in I$ that occupancy state ξ generates (*i.e.,* the set of labels generated from the histories which make up ξ), and B_{ϑ^i} be the labeled cluster of agent $i \in I$ with label $\vartheta^i \in L^i_\xi$ and let $L_\xi = \bigtimes_{i\in I} L^i_\xi$.

Definition 12. *A* ***compact occupancy state*** *of ξ, denoted $\tilde{\xi}$, is a distribution over states and joint labels: for all $s \in S$ and $(\vartheta^i)_{i\in I} \in L_\xi$,*

$$\tilde{\xi}(s, \vartheta^1, \vartheta^2, \ldots, \vartheta^{|I|}) \stackrel{\text{def}}{=} \sum_{\theta^1 \in B_{\vartheta^1}} \sum_{\theta^2 \in B_{\vartheta^2}} \cdots \sum_{\theta^{|I|} \in B_{\vartheta^{|I|}}} \xi(s, \theta^1, \theta^2, \theta^{|I|}). \tag{25}$$

Example 7 (Multi-agent tiger — from full occupancy states to compact occupancy states)**.** *Figure 11 depicts a full occupancy state (left-hand side) and a corresponding compact occupancy state (right-hand side) over joint histories and hidden states. To obtain this compact occupancy state, we show that $(\boldsymbol{a}_{\text{L}}, z_{\text{HL}})$ and $(\boldsymbol{a}_{\text{L}}, z_{\text{HR}})$ are ξ_1-LPE for agent green, and $(\boldsymbol{a}_{\text{L}}, z_{\text{HL}})$ and $(\boldsymbol{a}_{\text{OL}}, z_{\text{HR}})$ are ξ_1-LPE for agent red. As a consequence, one can group histories $B^{green} \equiv \{(\boldsymbol{a}_{\text{L}}, z_{\text{HL}}), (\boldsymbol{a}_{\text{L}}, z_{\text{HR}})\}$ for agent green, and $B^{red} \equiv \{(\boldsymbol{a}_{\text{OL}}, z_{\text{HL}}), (\boldsymbol{a}_{\text{OL}}, z_{\text{HR}})\}$ for agent red, and replace each cluster by a single label.*

A ***compact decision rule*** *w.r.t. ξ of agent $i \in I$, denoted $\tilde{d}^i_\xi \colon L^i_\xi \mapsto A^i$, is a mapping from label set to private action set. In addition, a* ***compact private policy*** *w.r.t. $(\xi_0, \xi_1, \ldots, \xi_{T-1})$ of agent $i \in I$, denoted $\tilde{\pi}^i = (\tilde{d}^i_{\xi_t})_{t\in\{0,1,\ldots,T-1\}}$, is a sequence of compact decision rules of agent $i \in I$. A similar definition follows for* ***compact separable joint policies*** *$\tilde{\pi} = (\tilde{\pi}^i)_{i\in I}$.*

A ***compact real vector*** *w.r.t. ξ, denoted $\tilde{\beta}_\xi \in \mathbb{R}^{n|L_\xi|}$, is a mapping from pairs of state $s \in S$ and joint label $\vartheta \in L_\xi$ to reals.*

Intuitively, distribution $\tilde{\xi}$ reassigns the probability mass of each joint cluster $(B_{\vartheta^i})_{i\in I}$ to the corresponding joint label $(\vartheta^i)_{i\in I}$. Algorithms 3 and 4 (see Appendix B) present a straightforward way to compute a compact occupancy state using local and truncation probabilistic equivalence relations, respectively. It is worth noticing that, for a given occupancy state $\xi \in \triangle$, ξ-LPE always produces a number of joint labels $|L_\xi|$ less than or equal to the number of labels that ξ-TPE produces. This is because ξ-TPE is a stricter form of local probabilistic equivalence, as discussed above. Hence, ξ-LPE produces compact occupancy states, decision rules and real vectors that are more concise than those from ξ-TPE.

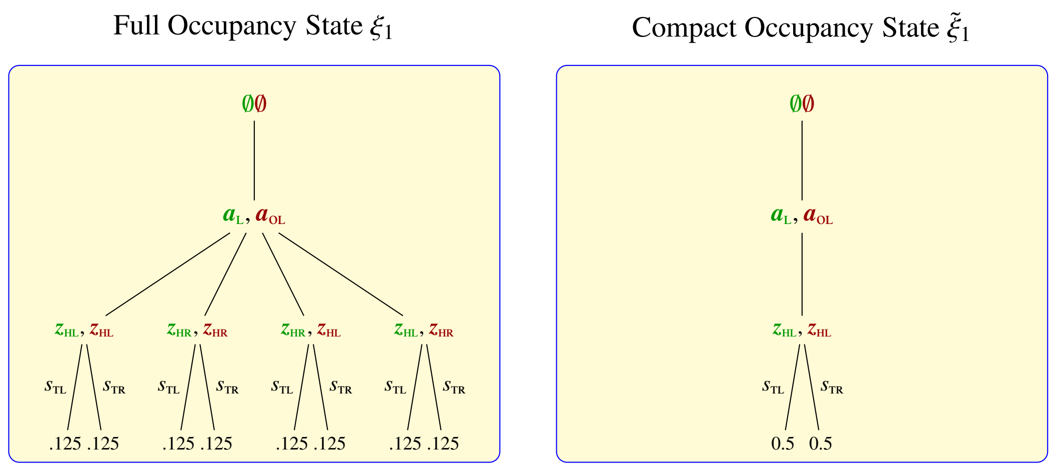

Figure 11: A step-1 compact occupancy state $\tilde{\xi}_1$ that corresponds to full occupancy state ξ_1.

In demonstrating that compact occupancy states are sufficient for optimal decision making in occupancy-state MDPs, we rely on the notion of *compact occupancy-state MDPs*.

Definition 13. *The* ***compact occupancy-state MDP*** *w.r.t. occupancy-state MDP $\check{M}$ is given by tuple $\tilde{M} \equiv (\tilde{\triangle}, \tilde{\boldsymbol{A}}, \tilde{\boldsymbol{R}}, \tilde{\boldsymbol{P}}, \tilde{\xi}_0, T)$:*

- $\tilde{\triangle} = \cup_{t\in\{0,1,\ldots,T\}} \tilde{\triangle}_t$ *is the space of compact occupancy states, with $\tilde{\xi}_0 = \xi_0$ the initial compact occupancy state;*
- $\tilde{\boldsymbol{A}} = \cup_{t\in\{0,1,\ldots,T\}} \tilde{\boldsymbol{A}}_t$ *is the space of compact joint (decentralized) decision rules;*
- $\tilde{\boldsymbol{R}} \colon \tilde{\triangle} \times \tilde{\boldsymbol{A}} \mapsto \mathbb{R}$ *is a compact reward function where $\tilde{\boldsymbol{R}}(\tilde{\xi}, \tilde{d}_{\xi}) = \sum_{s,\vartheta} \tilde{\xi}(s, \vartheta) \cdot r^{\tilde{d}_{\xi}(\vartheta)}(s)$;*
- $\tilde{\boldsymbol{P}} \colon \tilde{\triangle} \times \tilde{\boldsymbol{A}} \mapsto \tilde{\triangle}$ *is a compact transition function where $\tilde{\boldsymbol{P}}(\tilde{\xi}, \tilde{d}_{\xi}) \overset{\text{def}}{=} \tilde{\xi}'$, with $\xi' = \boldsymbol{P}(\tilde{\xi}, \tilde{d}_{\xi})$; and*
- *T denotes the planning horizon.*

Analogous to occupancy-state MDPs, compact occupancy states can be updated using known transition and observation functions of the Dec-POMDP given the current compact occupancy state and the chosen compact separable joint decision rule. The rewards are also calculated (as expectations) using the known reward model of the Dec-POMDP. Finally, the optimal value function of $\tilde{M}$ is solution of the optimality equations:

$$V^*_{\tilde{M},t}(\tilde{\xi}_t) \overset{\text{def}}{=} \max_{\tilde{d}_{\xi_t}\in\tilde{\boldsymbol{A}}} \left(\tilde{\boldsymbol{R}}(\tilde{\xi}_t, \tilde{d}_{\xi_t}) + V^*_{\tilde{M},t+1}(\tilde{\boldsymbol{P}}(\tilde{\xi}_t, \tilde{d}_{\xi_t}))\right), \qquad \forall t \in \{0, 1, \ldots, T-1\}, \tag{26}$$

with an added boundary condition $V^*_{\tilde{M},T}(\cdot) = 0$.

4.1.3 SUFFICIENCY OF COMPACT OCCUPANCY STATES

Below, we prove that, when planning over compact occupancy states instead of full occupancy states, the optimal separable joint policy for the compact model immediately induces a corresponding optimal separable joint policy for the original model. Before proceeding any further, we first demonstrate the optimality of compact policies.

Note that this proof mirrors a similar proof showing that histories for an agent can be clustered without any change to the optimal policy or loss in value by using probabilistic equivalence in the traditional Dec-POMDP representation (Oliehoek et al., 2013). We extend these ideas to the occupancy-state MDP case and incorporate truncation probabilistic equivalence here to show that the compact policies preserve optimality.

Theorem 4 (Optimality of compact policies)**.** *In occupancy-state MDPs, there exists optimal policies of an agent that depend only upon labels of that agent produced based on either local or truncation probabilistic equivalence with respect to occupancy states these optimal policies induce, and not on private histories.*

Proof. We construct the proof by induction. We first show that in the last step of the problem, an agent policy depends on labels, but not on its private histories.

Given occupancy state $\xi_{T-1} \in \triangle_{T-1}$ and policies $\pi^{\neg i}_{T-1} \in \Pi^{\neg i}_{T-1}$ of the other agents $I\backslash\{i\}$, the policy $\pi^i_{T-1} \in \Pi^i_{T-1}$ of agent $i \in I$ is a *best response* to $\pi^{\neg i}_{T-1}$. That is, it chooses for every private history $\theta^i_{T-1} \in \Theta^i_{T-1}$ a private action so as to maximize value based on model $(Pr(s, \theta^{\neg i}_{T-1}|\theta^i_{T-1}, \xi_{T-1}))_{s,\theta^{\neg i}_{T-1}}$ occupancy state ξ_{T-1} and private history θ^i_{T-1} induce over possible histories of the other agents and resulting states of the system:

$$\pi^i_{T-1}(\theta^i_{T-1}) \in \arg\max_{a^i \in A^i} \sum_{s,\theta^{\neg i}_{T-1}} Pr(s, \theta^{\neg i}_{T-1}|\theta^i_{T-1}, \xi_{T-1}) \cdot r^{a^i, \pi^{\neg i}_{T-1}(\theta^{\neg i}_{T-1})}(s). \tag{27}$$

In assigning private actions to private histories given ξ_{T-1}, only nonzero private histories w.r.t. ξ_{T-1} affect the outcome. That is, zero probability private histories w.r.t. ξ_{T-1} can be prescribed any private action without losing optimality (*e.g.,* a private action identical to that of the associated labels generated based on either local or truncation probabilistic equivalence relations). Hence, policy π^i_{T-1} depends on zero probability private histories w.r.t. ξ_{T-1} only through corresponding labels. Next, we restrict our attention to nonzero probability private histories w.r.t. ξ_{T-1}. Recall that private histories of this family that are ξ-TPE with one another are also ξ-LPE with one another. Therefore, if we demonstrate the property for private histories of this family that are ξ-LPE with one another, the proof follows immediately for those that are ξ-TPE.

Assume θ^i_{T-1} is a nonzero probability private history w.r.t. ξ_{T-1}. If θ^i_{T-1} is in the cluster of ξ_{T-1}-LPE private histories with label ϑ^i_{T-1} (another nonzero probability private history w.r.t. ξ_{T-1}), then equality $(Pr(s, \theta^{\neg i}_{T-1}|\theta^i_{T-1}, \xi_{T-1}))_{s,\theta^{\neg i}_{T-1}} = (Pr(s, \theta^{\neg i}_{T-1}|\vartheta^i_{T-1}, \xi_{T-1}))_{s,\theta^{\neg i}_{T-1}}$ holds. As a consequence, policy π^i_{T-1} depends on nonzero probability private history w.r.t. ξ_{T-1} only through corresponding labels:

$$\pi^i_{T-1}(\theta^i_{T-1}) \in \arg\max_{a^i \in A^i} \sum_{s,\theta^{\neg i}_{T-1}} Pr(s, \theta^{\neg i}_{T-1}|\vartheta^i_{T-1}, \xi_{T-1}) \cdot r^{a^i, \pi^{\neg i}_{T-1}(\theta^{\neg i}_{T-1})}(s). \tag{28}$$

Therefore, the property holds at the last step of the problem, for any arbitrary agent $i \in I$. This allows us to define policies on the last step as mappings from labels to private actions, *i.e.,* compact policies.

Next, we rely on the concept of private *policy tree*, which is a tree that represents actions in nodes and observations in edges with a depth that is the number of stages to go. The root node determines the first private action to be taken. Then depending on private observation received, the agent executes another private action; and so on until a leaf node is reached. A policy tree

$\delta^i_{t:T-1}$ is a portion of a private policy $\pi^i_{t:T-1}$, which prescribes private actions to be taken by agent $i \in I$ in the remaining stages starting at a given private history θ^i_t. By an abuse of the notation, let $\delta^i_{t:T-1} = \pi^i_{t:T-1}(\theta^i_t)$, for all agents $i \in I$ and all time steps $t \in \{0, 1, \ldots, T\}$.

For the induction step, we can show that, if an agent policy-tree from step $t+1$ onward depends on private histories only through the corresponding labels for either equivalence relations, then that agent's policy tree from step t onward also depends on its private histories only through the corresponding labels for either equivalence relations. Given occupancy state $\xi_t \in \triangle_t$ and policy $\pi^{\neg i}_{t:T-1} \in \Pi^{\neg i}_{t:T-1}$ of the other agents $I\backslash\{i\}$, policy $\pi^i_{t:T-1} \in \Pi^i_{t:T-1}$ of agent $i \in I$ is as follows:

$$\pi^i_{t:T-1}(\theta^i_t) \in \arg\max_{\delta^i_{t:T-1} \in \Pi_{t:T-1}} \sum_{s, \theta^{\neg i}_t} Pr(s, \theta^{\neg i}_t | \theta^i_t, \xi_t) \cdot \beta^{\delta^i_{t:T-1}, \pi^{\neg i}_{t:T-1}(\theta^{\neg i}_t)}(s), \tag{29}$$

where $(\delta^i_{t:T-1}, \pi^{\neg i}_{t:T-1}(\theta^{\neg i}_t))$ is a separable joint policy-tree and $\beta^{\delta^i_{t:T-1}, \pi^{\neg i}_{t:T-1}(\theta^{\neg i}_t)}$ is the associated vector value. Again, in assigning private actions to private histories given ξ_t, only nonzero probability private histories w.r.t. ξ_t matter. In fact, the property trivially holds for zero probability private histories w.r.t. ξ_t. For nonzero probability private histories w.r.t. ξ_t, the property holds for private histories that belong to a cluster of ξ_t-TPE private histories since those histories would belong to a cluster of ξ_t-LPE private histories, as previously discussed.

For this reason, we restrict our attention to nonzero probability private histories θ^i_t w.r.t. ξ_t that belong to a cluster of ξ_t-TPE private histories with label ϑ^i_t. Then, equality $(Pr(s, \theta^{\neg i}_t | \theta^i_t, \xi_t))_{s, \theta^{\neg i}_t} = (Pr(s, \theta^{\neg i}_t | \vartheta^i_t, \xi_t))_{s, \theta^{\neg i}_t}$ holds. Hence,

$$\pi^i_{t:T-1}(\theta^i_t) \in \arg\max_{\delta^i_{t:T-1} \in \Pi_{t:T-1}} \sum_{s, \theta^{\neg i}_t} Pr(s, \theta^{\neg i}_t | \vartheta^i_t, \xi_t) \cdot \beta^{\delta^i_{t:T-1}, \pi^{\neg i}_{t:T-1}(\theta^{\neg i}_t)}(s). \tag{30}$$

Consequently, an agent policy depends on private history for any step of the problem only through corresponding labels for either equivalence relations. This demonstrates that a compact policy does not lose information. □

Theorem 5. *Compact occupancy state $\tilde{\xi}$ based on either local or truncation probabilistic equivalence relations is a* ***sufficient statistic*** *of occupancy state $\xi \in \triangle_t$. Furthermore, an optimal separable joint policy for compact occupancy-state MDP $\tilde{M}$, together with the correct estimation of the compact occupancy states, immediately induces an optimal separable joint policy for occupancy-state MDP $\check{M}$.*

Proof. The proof proceeds similarly to that of Theorem 2. That is, in proving the sufficiency of the compact occupancy state with respect to its corresponding full occupancy state, we need to demonstrate: (*a*) the optimal value function at a compact occupancy state is identical to that of its corresponding occupancy state and (*b*) the next-step compact occupancy states depend only upon the current compact occupancy states (and next-step compact separable joint decision rules). We stated (*b*) in Definition 13, so only statement (*a*) remains to be proved. We show this by induction.

The sufficiency of the compact occupancy state with respect to its corresponding occupancy state trivially holds at the last step of the problem. In fact, $V^*_{\tilde{M},T}(\tilde{\xi}_T) = V^*_{\check{M},T}(\xi_T) = 0$ for any arbitrary occupancy state ξ_T and its corresponding compact occupancy state $\tilde{\xi}_T$ (since the horizon has been reached). If we assume the statement (*a*) holds for time-step $t+1$, we can now show that it holds for time-step t. For any arbitrary step-t occupancy state, Bellman's optimality criterion produces the following equality:

$$V^*_{\check{M},t}(\xi_t) \overset{\text{def}}{=} \max_{d_t \in \boldsymbol{A}} \boldsymbol{R}(\xi_t, d_t) + V^*_{\check{M},t+1}(\xi_{t+1}), \tag{31}$$

where $\xi_{t+1} = \boldsymbol{P}(\xi_t, d_t)$. By the inductive hypothesis, we have that $V^*_{\tilde{M},t+1}(\tilde{\xi}_{t+1}) = V^*_{\check{M},t+1}(\xi_{t+1})$. This results in equality:

$$V^*_{\check{M},t}(\xi_t) = \max_{d_t \in \boldsymbol{A}} \boldsymbol{R}(\xi_t, d_t) + V^*_{\tilde{M},t+1}(\tilde{\boldsymbol{P}}(\xi_t, d_t)). \tag{32}$$

In addition, Theorem 4 demonstrated that by restricting our attention to compact (joint) decision rules $\tilde{\boldsymbol{A}}$, we preserve optimality. Thus, we obtain:

$$\begin{aligned}
\boldsymbol{R}(\xi_t, d_t) &= \sum_{\phi_{\dot{s},\vartheta} \in \Phi_{\xi_t}} r^{d_t(\vartheta)}(\dot{s}) \sum_{s,\theta} \phi_{\dot{s},\vartheta}(s,\theta) \cdot \xi_t(s,\theta), && \text{(Theorem 4)}\\
&= \sum_{\phi_{\dot{s},\vartheta} \in \Phi_{\xi_t}} r^{\tilde{d}_{\xi_t}(\vartheta)}(\dot{s}) \cdot \tilde{\xi}_t(\dot{s},\vartheta), && \text{(Definition of } \tilde{\xi}_t)\\
&= \boldsymbol{R}(\tilde{\xi}_t, \tilde{d}_{\xi_t}), && (\tilde{d}_{\xi_t}(\vartheta) = d_t(\vartheta)).
\end{aligned}$$

Hence,

$$\begin{aligned}
V^*_{\check{M},t}(\xi_t) &= \max_{\tilde{d}_{\xi_t} \in \tilde{\boldsymbol{A}}} \tilde{\boldsymbol{R}}(\tilde{\xi}_t, \tilde{d}_{\xi_t}) + V^*_{\tilde{M},t+1}(\tilde{\boldsymbol{P}}(\tilde{\xi}_t, \tilde{d}_{\xi_t})), && (33)\\
&= V^*_{\tilde{M},t}(\tilde{\xi}_t). && (34)
\end{aligned}$$

Therefore, statement (a) holds for any arbitrary occupancy state $\xi_t \in \triangle_t$ and any arbitrary time step $t \in \{0, 1, \ldots, T-1\}$. Combining statements ($a$) and ($b$), we are guaranteed to find the optimal value function for $\check{M}$ by using compact occupancy states instead of occupancy states. In addition, given the optimal value function $(V^*_{\tilde{M},t})_{t\in\{0,1,\ldots,T\}}$, the optimal compact policy $\tilde{\pi} \overset{\text{def}}{=} (\tilde{d}_t)_{t\in\{0,1,\ldots,T-1\}}$ is obtained by successive one-step lookaheads: for all $t \in \{0, 1, \ldots, T-1\}$,

$$\tilde{d}_t \in \max_{\tilde{d}_{\xi_t} \in \tilde{\boldsymbol{A}}} \tilde{\boldsymbol{R}}(\tilde{\xi}_t, \tilde{d}_{\xi_t}) + V^*_{\tilde{M},t+1}(\tilde{\boldsymbol{P}}(\tilde{\xi}_t, \tilde{d}_{\xi_t})), \tag{35}$$

where $\tilde{\xi}_0 = \xi_0$ and $\tilde{\xi}_{t+1} = \tilde{\boldsymbol{P}}(\tilde{\xi}_t, \tilde{d}_t)$. This immediately induces a separable joint policy $\pi \overset{\text{def}}{=} (d_t)_{t\in\{0,1,\ldots,T-1\}}$ for the original occupancy-state MDP $\check{M}$ such that, for every agent $i \in I$ and every private history θ^i_t, we set $d^i_t(\theta^i_t) = \tilde{d}^i_t(\vartheta^i_t)$, where θ^i_t is in a cluster with label ϑ^i_t. Since both $\tilde{\pi}$ and π yield the same expected value starting at the initial occupancy state, separable joint policy π is optimal for the original occupancy-state MDP $\check{M}$. □

By solving the compact problem instead of the original one, we circumvent the exhaustive enumeration of occupancy states and decision rules and preserve ability to find an optimal solution for the original problem. It is worth noticing that the optimal value function in the compact occupancy space is not PWLC. This is because compact occupancy states are expressed using different label sets. However, by exploiting the PWLC property of the optimal value function in the full occupancy-state space, we develop methods that can generalize value from one compact occupancy state to another. Next, we propose a method for incrementally improving lower and upper bounds that narrow the range of the optimal value function. But, in contrast to the OHSVI algorithm, we do this using compact occupancy states and compact real vectors. More precisely, our compact upper and lower bounds are defined on full occupancy states, but only through their compact representations.

4.2 Feature-Based Compact Bounds

In our algorithm, upper and lower bounds are of crucial importance. They can narrow the range of the optimal value function, determine suboptimal regions of the search space, and speed up the convergence towards an exact solution. Our approach approximates the full lower- and upper-bound heuristic functions with heuristic functions defined over the same full occupancy space (Tsitsiklis & van Roy, 1996; Hauskrecht, 2000; Roy, Gordon, & Thrun, 2005). The new heuristic functions are typically more compact (with respect to traditional high-dimensional vector or point sets discussed in Section 3.2.2 and Section 3.2.3), and are easier to compute than the full bounds. Our heuristic functions can be formulated as feature-based compact functions which combine dimensionality reduction (using the clustering methods discussed above) and function approximation (Tsitsiklis & van Roy, 1996). Thus, to demonstrate that the new heuristic functions are valid bounds, we will analyze our dimension reduction and approximation methods.

4.2.1 Feature-Based Compact States and Vectors

One can think of feature-based compact representations as a dimension reduction model for representing high-dimensional bounds using bounds of lower dimensionality. However, this approach requires a set of basis functions (or feature set), in which lower-dimensional bounds can be expressed effectively. A basis function (or feature) is a function which maps high-dimensional data to one salient feature of the problem at hand. In our setting for example, a feature can be an indicator function of whether a private history matches one specified label. Features are at the core of the feature-based compact representations of full occupancy states and real vectors, which ultimately serve to represent upper and lower bounds using either feature-based compact vector or point sets in a similar way as standard vector and point set representations.

Definition 14. *A* ***feature*** *is an indicator function $\phi_{s,(\vartheta^i)_{i\in I}} : S \times \Theta \mapsto \{0,1\}$ for one specified state $s \in S$ and one specified joint label $(\vartheta^i)_{i\in I} \in L_\xi$ that occupancy state ξ induces — i.e., for all $\dot{s} \in S$ and $(\theta^i)_{i\in I} \in \Theta$,*

$$\phi_{s,(\vartheta^i)_{i\in I}}(\dot{s},(\theta^i)_{i\in I}) = \begin{cases} 1, & \textit{if } s = \dot{s} \textit{ and, for all } i \in I,\ \theta^i \textit{ belongs to cluster with label } \vartheta^i, \\ 0, & \textit{otherwise}. \end{cases}$$

The ***feature set*** *w.r.t. ξ, denoted Φ_ξ, is given by $\Phi_\xi \overset{\text{def}}{=} \{\phi_{s,\vartheta} | s \in S, \vartheta \in \times_{i\in I} L^i_\xi\}$.*

The feature set w.r.t. ξ represents the partition of the state and joint history space that equivalence relation ξ-LPE or ξ-TPE induces. This partition is a set of nonempty subsets $(B_{s,\vartheta})_{s\in S,\vartheta\in L_\xi}$ (called labeled joint clusters) such that $\cup_{s\in S,\vartheta\in L_\xi} B_{s,\vartheta} = S \times \Theta$. A labeled joint cluster $B_{s,\vartheta}$ consists of the cross-product between the singleton $\{s\}$ and labeled clusters $B_{\vartheta^1}, B_{\vartheta^2}, \ldots, B_{\vartheta^{|I|}}$ that equivalence relation ξ-LPE or ξ-TPE generates. Therefore, a feature can be interpreted as a way to check whether a state and joint history pair belongs to a specified labeled joint cluster. Thus, features provide an alternative (possibly lossy) representation of compact occupancy states. One can express the compact representation $\tilde{\xi}$ of full occupancy state ξ using its feature set Φ_ξ, as follows

$$\tilde{\xi} = \left(\sum_{s \in S} \sum_{\theta \in \Theta(\xi)} \phi_{\dot{s},\vartheta}(s,\theta) \cdot \xi(s,\theta) \right)_{\phi_{\dot{s},\vartheta} \in \Phi_\xi}.$$

Specifically, for all state $\dot{s} \in S$ and all labels $\vartheta^i \in L^i_\xi$ and all agent $i \in I$,

$$\begin{aligned}
\tilde{\xi}(\dot{s}, \vartheta^1, \vartheta^2, \ldots, \vartheta^{|I|}) &\stackrel{\text{def}}{=} \sum_{\theta^1 \in B_{\vartheta^1}} \sum_{\theta^2 \in B_{\vartheta^2}} \cdots \sum_{\theta^{|I|} \in B_{\vartheta^{|I|}}} \xi(\dot{s}, \theta^1, \theta^2, \theta^{|I|}), \\
&= \sum_{(\dot{s},\theta) \in B_{\dot{s},\vartheta}} \xi(\dot{s}, \theta), && (\vartheta \stackrel{\text{def}}{=} (\vartheta^1, \vartheta^2, \ldots, \vartheta^{|I|})) \\
&= \sum_{s \in S} \sum_{\theta \in \Theta(\xi)} \phi_{\dot{s},\vartheta}(s,\theta) \cdot \xi(s,\theta),
\end{aligned}$$

An analogous property holds for feature-based compact real vectors w.r.t. ξ. That is, a feature-based compact real vector $\tilde{\beta}_\xi$ is a $|\Phi_\xi|$-dimensional real vector that is expressed using Φ_ξ.

It is worth noticing that standard feature-based approaches assume a unique feature set and data are all expressed based on that unique feature set, which eases bound generalization (Tsitsiklis & van Roy, 1996). In our setting, however, different occupancy states yield different (possibly disjoint) feature sets. Hence, feature-based compact occupancy states (or real vectors) are expressed based on different feature sets, making it hard to transfer value from one feature-based compact occupancy state to another one. Bounds generalize naturally among feature-based compact occupancy states only if they share the same feature set. To remedy this, we introduce basis change operations for both feature-based compact occupancy states and real vectors.

4.2.2 CHANGE OF FEATURE SET

In enabling the bound generalization, it is necessary to work with more than one feature set. Hence, it is important to be able to easily transform feature vectors calculated with respect to one feature set to their corresponding (possibly lossy) representations with respect to another feature set. To ease the change of feature set, it is necessary to match features from the original feature set to those from the destination feature set. We introduce two heuristic methods to match features from different feature sets, thereby allowing the change of feature sets.

Definition 15. *Let Φ_ξ and $\Phi_{\xi'}$ be feature sets w.r.t. occupancy states ξ and ξ', respectively. The projection of feature-based compact occupancy state $\tilde{\xi}$ onto feature set $\Phi_{\xi'}$, denoted $F_{\Phi_{\xi'}}(\tilde{\xi})$, is given by probability distribution:*

$$F_{\Phi_{\xi'}}(\tilde{\xi}) \stackrel{\text{def}}{=} \left(\sum_{\phi_{s,\vartheta} \in \Phi_\xi} \phi_{\dot{s},\dot{\vartheta}}(s,\vartheta) \cdot \tilde{\xi}(s,\vartheta) \right)_{\phi_{\dot{s},\dot{\vartheta}} \in \Phi_{\xi'}}, \tag{36}$$

where $F_{\Phi_{\xi'}}(\tilde{\xi})$ reassigns the probability mass $\tilde{\xi}(s,\vartheta)$ of each pair (s,ϑ), such that $\phi_{s,\vartheta} \in \Phi_\xi$, to pair $(\dot{s},\dot{\vartheta})$, such that $\phi_{\dot{s},\dot{\vartheta}} \in \Phi_{\xi'}$, if and only if they match, i.e., $\phi_{\dot{s},\dot{\vartheta}}(s,\vartheta) = 1$.

This change of feature set describes a function from original feature set Φ_ξ to destination feature set $\Phi_{\xi'}$. In particular, it is a surjective function (*i.e.,* every feature $\phi_{\dot{s},\dot{\vartheta}}$ in the destination set has at least one corresponding feature $\phi_{s,\vartheta}$ in the original set — those such that $\phi_{\dot{s},\dot{\vartheta}}(s,\vartheta) = 1$). The transformation assigns to each feature in the destination set the probability mass of all corresponding features in the original set. This heuristic method provides no guarantee that both $F_{\Phi_{\xi'}}(\tilde{\xi})$ and $\tilde{\xi}$ share the same optimal value. By replacing a range of features in the original set by a single feature in the destination set, we produce compact but possibly lossy representations, which ultimately precludes ability to preserve the value of the original compact occupancy state. Fortunately, since we will use these representations to provide bounds, even lossy representations can generate useful bounds.

Definition 16. *Let Φ_ξ and $\Phi_{\xi'}$ be feature sets w.r.t. occupancy states ξ and ξ', respectively. The projection of feature-based compact real vector $\tilde{\beta}_\xi$ using feature set $\Phi_{\xi'}$, denoted $G_{\Phi_{\xi'}}(\tilde{\beta}_\xi)$, is given by feature vector:*

$$G_{\Phi_{\xi'}}(\tilde{\beta}_\xi) \stackrel{\text{def}}{=} \left(\sum_{\phi_{s,\vartheta} \in \Phi_\xi} \phi_{s,\vartheta}(\dot{s},\dot{\vartheta}) \cdot \tilde{\beta}_\xi(s,\vartheta) \right)_{\phi_{\dot{s},\dot{\vartheta}} \in \Phi_{\xi'}} \tag{37}$$

This change of feature set applies to real vectors, and describes a function from destination feature set $\Phi_{\xi'}$ to original feature set Φ_ξ. Specifically, every pair $(\dot{s},\dot{\vartheta})$ along with feature $\phi_{\dot{s},\dot{\vartheta}}$ in the destination set has a corresponding pair (s,ϑ) along with a feature $\phi_{s,\vartheta}$ in the original set — *i.e.,* the only one such that $\phi_{s,\vartheta}(\dot{s},\dot{\vartheta}) = 1$. The transformation assigns to each pair $(\dot{s},\dot{\vartheta})$ in the destination set the value $\tilde{\beta}_\xi(s,\vartheta)$ from their corresponding pair (s,ϑ) in the original set. Since not all pairs in the original set can have their values represented in $G_{\Phi_{\xi'}}(\tilde{\beta}_\xi)$, the resulting feature vector is a lossy representation of the original vector $\tilde{\beta}_\xi$. The loss in the resulting feature vector depends on original and destination feature sets, and the choice of the equivalence relation between histories. As previously mentioned, we will make use of either ξ-LPE or ξ-TPE relations.

Using ξ-LPE, we distinguish between state and joint-history pairs that involve only non-zero probability private history sets w.r.t. ξ (*i.e., non-zero probability pairs*), and state and joint-history pairs that involve zero probability private history sets w.r.t. ξ (*i.e., zero probability pairs*). For the sake of conciseness, compact real vectors maintain only values associated to non-zero probability pairs. All zero probability pairs have the same default value, for example $(T-t)\min_{s\in S, a\in A} r^a(s)$. Hence, the change of feature set is such that all pairs $(\dot{s},\dot{\vartheta})$ in the destination set, with a corresponding non-zero probability pair (s,ϑ) in the original set, inherits the value $\tilde{\beta}_\xi(s,\vartheta)$. However, all pairs $(\dot{s},\dot{\vartheta})$ in the destination set, with a corresponding zero probability pair (s,ϑ) in the original set, inherits the default (and loose) value. Because the number of zero probability pairs in occupancy state ξ is far larger than the number of non-zero probability pairs, feature vectors that result from the change of basis via ξ-LPE have multiple components with a loose value.

Using ξ-TPE, we distinguish between state and joint-history pairs only through joint-history suffixes of length m_ξ — see Definition 10. There are many pairs in the destination set, that would have been associated to a corresponding zero probability pair in the original set using ξ-LPE. Using ξ-TPE, however, these pairs are associated to a non-zero probability pair in the original set. Specifically, pair $(\dot{s},\dot{\vartheta})$ in the destination set, whose probability is zero with respect to ξ, has a corresponding zero probability pair in the original set using ξ-LPE. Yet, if pair $(\dot{s},\dot{\vartheta})$ in the destination set and non-zero probability pair (s,ϑ) in the original set share a common length m_ξ history suffix, then $(\dot{s},\dot{\vartheta})$ is associated to (s,ϑ) using ξ-TPE. Thus, feature vectors that result from the change of feature set using ξ-TPE involve fewer components with a default, loose value than those from using ξ-LPE.

Although heuristic methods for the change of feature set are not guaranteed to produce representations that retain all information of their compact counterparts, in many cases the resulting (possibly lossy) representation is sufficient to generalize bounds over the entire compact occupancy space. The bound property (*i.e.,* the ability to overestimate or underestimate the optimal value) of the resulting representation can be determined by examining methods for the change of feature set. The following theorem (proved in the Appendix) establishes that the F_Φ and G_Φ mappings we use preserve the bound property.

Theorem 6. *For any arbitrary feature set Φ, the change of feature set based on mappings G_Φ and F_Φ preserve the bound property — i.e., $\forall \tilde{\xi} \in \tilde{\triangle}_t$,*

a. $\forall v \in \mathbb{R}$, *if* $v \geq V^*_{\tilde{M},t}(\tilde{\xi})$, *then* $v \geq V^*_{\tilde{M},t}(F_\Phi(\tilde{\xi}))$;

b. $\forall \tilde{\beta} \in \mathbb{R}^{|\Phi'|}$, *if* $V^*_{\tilde{M},t}(\tilde{\xi}) \geq \langle \tilde{\xi}, G_{\Phi_\xi}(\tilde{\beta}) \rangle$, *then* $V^*_{\tilde{M},t}(\tilde{\xi}) \geq \langle \tilde{\xi}, G_{\Phi_\xi}(G_\Phi(\tilde{\beta})) \rangle$.

Theorem 6 shows that bound properties for a given occupancy state are preserved for the occupancy state obtained upon the change of feature set using heuristic methods F or G. Next, we discuss how to approximate full upper and lower bounds over the entire occupancy space.

4.2.3 Compact Point-Value Mappings: Upper Bounds

The full upper-bound value function can be approximated by a finite set of points and the sawtooth interpolation rule that estimates the value of an arbitrary point of the compact occupancy space by relying on the points already experienced and their associated values. A key aspect of this heuristic approximation is the sawtooth interpolation rule over compact occupancy states.

In the full upper-bound value function, the sawtooth interpolation can only approximate points expressed within the same basis set. Here, we demonstrate that it can apply even when points are expressed in different feature sets by means of the change of feature set. Let $\tilde{\Psi} = \{(\tilde{\xi}_1 \mapsto v^{\tilde{\xi}_1}), (\tilde{\xi}_2 \mapsto v^{\tilde{\xi}_2}), \ldots, (\tilde{\xi}_k \mapsto v^{\tilde{\xi}_k})\}$ be a set of point-value pairs that represents approximate function $U_{\check{M}}$ defined over the compact occupancy space, such that each point satisfies Theorem 6*a*. Then the approximate value for an arbitrary compact occupancy state $\tilde{\xi}'$ based on point-value pair $(\tilde{\xi} \mapsto v^{\tilde{\xi}})$ can be obtained using the sawtooth interpolation in a way that is similar to the calculation in the full upper-bound value function:

$$v^{\tilde{\xi}}_{\tilde{\xi}'} = v^*_{\tilde{\xi}'} + (v^{\tilde{\xi}} - v^*_{\tilde{\xi}}) \cdot D(F_{\Phi_{\xi'}}(\tilde{\xi}), \tilde{\xi}'), \tag{38}$$

where $v^*_{\tilde{\xi}} = \sum_{s,\vartheta} \tilde{\xi}(s,\vartheta) \cdot v^{s,*}$. From Theorem 6*a*, we know that feature vector $F_{\Phi_{\xi'}}(\tilde{\xi})$ shares the same upper bound $v^{\tilde{\xi}}$ with $\tilde{\xi}$. In addition, $F_{\Phi_{\xi'}}(\tilde{\xi})$ and $\tilde{\xi}'$ are expressed using the same feature set $\Phi_{\xi'}$. Hence the sawtooth interpolation can apply and produce upper-bound value $v^{\tilde{\xi}}_{\tilde{\xi}'}$ for compact occupancy state $\tilde{\xi}'$ based on point-value pair $(\tilde{\xi} \mapsto v^{\tilde{\xi}})$. The optimization with respect to compact occupancy state $\tilde{\xi}'$ is then acquired by choosing the best overall upper-bound value from all point-value pairs in $\tilde{\Psi}$:

$$U_{\check{M}}(\tilde{\xi}') = \min\ \{v^*_{\tilde{\xi}'}, v^{\tilde{\xi}}_{\tilde{\xi}'} \mid (\tilde{\xi} \mapsto v^{\tilde{\xi}}) \in \tilde{\Psi}_t\}. \tag{39}$$

This heuristic approximation differs from the full upper-bound value function only because it requires the change of feature set F_Φ and compact occupancy states instead of full occupancy states. Hence, this sawtooth interpolation is computationally more efficient than that of the full upper-bound value function, because compact occupancy states are of lower dimensionality. As for the accuracy of the resulting upper-bound values, it is not clear how this sawtooth interpolation compares with that from the full upper-bound value function (*i.e.,* whether or not the sawtooth interpolation weakens the upper bounds). Yet, a collection of point-value pairs obtained for a selection of compact occupancy states can be combined to define an approximate function of the upper-bound value function as discussed next.

A **feature-based compact value function** $(U_{\tilde{M},t})_{t\in\{0,1,\dots,T\}}$ over compact occupancy space $\tilde{\triangle}$ is represented using sets $(\tilde{\Psi}_t)_{t\in\{0,1,\dots,T\}}$ of point-value pairs that estimate values of any arbitrary compact occupancy state. Initially, each set $\tilde{\Psi}_t$ contains $|S|$ point-value pairs $\{\xi^{s,*} \mapsto v_t^{s,*} \mid s \in S\}$ that represent the step-t optimal value function of the underlying MDP. The sawtooth interpolation estimates $U_{\tilde{M},t}$ at any compact occupancy state $\tilde{\xi}' \in \tilde{\triangle}_t$ as follows:

$$U_{\tilde{M},t}(\tilde{\xi}') = \min\ \{v^*_{\tilde{\xi}'}, v^{\tilde{\xi}}_{\tilde{\xi}'} \mid (\tilde{\xi} \mapsto v^{\tilde{\xi}}) \in \tilde{\Psi}_t\}, \qquad \forall t \in \{0, 1, \dots, T\}.$$

Set $\tilde{\Psi}_t$ is updated for every compact occupancy state $\tilde{\xi}' \in \tilde{\triangle}_t$, using a point-based backup step as follows:

$$\tilde{\Psi}_t = \tilde{\Psi}_t \bigcup \left\{(\tilde{\xi}' \mapsto v^{\tilde{\xi}'})\right\}, \qquad \forall t \in \{0, 1, \dots, T\}$$

where $v^{\tilde{\xi}'} = \max_{\tilde{d}_{\xi'}\in\tilde{\boldsymbol{A}}}\ \tilde{\boldsymbol{R}}(\tilde{\xi}', \tilde{d}_{\xi'}) + U_{\tilde{M},t+1}(\tilde{\boldsymbol{P}}(\tilde{\xi}', \tilde{d}_{\xi'}))$. Approximate function $(U_{\tilde{M},t})_{t\in\{0,1,\dots,T\}}$ is an upper-bound value function similarly to the full upper-bound value function as stated below and proven in Appendix A.

Theorem 7. *Feature-based compact value function $(U_{\tilde{M},t})_{t\in\{0,1,\dots,T\}}$, as iteratively updated, upper bounds the optimal value function over the entire compact occupancy space.*

4.2.4 COMPACT VECTOR SETS: LOWER BOUNDS

A full lower bound over the full occupancy space can be approximated by a finite set of compact real vectors along with their associated feature sets and linear function updates. We take inspiration from initialization, evaluation and update routines from the full lower-bound value function. One important aspect of our approximation lies in the definition of the update operation, denoted $\widetilde{backup}$.

Let $\tilde{\Lambda}$ be a set of compact real vectors that represents the approximate value function, such that each compact real vector satisfies Theorem 6*b*. Then, a new compact real vector for any compact occupancy state $\tilde{\xi}$ and compact decision rule $\tilde{d}_\xi$ can be computed efficiently as in the full lower-bound value function (Section 3.2.2):

$$g^{\tilde{\xi}}_{\tilde{d}_\xi} = r^{\tilde{d}_\xi} + \arg\max_{g^\alpha_{\tilde{d}_\xi} : \alpha\in\tilde{\Lambda}} \left\langle \tilde{\xi}, g^\alpha_{\tilde{d}_\xi} \right\rangle, \tag{40}$$

where $g^\alpha_{\tilde{d}_\xi} = G_{\Phi_{\boldsymbol{P}(\tilde{\xi},\tilde{d}_\xi)}}(\alpha) \cdot (p^{\tilde{d}_\xi})^\top$ is the expression of the projection of $G_{\Phi_{\boldsymbol{P}(\tilde{\xi},\tilde{d}_\xi)}}(\alpha)$ onto feature set Φ_ξ, and $G_{\Phi_{\boldsymbol{P}(\tilde{\xi},\tilde{d}_\xi)}}(\alpha)$ is the expression of α in feature set $\Phi_{\boldsymbol{P}(\tilde{\xi},\tilde{d}_\xi)}$. The optimization with respect to compact occupancy state $\tilde{\xi}$ is then acquired by choosing the compact vector with the best overall value from all vectors $g^{\tilde{\xi}}_{\tilde{d}_\xi}$:

$$\widetilde{backup}(\tilde{\Lambda}, \tilde{\xi}) = \arg\max_{g^{\tilde{\xi}}_{\tilde{d}_\xi} \,:\, \tilde{d}_\xi \in \tilde{\boldsymbol{A}}} \left\langle \tilde{\xi}, g^{\tilde{\xi}}_{\tilde{d}_\xi} \right\rangle. \tag{41}$$

The principal difference with respect to the full lower-bound value function lies in the use of transformation G_Φ and compact real vectors instead of high-dimensional real vectors. Hence, this backup operation is more efficient than that of the full lower-bound value function, since operations are the same but we now use only lower dimensional vectors, which is likely to save significant time. But the change of feature set may produce weaker bounds. Nonetheless, a collection of compact vectors obtained for a selection of compact occupancy states can be combined to define an approximate function of the lower-bound value function as discussed next.

A **feature-based compact value function** $(L_{\tilde{M},t})_{t\in\{0,1,\ldots,T-1\}}$ over the compact occupancy space $\tilde{\triangle}$ is represented using sets $(\tilde{\Lambda}_t)_{t\in\{0,1,\ldots,T-1\}}$ of compact real vectors along with their associated feature sets that estimate values of any arbitrary compact occupancy state. Initially, each set $\tilde{\Lambda}_t$ contains a single compact real vector $\tilde{\beta}_t$ given by

$$\tilde{\beta}_t(\cdot) = (T-t) \min_{s\in S, a\in A} r^a(s), \qquad \forall t \in \{0, 1, \ldots, T\}. \tag{42}$$

The max-vector rule estimates $L_{\tilde{M},t}$ at any compact occupancy $\tilde{\xi}_t \in \tilde{\triangle}_t$ as follows:

$$L_{\tilde{M},t}(\tilde{\xi}_t) = \max_{(\Phi\mapsto\tilde{\beta}_t)\in\tilde{\Lambda}_t} \langle \tilde{\xi}_t, G_{\Phi_{\xi_t}}(\tilde{\beta}_t)\rangle, \qquad \forall t \in \{0, 1, \ldots, T\}. \tag{43}$$

Set $\tilde{\Lambda}_t$ is updated for every compact occupancy state $\tilde{\xi}_t \in \tilde{\triangle}_t$, using point-based backup steps as follows:

$$\tilde{\Lambda}_t = \tilde{\Lambda}_t \bigcup \left\{ (\Phi_{\xi_t} \mapsto \widetilde{backup}(\tilde{\Lambda}_{t+1}, \tilde{\xi}_t)) \right\}. \tag{44}$$

Approximate function $(L_{\tilde{M},t})_{t\in\{0,1,\ldots,T-1\}}$ is a lower-bound value function since its main difference with respect to the full lower-bound value function lies in the use of G_Φ, which preserves the bound property. For a complete proof, the reader can refer to Appendix A.

Theorem 8. *Feature-based compact value function $(L_{\tilde{M},t})_{t\in\{0,1,\ldots,T\}}$, lower bounds the optimal value function over the entire compact occupancy space.*

4.3 Feature-Based Heuristic Search Value Iteration Algorithm

This section presents the feature-based heuristic search value iteration algorithm (FB-HSVI) that iteratively updates feature-based compact lower- and upper-bound representations. We also discuss FB-HSVI's theoretical guarantees.

4.3.1 Algorithm Description

Similar to OHSVI (Algorithm 1), FB-HSVI (Algorithm 2) solves occupancy-state MDPs by generating trajectories of occupancy states and iteratively updating lower and upper bounds, but in the case of FB-HSVI, these are now compact occupancy states and feature-based compact lower $(L_{\tilde{M},t})_{t\in\{0,1,\ldots,T\}}$ and upper bounds $(U_{\tilde{M},t})_{t\in\{0,1,\ldots,T\}}$. FB-HSVI improves the scalability of OHSVI in several ways. First, FB-HSVI replaces full exact representations by compact representations for all: occupancy states, decision rules, lower and upper bounds. In addition, it combines stopping criteria from HSVI (Smith, 2007) and optimal classical heuristic search methods (e.g., Hart et al., 1968; Korf, 1990), which may result in more efficient pruning of unnecessary subspaces.

Algorithm 2: The FB-HSVI Algorithm.

```
function FB-HSVI((L_{M̃,t})_{t∈{0,1,···,T}}, (U_{M̃,t})_{t∈{0,1,···,T}})
    initialize L_{M̃,t} and U_{M̃,t} for all t ∈ {0, ··· , T − 1}.
    while ¬ Stop (ξ̃_0, 0) do Explore(ξ̃_0, 0)

function Explore(ξ̃_t, g_t)
    if ¬ Stop (ξ̃_t, g_t) then
        d̃_t ∈ arg max_{d̃'_t ∈ Ã} R̃(ξ̃_t, d̃'_t) + U_{M̃,t+1}(P̃(ξ̃_t, d̃'_t))
        Update U_{M̃,t}
        Explore(P̃(ξ̃_t, d̃_t), R̃(ξ̃_t, d̃_t) + g_t)
        Update L_{M̃,t}
    return g_t

function Stop(ξ̃_t, g_t)
    return U_{M̃,t}(ξ̃_t) ≤ L_{M̃,t}(ξ̃_t) ∨ L_{M̃,0}(ξ̃_0) ≥ g_t + U_{M̃,t}(ξ̃_t)
```

FB-HSVI differs from OHSVI in four main ways:

1. *compact representation of occupancy states*, which significantly reduces the search space;
2. *compact representation of decision rules*, which speeds up the decision-rule selection;
3. *compact representation of lower and upper bounds*, which speeds up the convergence;
4. *enhanced value function generalization and combination of stopping criteria*, which results in more efficient pruning of unnecessary subspaces.

4.3.2 STOPPING CRITERIA

The stopping criteria of FB-HSVI build upon those from optimal classical heuristic search methods (e.g., Hart et al., 1968; Korf, 1990; Smith, 2007). They determine when to stop the current trajectory of compact occupancy states in the algorithm. Ideally, an optimal criterion would measure the distance from the current trajectory to an optimal trajectory, but this is not known. Instead, we use two criteria based on the upper and lower bound values of trajectories and compact occupancy states. The upper bound of the current trajectory, which we denote $f(\tilde{\xi}_t)$, is the sum of two functions: (1) the past trajectory-reward function $g(\tilde{\xi}_0, \tilde{\xi}_t)$, which is the sum of rewards from the starting compact occupancy state $\tilde{\xi}_0$ to the current occupancy state $\tilde{\xi}_t$; and (2) the future trajectory-reward from compact occupancy state $\tilde{\xi}_t$, which is an admissible heuristic estimate, *e.g.,* upper-bound $U_{\tilde{M},t}(\tilde{\xi}_t)$ at compact occupancy state $\tilde{\xi}_t$.

The first criterion relies on the fact that there is no reason to expand an occupancy state $\tilde{\xi}_t$ that has $f(\tilde{\xi}_t)$ less than or equal to $L_{\tilde{M},0}(\tilde{\xi}_0)$, since it cannot lead to a solution better than the current best solution; a criterion previously used in optimal classical heuristic search methods (e.g., Hart et al., 1968; Korf, 1990).

Criterion 1. *A trajectory of occupancy states* $(\tilde{\xi}_0, \dots, \tilde{\xi}_t)$ *is interrupted whenever heuristic value* $f(\tilde{\xi}_0)$ *is less than or equal to* $L_{\tilde{M},0}(\tilde{\xi}_0)$ *— i.e.,* $L_{\tilde{M},0}(\tilde{\xi}_0) \geq f(\tilde{\xi}_0)$*. The best solution found so far is optimal if there is no expanded occupancy state* $\tilde{\xi}_t$ *on the frontier of the search space*[2] *with a heuristic-value* $f(\tilde{\xi}_t)$ *higher than* $L_{\tilde{M},0}(\tilde{\xi}_0)$*.*

The second criterion builds upon the fact that there is no reason to expand an occupancy state $\tilde{\xi}_t$ that has upper bound less than or equal to its lower bound (Smith, 2007).

Criterion 2. *A trajectory of occupancy states* $(\tilde{\xi}_0, \dots, \tilde{\xi}_t)$ *is interrupted whenever upper bound* $U_{\tilde{M},t}(\tilde{\xi}_t)$ *is less than or equal to lower bound* $L_{\tilde{M},t}(\tilde{\xi}_t)$ *— i.e.,* $L_{\tilde{M},t}(\tilde{\xi}_t) \geq U_{\tilde{M},t}(\tilde{\xi}_t)$*. The best solution found so far is optimal if upper and lower bounds at the initial occupancy state are equal.*

By interrupting any trajectory that satisfies either of criterion 1 or 2, FB-HSVI preserves the ability to find an optimal separable joint policy, as shown below.

4.3.3 Convergence Guarantees

Theorems 7 and 8 show the feature-based compact functions $(L_{\tilde{M},t})_{t\in\{0,1,\cdots,T\}}$ and $(U_{\tilde{M},t})_{t\in\{0,1,\cdots,T\}}$ as iteratively updated by FB-HSVI (Algorithm 2) upper and lower-bounds the optimal value function. Next, we prove that upon the update of each trajectory no compact occupancy state has its bounds depreciated, and at least one compact occupancy state improves its bounds. Since there is a finite number of compact occupancy states, the bounds ultimately converge to the optimal value at the initial occupancy state. Here, we use this argument to show that FB-HSVI converges to an optimal separable joint policy after a finite number of iterations. To this end, a compact occupancy state is said to be *finished* if either criterion 1 or 2 is satisfied; otherwise it is *not finished*. Moreover, all compact occupancy states $\tilde{\xi}_T$ at the last time step T are finished since criterion 2 is satisfied at the last time step T.

Theorem 9. *The FB-HSVI algorithm always terminates after a finite number of trials and the solution found at termination — the separable joint policy the lower bound induces — is optimal.*

Proof. First, we show by contradiction that the algorithm cannot terminate before an optimal solution is found. Suppose the algorithm terminates before finding an optimal solution which has value $f^*(\tilde{\xi}_0)$. Then, the sequence of $f(\tilde{\xi}_0)$ values generated while planning is $f^0(\tilde{\xi}_0), f^1(\tilde{\xi}_0), \dots, f^k(\tilde{\xi}_0)$, where $f^0(\tilde{\xi}_0)$ is the initial lower bound before any solution is found, $f^1(\tilde{\xi}_0)$ is the value of the first solution found, and $f^k(\tilde{\xi}_0)$ that of the last solution found. In addition, we know by hypothesis that $f^0(\tilde{\xi}_0) < f^1(\tilde{\xi}_0) < \dots < f^k(\tilde{\xi}_0) \leq f^*(\tilde{\xi}_0)$, where the last inequality holds under the assumption that the algorithm may terminate with a suboptimal solution.

Now consider an optimal path $\tilde{\xi}_0, \tilde{\xi}_1, \dots, \tilde{\xi}_T$ leading from the initial occupancy state to a terminal occupancy state. Under the assumption that this optimal path was not found, there must be some occupancy state $\tilde{\xi}_t$ along this path that was generated but not expanded. This is only possible if $f(\tilde{\xi}_t) \leq f^k(\tilde{\xi}_0)$. But by the admissibility of f, we know that $f(\tilde{\xi}_t) \geq f^*(\tilde{\xi}_0)$ and therefore $f(\tilde{\xi}_t) \geq f^*(\tilde{\xi}_0) > f^m(\tilde{\xi}_0)$ for any $m \in \{0, 1, \dots, k\}$. From this contradiction, it follows that the algorithm cannot terminate before an optimal solution is found.

Next, we show that each trial turns at least one not finished occupancy state into a finished one. Suppose the algorithm has not yet terminated and a trial is executed. Let the last two occupancy

2. Typically, search algorithms involves expanding nodes by adding all unexpanded neighboring nodes into a priority queue, called a frontier of the search space.

states encountered during the forward expansion be $\tilde{\xi}_t$ and $\tilde{\xi}_{t+1}$. Given that the trial terminated at $\tilde{\xi}_{t+1}$, we know that before the trial, $\tilde{\xi}_t$ was not finished but $\tilde{\xi}_{t+1}$ was finished. Because $\tilde{\xi}_{t+1}$ results from a greedy separable joint decision rule selection at $\tilde{\xi}_t$, we know $\tilde{\xi}_t$ will be finished after being updated. This is because only two scenarios are possible, each of which corresponds to a stopping condition:

- either $\tilde{\xi}_{t+1}$ yields its optimal value, then $\tilde{\xi}_t$ will also yield its optimal value after being updated, making it a finished occupancy state after the update;
- or $\tilde{\xi}_{t+1}$ has $f(\xi_{t+1})$ lower than or equal to $L_{\tilde{M},0}(\tilde{\xi}_0)$, then $\tilde{\xi}_t$ will also have an $f(\tilde{\xi}_t)$ lower than or equal to $L_{\tilde{M},0}(\tilde{\xi}_0)$ after being updated.

Thus, executing a trial causes occupancy state $\tilde{\xi}_t$, which was not finished, to become finished.

Finally, we show that the algorithm terminates after a finite number of trials. To this end, we note that the search graph of the algorithm is a tree similar to Figure 8, with a bounded branching factor $|\tilde{\boldsymbol{A}}_t|$ at depth $t \in \{0, 1, \ldots, T\}$. By hypothesis, only occupancy states that appear at depth $t < T$ are not finished. Thus, the total number of occupancy states at all depths up to time step T is upper bounded by the total number of information states at all depths up to time step T:

$$| \cup_{t=0}^{T-1} \boldsymbol{S}_t| = |A^*| \frac{|I||Z^*|^T |A^*|^T - 1}{|Z^*||A^*| - 1}, \tag{45}$$

where $A^* = \max_{i \in I} A^i$ and $Z^* = \max_{i \in I} Z^i$. Given that at least one occupancy state becomes finished after each trial, the initial occupancy state must become finished after at most $|\cup_{t=0}^{T-1} \boldsymbol{S}_t|$ trials, causing the algorithm to terminate. □

Another important property of FB-HSVI is that it refines both upper and lower bounds throughout planning. Since compact occupancy state expansions are interleaved with updates, FB-HSVI offers an anytime solution. Furthermore, cutting off FB-HSVI trials at any time, we know that the difference between the current best solution and the optimal one is bounded.

Theorem 10. *At any iteration of FB-HSVI, the current solution — and the separable joint policy induced by the current lower bound — is within $\epsilon = U_{\tilde{M},0}(\xi_0) - L_{\tilde{M},0}(\xi_0)$ of the optimal solution.*

Proof. Formally, the difference in value between executing the separable joint policy π^{LB} induced by the current lower bound instead of an optimal separable joint policy π^* is written as follows:

$$V_{\check{M},\pi^*}(\xi_0) - V_{\check{M},\pi^{\text{LB}}}(\xi_0) = V_{\check{M},\pi^*}(\xi_0) - L_{\tilde{M},0}(\xi_0), \qquad (V_{\check{M},\pi^{\text{LB}}}(\xi_0) = L_{\tilde{M},0}(\xi_0)) \tag{46}$$

$$\leq U_{\tilde{M},0}(\xi_0) - L_{\tilde{M},0}(\xi_0). \qquad (V_{\check{M},\pi^*}(\xi_0) \leq U_{\tilde{M},0}(\xi_0)) \tag{47}$$

Consequently, whenever FB-HSVI is interrupted, its current solution is within the ϵ given above of the optimal solution. □

5. Experiments

This section empirically demonstrates and validates the importance of our feature-based heuristic search value iteration (FB-HSVI) algorithm. We show that FB-HSVI outperforms all existing exact algorithms on all tested domains from the literature and that FB-HSVI can solve those problems over unprecedented time horizons.

5.1 Experimental Setup

As discussed throughout this paper, there are many key components that can affect the performance of FB-HSVI. These key components include the (upper and lower) bound representations, the bound update methods, the history compression, the value generalization, and the initial upper bound. We present three variants of FB-HSVI, denoted OHSVI, FB-HSVI-LPE and FB-HSVI-TPE. The two latter differ only on the notion of history equivalence they use in the feature-based compact representations (see Table 2). When no equivalence relation is given, FB-HSVI refers to its default (and better performing) implementation, FB-HSVI-TPE.

Algorithm	Bound Representations	Compression
OHSVI	full	NONE
FB-HSVI-LPE	feature-based compact	LPE
FB-HSVI-TPE	feature-based compact	TPE

Table 2: A review of the selected algorithmic components we use.

We selected benchmarks with the goal of spanning the range of properties that may affect the performance of a Dec-POMDP solver. In Table 3, we review the selected domains and their properties. These domains can be downloaded at `http://masplan.org`.

	domain M parameters				$\lvert\Pi_{0:t}\rvert$ for different T		
	N	$\lvert S\rvert$	$\lvert A^i\rvert$	$\lvert Z^i\rvert$	$T = 2$	$T = 5$	$T = 10$
DEC-TIGER	2	2	3	2	6561	3.43×10^{30}	1.39×10^{977}
MABC	2	4	2	2	256	1.84×10^{19}	3.23×10^{616}
GRID-SMALL	2	16	5	2	390625	5.42×10^{44}	3.09×10^{1431}
RECYCLING-ROBOTS	2	4	3	2	6561	3.43×10^{30}	1.39×10^{977}
BOX PUSHING	2	100	4	5	3.34×10^{7}	5.23×10^{940}	1.25×10^{2939746}
MARS ROVERS	2	256	6	8	1.69×10^{14}	1.88×10^{7285}	$2.57 \times 10^{238723869}$

Table 3: Domain parameters and maximum number of separable joint policies per horizons.

5.2 Empirical Analysis of our Algorithms

In this section, we compare FB-HSVI to other exact solvers. The exact Dec-POMDP solvers considered are the state-of-the-art methods including: GMAA*-ICE (Oliehoek et al., 2013), IPG (Amato et al., 2009), MILP (Aras & Dutech, 2010), and LPC (Boularias & Chaib-draa, 2008). IPG and LPC perform dynamic programming, GMAA*-ICE performs heuristic search and MILP is a mixed integer linear programming method. Results for GMAA*-ICE (provided by Matthijs Spaan), IPG, MILP, LPC were conducted on different machines. Because of this, the timing results are not directly comparable, but are likely to only differ by a small constant factor. Our three FB-HSVI variants (Table 2) were implemented in the same framework, using identical basic operations, such as occupancy state and value function updates, and separable joint decision rule selection. We terminate FB-HSVI whenever the distance between lower and upper bounds is within $\epsilon = 0.01$. A time limit was set to 1000ms.

5.2.1 COMPARING TO OTHER EXACT PLANNERS

T	MILP	LPC	IPG	ICE	OHSVI	**FB-HSVI**	$L_{0,\check{M}}(\xi_0)$
MULTI-AGENT TIGER							
2	–	0.17	0.32	0.01	0.161	**0.03**	−4.00
3	4.9	1.79	55.4	0.01	28.567	**0.40**	5.1908
4	72	534	2286	108		**1.36**	4.8027
5				347		**9.65**	7.0264
6						**24.42**	10.381
7						**33.11**	**9.9935**
8						**41.21**	**12.217**
9						**58.51**	**15.572**
10						**65.57**	**15.184**
RECYCLING ROBOT							
2	–	–	0.30	36	0.04	**0.01**	7.000
3	–	–	1.07	36	0.555	**0.10**	10.660
4	–	–	42.0	72	696.8	**0.30**	13.380
5	–	–	1812	72		**0.34**	16.486
10						**0.52**	31.863
30						**1.13**	93.402
70						**2.13**	216.47
100						**2.93**	**308.78**
MEETING IN A 3x3 GRID							
2	–	–	–	–	93.029	**0.03**	0.0
3	–	–	–	–		**0.04**	0.133
4	–	–	–	–		**0.79**	0.432
5	–	–	–	–		**1.30**	0.894
6						**1.88**	**1.491**
7						**2.55**	**2.19**
8						**18.06**	**2.96**
9						**24.39**	**3.80**
10						**34.42**	**4.68**
20						**291.1**	**14.35**
30						**456.6**	**24.33**

Table 4: Experiments comparing the computation times (in seconds) of all exact solvers (part 1). Time limit violations are indicated by “ ”, “–” indicate unknown values. Bold entries correspond to the best known results for these benchmarks, both in terms of computational time and expected value.

Tables 4 and 5 show performance results for the exact algorithms. For each algorithm, we reported the computation time, which includes the time to compute heuristic values when appropriate (since all algorithms do not use the same heuristics). We also reported the best expected cumulative reward $L_{\check{M},0}(\xi_0)$ at the initial occupancy state. Tables 4 and 5 clearly show that FB-HSVI allows for significant improvement over the state-of-the-art solvers: for all tested benchmarks we provide results for longer horizons than have been solved previously (the bold entries). In many cases, an (epsilon) optimal solution can be found for horizons that are an order of magnitude larger than was previously solvable. There are two main reasons for FB-HSVI's performance. First, it searches in the space of policies mapping lower-dimensional features to actions, whereas the other exact solvers search in the space of policies mapping full histories to actions. In addition, it uses a value function mapping occupancy states to reals allowing it to generalize the value function over unvisited occupancy states whereas all other solvers use value functions mapping partial policies to reals. FB-HSVI performs best when the domain possesses structure that results in a compact value function, as in the RECYCLING ROBOT and MABC domains.

T	MILP	LPC	IPG	ICE	OHSVI	**FB-HSVI**	$L_{0,\tilde{M}}(\xi_0)$
BROADCAST CHANNEL							
2	–	–	–	–	0.036	**0.02**	2
3	–	–	–	–	3.446	**0.22**	2.99
4	–	–	–	–		**0.32**	3.89
5	–	–	–	–		**0.33**	4.79
10						**0.78**	9.29
30						**14.0**	27.42
50						**41.7**	45.50
100						**473.3**	90.76
GRID SMALL							
2	0	–	0	36	0.911	**0**	0.37
3	0.65	–	0.18	36		**0.1**	0.91
4	1624	–	4.09	1512		**0.73**	1.55
5		–	77.4	242605		**1.39**	2.24
6						**3.51**	2.97
7						**8.30**	3.71
8						**42.2**	**4.47**
9						**69.03**	**5.23**
10						**581.2**	**6.03**
COOPERATIVE BOX-PUSHING							
2	–	–	1.07	36	0.294	**0.1**	17.600
3	–	–	6.43	540		**0.457**	66.081
4	–	–	1138	2596		**0.622**	98.593
5						**5.854**	**107.72**
6						**10.7**	**120.67**
7						**24.96**	**156.42**
8						**28.97**	**191.22**
9						**184.3**	**210.27**
10						**293.7**	**223.74**
MARS ROVERS							
2	–	–	83	1.0	0.027	**0.10**	5.80
3	–	–	389	1.0	1.881	**0.23**	9.38
4				103		**0.47**	10.18
5						**0.82**	**13.26**
6						**3.97**	**18.62**
7						**5.81**	**20.90**
8						**22.8**	**22.47**
9						**26.5**	**24.31**
10						**62.7**	**26.31**

Table 5: Experiments comparing the computation times (in seconds) of all exact solvers (part 2).

5.2.2 Choosing a Method to Keep Information Concise

We now compare the local and truncation probabilistic equivalence notions we introduced to maintain concise the representations of the occupancy states, decision rules, and value functions.

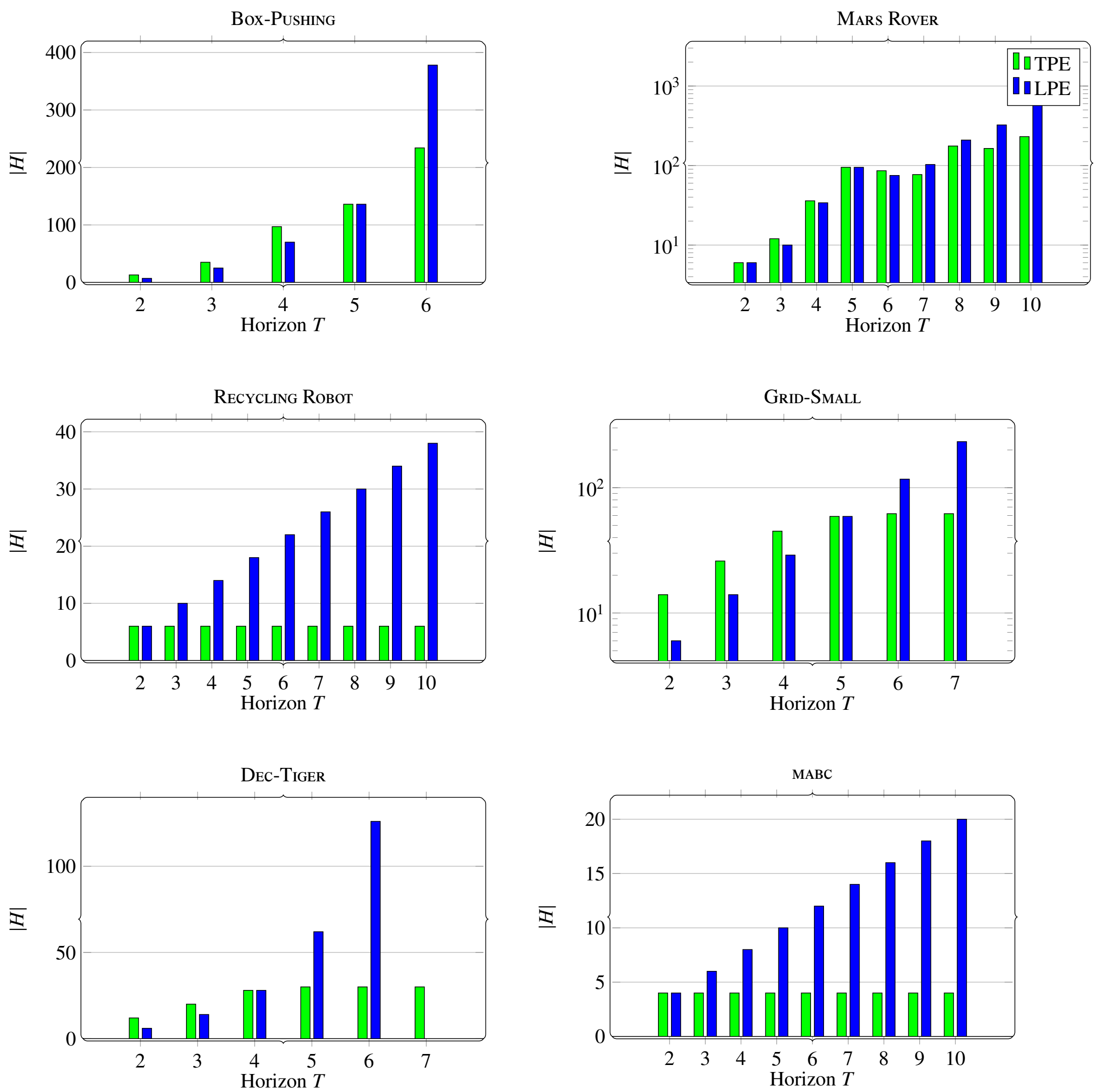

Figure 12: Comparison of compression methods to maintain concise data through FB-HSVI-LPE and FB-HSVI-TPE. All graphs shows the memory requirements until convergence or time exceeds in the y-axis given the various number of planning horizons in x-axis.

Clearly, algorithms that use feature-based compact representations provide significant savings in the number of maintained histories over those that do not (*e.g.,* OHSVI). Using OHSVI, the number of generated histories grows (in the worst case) exponentially with the planning horizon. It is this exponential growth that explains why OHSVI, which does not use history aggregation, cannot scale beyond planning horizon $T = 4$ over all tested domains (see Tables 4 and 5). For Recycling Robot at horizon $T = 5$, experimental results together with Table 3 show that algorithms that use our compression methods maintain up to 30 orders of magnitude less separable joint policies than algorithms that do not. The number of histories retained is important as all occupancy states, decision rules, and value functions are mappings from reachable histories (or corresponding labels). To this end, we compare LPE and TPE over our selection of benchmarks and various planning horizons.

As previously discussed, though LPE yields compact occupancy states that are more concise than those that result from TPE, the latter eases the generalization of bounds, which speeds up the convergence to an optimal solution. Figure 12 reports the total number of joint labels — denoted $|L|$ — that are explicitly maintained for FB-HSVI-LPE and FB-HSVI-TPE over various planning horizons. We observe that TPE yields more concise bound representations than LPE over most benchmarks and planning horizons (*i.e.,* using TPE number $|L|$ is lower than that using LPE). In particular, we notice that, in all tested domains, there is a bounded number of labels that is sufficient for representing optimal or near-optimal value functions. TPE often succeeds in identifying this memory-bounded parametric space, resulting in more concise value functions, whereas LPE often fails.

In the Recycling Robot problem for example, TPE yields no more than 6 joint labels (*i.e.,* histories) for all horizons whereas LPE maintains up to 38 different joint labels, a number that keeps growing as the planning horizon increases. In fact, (Dibangoye, Amato, & Doniec, 2012; Becker, Zilberstein, Lesser, & Goldman, 2004) demonstrated that in the Recycling-Robot problem, the most recent private observation is sufficient to summarize all past private histories of each agent (*i.e.,* only the four joint observations are necessary). Here, TPE yields 6 joint labels because it relies on joint action-observation histories rather than joint observation histories. In the Broadcast-Channel domain, TPE yields no more than 4 joint labels for all horizons whereas LPE produces up to 20 different joint labels. Again, these results are due to the underlying structure of the Broadcast-Channel domain. In such a scenario, the future states of the world are conditionally independent of joint histories. Hence, TPE can forget about joint histories, and reason only about states. Another domain of interest is the Dec-Tiger problem. In this problem, for $T = 6$, TPE produces no more than 30 joint labels for all horizons whereas LPE maintains up to 126 different joint labels. Our assumption is that there always exists an optimal separable joint policy that is periodic (with period 3) for the Dec-Tiger domain. In other words, there exists an optimal separable joint policy that depends on histories only upon the most recent three action-observation pairs. Also, there are many scenarios in which both equivalence relations would fail to identify a memory-bounded space of histories, even if such a space exists. For example, important information in a history may be spread over a few time steps, but not necessarily the last ones.

6. Discussion

While we have demonstrated that our method can solve Dec-POMDPs which are larger than those previously solved, many practical applications are much larger than domains considered in this paper. As a result, additional methods may be necessary to solve very large problems which do not permit the construction of a concise feature space while preserving optimality. This is of concern since the numbers of states and histories impact all occupancy states, separable joint decision rules, and value functions. Maintaining these objects for large feature spaces is prohibitive. This highlights the necessity of addressing the scalability issue of FB-HSVI through more concise (and possibly lossy) feature spaces. In that direction, we have already extended the general methodology presented in this paper along two lines: *error-bounded approximations* and *tractable subclasses*.

6.1 Error-Bounded Approximations

FB-HSVI can find an optimal solution because it maintains concise representations that preserve optimality. This is both an advantage and a liability. On the one hand, for problems of a reasonable size, the algorithm can find an optimal solution. On the other hand, in many realistic applications, it will run out of time or memory. These scalability limitations are because FB-HSVI maintains accurate estimates of (compact) occupancy states, value functions and decision rules. To improve the scalability of Dec-POMDP solvers, many researchers have investigated approximate solutions.

A notable example of this family includes the *memory-bounded dynamic programming* (MBDP) algorithm for finite-horizon Dec-POMDPs (Seuken & Zilberstein, 2007; Carlin & Zilberstein, 2008; Dibangoye, Mouaddib, & Chaib-draa, 2009; Kumar & Zilberstein, 2010; Wu, Zilberstein, & Chen, 2010). These are dynamic programming methods that require bounded computational resources to produce heuristic solutions that empirically perform well in standard Dec-POMDP benchmarks. However, these methods do not possess any theoretical guarantees concerning the quality of their solutions.

Recently, we introduced a framework for monitoring the error in FB-HSVI by replacing an exact estimate of (compact) occupancy states, decision rules and value functions, by their approximate counterparts (Dibangoye, Mouaddib, & Chaib-draa, 2011; Dibangoye, Buffet, & Charpillet, 2014). The resulting algorithm can solve Dec-POMDP instances with larger planning horizon while still providing strong theoretical guarantees.

It is also worth noting that, because FB-HSVI is trial-based, it can be used as an anytime algorithm. That is, it alternates between the generation of an occupancy-state trajectory and the update of the current best value function. As the algorithm proceeds, the current (best) value function is improved at the expense of increased computational time. The algorithm can be terminated either when a satisfactory value function is attained, or when allocated planning time is exceeded. In either case, this algorithm can always provide online performance bounds on the returned value function illustrating how far from the optimal value function the current one is.

In the future, we also would like to explore using occupancy states over observation histories (rather than action-observation histories), which were shown to be sufficient (along with action-observation histories) simultaneous to this work (Oliehoek, 2013). The inclusion of observation histories could lead to further scalability gains by reducing the dimensionality of the feature space.

6.2 Tractable Subclasses

Many attempts to address the scalability issues in Dec-POMDPs rely on the use of tractable subclasses. These subclasses have additional assumptions that allow more concise representations for all occupancy states, decision rules and value functions; and therefore speed up the convergence towards an optimal solution.

For instance, we have already shown that occupancy states over just states (and not agent histories) can be used in transition- and observation-independent Dec-MDPs (Becker et al., 2004) (where the state is fully determined by the joint observation) to greatly increase scalability while preserving optimality (Dibangoye et al., 2012; Dibangoye, Amato, Doniec, & Charpillet, 2013). By restricting attention to decentralized Markov policies (*i.e.,* mappings from private states to private actions) we reduce the complexity significantly (NP versus NEXP), and make it possible to optimally solve larger problems. We plan to investigate other forms of tractable structures including temporal dependencies or constraints that induce structured domains in both single and multi-agent settings (Dibangoye, Chaib-draa, & Mouaddib, 2008; Dibangoye, Shani, Chaib-Draa, & Mouaddib, 2009; Pajarinen, Hottinen, & Peltonen, 2013). In that line of research, we introduced a novel approach called structural analysis as a means of discovering the underlying structural properties embedded in certain decentralized decision-making problems (Dibangoye, Buffet, & Simonin, 2015).

We also applied the general methodology presented in this paper to scale up the number of agents involved in the process. To this end, we consider domains that exhibit the locality of interactions (Dibangoye, Amato, Buffet, & Charpillet, 2015, 2014). Examples include the networked distributed partially observable Markov decision processes (ND-POMDPs) (Nair, Varakantham, Tambe, & Yokoo, 2005). We plan to explore applying our methodology and FB-HSVI to Dec-POMDPs where agents have joint dynamics or rewards, as well as domains with delayed communication (Ooi & Wornell, 1996; Grizzle, Marcus, & Hsu, 1981; Oliehoek & Spaan, 2012), as a means of reducing the memory burden.

A secondary (but no less important) issue concerning scalability in Dec-POMDPs pertains to efficient methods to update occupancy states and value functions at the planning stage. The locality of interaction among agents may be exploited statically (e.g., Nair et al., 2005; Kumar & Zilberstein, 2009; Amato, Konidaris, & Kaelbling, 2014) or dynamically (e.g., Canu & Mouaddib, 2011) by considering factorization and graphical models in our representation and hence improve scalability. This is a critical issue as the number of occupancy states necessary to obtain a good solution may be exponential in the planning horizon. So, techniques that can efficiently update both occupancy states and value functions are of great importance.

7. Conclusion

This paper describes a novel way of representing Dec-POMDPs, as continuous-state MDPs with piecewise-linear convex value functions, and a scalable algorithm for generating ϵ-optimal solutions. We summarize the key contributions below.

By exploiting the assumption of centralized planning for decentralized execution, our method recasts the Dec-POMDP problem into an equivalent deterministic and centralized fully observable MDP (using information that is available to all agents). Next, we identify a concise statistic — *the occupancy state* — that represents the state of the resulting fully observable MDP, which we call the *occupancy-state MDP*. We demonstrate that the optimal value functions of occupancy MDPs are piecewise linear and convex functions of the occupancy states. We also prove that an optimal solution of the occupancy-state MDP is an optimal solution to the corresponding Dec-POMDP.

We also present the feature-based heuristic search value iteration (FB-HSVI) algorithm to find an optimal solution to the occupancy-state MDP. This algorithm builds off the theory for solving POMDPs and MDPs, as our occupancy-state MDP allows these methods to be directly applied to Dec-POMDPs for the first time. We believe FB-HSVI is a major step forward in scalable exact solutions for Dec-POMDPs. This scalability is achieved by defining feature-based compact occupancy states and decision rules through the use of equivalence relations between private histories. These concise representations permit us to circumvent the exhaustive enumeration of an otherwise intractable number of occupancy states and decision rules.

Another aspect of the improved scalability stems from generalization of the value function. This is achieved through the piecewise linear and convex functions in the occupancy-state MDP. We show that, although feature-based compact lower and upper bounds are no longer piecewise-linear and convex, they can still generalize value functions over the entire feature-based compact occupancy-state space.

Experimentally, we show that FB-HSVI is able to outperform all current state-of-the-art exact Dec-POMDP solvers in common benchmark domains. These results show that ϵ-optimal solutions can be found for larger horizons in all problems and for horizons that are sometimes an order of magnitude larger than those that have previously been solved.

8. Acknowledgements

We would like to thank Matthijs Spaan for providing results for GMAA*-ICE as well as Frans Oliehoek, Akshat Kumar and the anonymous reviewers for their helpful comments. Research supported in part by AFOSR MURI project #FA9550-09-1-0538.

Appendix A. Correctness of Feature-Based Compact Bounds

A.1 Proof of Theorem 6

(a) By hypothesis, we have $\tilde{\xi} \in \tilde{\triangle}_t$, such that $\upsilon \geq V^*_{\tilde{M},t}(\tilde{\xi})$. Hence, we obtain successively:

$$\begin{aligned}
\upsilon &\geq V^*_{\tilde{M},t}(\tilde{\xi}) && (48)\\
&= V^*_{\check{M},t}(\xi) && (\tilde{\xi} = F_{\Phi_\xi}(\xi) \text{ by Theorem 5}), \quad (49)\\
&\overset{\text{def}}{=} \max_{\beta \in \Lambda_t} \sum_{s,\theta} \xi(s,\theta) \cdot \beta(s,\theta) && (\text{by definition of } V^*_{\check{M},t}(\xi)), \quad (50)\\
&\geq \max_{\beta \in \Lambda_t} \sum_{\phi_{\dot{s},\dot{\vartheta}} \in \Phi_\xi} \left(\sum_{s,\theta} \phi_{\dot{s},\dot{\vartheta}}(s,\theta) \cdot \xi(s,\theta) \right) \cdot \beta(\dot{s},\dot{\vartheta}) && (\xi \text{ projected onto } \Phi_\xi), \quad (51)\\
&= \max_{\beta \in \Lambda_t} \sum_{\phi_{\dot{s},\dot{\vartheta}} \in \Phi_\xi} \tilde{\xi}(\dot{s},\dot{\vartheta}) \cdot \beta(\dot{s},\dot{\vartheta}), && (52)\\
&\geq \max_{\beta \in \Lambda_t} \sum_{\phi_{s,\vartheta} \in \Phi} \left(\sum_{\phi_{\dot{s},\dot{\vartheta}} \in \Phi_\xi} \phi_{s,\vartheta}(\dot{s},\dot{\vartheta}) \cdot \tilde{\xi}(\dot{s},\dot{\vartheta}) \right) \cdot \beta(s,\vartheta) && (\tilde{\xi} \text{ projected onto } \Phi), \quad (53)\\
&\overset{\text{def}}{=} V^*_{\tilde{M},t}(F_\Phi(\tilde{\xi})), && (54)
\end{aligned}$$

which ends the proof of Theorem 6.a. □

(b) By hypothesis, we have $\tilde{\xi} \in \tilde{\triangle}_t$ such that $V^*_{\tilde{M},t}(\tilde{\xi}) \geq \langle \tilde{\xi}, G_{\Phi_\xi}(\tilde{\beta}) \rangle$ for any arbitrary $\tilde{\beta} \in \mathbb{R}^{|\Phi'|}$. To prove $V^*_{\tilde{M},t}(\tilde{\xi}) \geq \langle \tilde{\xi}, G_{\Phi_\xi}(G_\Phi(\tilde{\beta})) \rangle$ for any arbitrary feature set Φ, we successively show:

$$\begin{aligned}
\langle \tilde{\xi}, G_{\Phi_\xi}(G_\Phi(\tilde{\beta})) \rangle &\overset{\text{def}}{=} \sum_{\phi_{s,\vartheta} \in \Phi_\xi} \tilde{\xi}(s,\vartheta) \sum_{\phi_{\dot{s},\dot{\vartheta}} \in \Phi} \phi_{\dot{s},\dot{\vartheta}}(s,\vartheta) \sum_{\phi_{\ddot{s},\ddot{\vartheta}} \in \Phi'} \phi_{\ddot{s},\ddot{\vartheta}}(\dot{s},\dot{\vartheta}) \cdot \tilde{\beta}(\ddot{s},\ddot{\vartheta}), && (55)\\
&= \sum_{\phi_{s,\vartheta} \in \Phi_\xi} \tilde{\xi}(s,\vartheta) \sum_{\phi_{\ddot{s},\ddot{\vartheta}} \in \Phi'} \tilde{\beta}(\ddot{s},\ddot{\vartheta}) \cdot \left(\sum_{\phi_{\dot{s},\dot{\vartheta}} \in \Phi} \phi_{\ddot{s},\ddot{\vartheta}}(\dot{s},\dot{\vartheta}) \cdot \phi_{\dot{s},\dot{\vartheta}}(s,\vartheta) \right), && (\text{Re-ordering}). \quad (56)
\end{aligned}$$

Before proceeding any further, we need to prove quantity $\phi_{\ddot{s},\ddot{\vartheta}}(s,\vartheta)$ is greater or equal to quantity $\sum_{\phi_{\dot{s},\dot{\vartheta}}\in\Phi}\phi_{\ddot{s},\ddot{\vartheta}}(\dot{s},\dot{\vartheta})\cdot\phi_{\dot{s},\dot{\vartheta}}(s,\vartheta)$ (in the bracket of the last expression above). To this end, we start with the interpretations of each expression. The first expression asks whether state-history pair (s,ϑ) belongs to cluster along with feature $\phi_{\ddot{s},\ddot{\vartheta}}$, an affirmative answer results in value $\phi_{\ddot{s},\ddot{\vartheta}}(s,\vartheta) = 1$ otherwise 0. Let $\phi_{\dot{s}^*,\dot{\vartheta}^*}$ be the feature in Φ whose cluster includes state-history pair (s,ϑ). Then, the second expression asks whether both state-history pairs $(\dot{s}^*,\dot{\vartheta}^*)$ and (s,ϑ) belong to cluster along with feature $\phi_{\ddot{s},\ddot{\vartheta}}\in\Phi'$, an affirmative answer will result in value $\sum_{\phi_{\dot{s},\dot{\vartheta}}\in\Phi}\phi_{\ddot{s},\ddot{\vartheta}}(\dot{s},\dot{\vartheta})\cdot\phi_{\dot{s},\dot{\vartheta}}(s,\vartheta) = 1$ otherwise 0. Clearly, the second expression is a stricter form of the first expression, hence $\phi_{\ddot{s},\ddot{\vartheta}}(s,\vartheta)$ is greater or equal to $\sum_{\phi_{\dot{s},\dot{\vartheta}}\in\Phi}\phi_{\ddot{s},\ddot{\vartheta}}(\dot{s},\dot{\vartheta})\cdot\phi_{\dot{s},\dot{\vartheta}}(s,\vartheta)$. Thus, by replacing $\sum_{\phi_{\dot{s},\dot{\vartheta}}\in\Phi}\phi_{\ddot{s},\ddot{\vartheta}}(\dot{s},\dot{\vartheta})\cdot\phi_{\dot{s},\dot{\vartheta}}(s,\vartheta)$ by $\phi_{\ddot{s},\ddot{\vartheta}}(s,\vartheta)$, we obtain:

$$\begin{aligned}
\langle\tilde{\xi}, G_{\Phi_\xi}(G_\Phi(\tilde{\beta}))\rangle &= \sum_{\phi_{s,\vartheta}\in\Phi_\xi}\tilde{\xi}(s,\vartheta)\sum_{\phi_{\ddot{s},\ddot{\vartheta}}\in\Phi'}\tilde{\beta}(\ddot{s},\ddot{\vartheta})\cdot\left(\sum_{\phi_{\dot{s},\dot{\vartheta}}\in\Phi}\phi_{\ddot{s},\ddot{\vartheta}}(\dot{s},\dot{\vartheta})\cdot\phi_{\dot{s},\dot{\vartheta}}(s,\vartheta)\right) && (57)\\
&\leq \sum_{\phi_{s,\vartheta}\in\Phi_\xi}\tilde{\xi}(s,\vartheta)\sum_{\phi_{\ddot{s},\ddot{\vartheta}}\in\Phi'}\phi_{\ddot{s},\ddot{\vartheta}}(s,\vartheta)\cdot\tilde{\beta}(\ddot{s},\ddot{\vartheta}), && (58)\\
&\overset{\text{def}}{=} \langle\tilde{\xi}, G_{\Phi_\xi}(\tilde{\beta})\rangle && (59)\\
&\leq V^*_{\tilde{M},t}(\tilde{\xi}), && (60)
\end{aligned}$$

which ends the proof Theorem 6.*b*. □

A.2 Proof of Theorem 7

The proof proceeds by induction. Heuristic function $(U_{\tilde{M},t})_{t\in\{0,1,\ldots,T\}}$, initially upper bounds the optimal value function, since it is initialized using the underlying MDP value function.

For the induction step, we assume heuristic function $(U_{\tilde{M},t})_{t\in\{0,1,\ldots,T\}}$ represented using point sets $(\tilde{\Psi}_t)_{t\in\{0,1,\ldots,T\}}$ upper bounds the optimal value function.

Next, we show that, at any arbitrary time step $t\in\{0,1,\ldots,T\}$, heuristic function $U'_{\tilde{M},t}$, after the update of $U_{\tilde{M},t}$ resulting in upper bound $v^{\tilde{\xi}}$ at occupancy state $\tilde{\xi}$, is also an upper bound on $V^*_{\tilde{M},t}$ over the entire compact occupancy space $\tilde{\triangle}_t$. That is,

$$\forall\tilde{\xi}'\in\tilde{\triangle}_t:\qquad U_{\tilde{M},t}(\tilde{\xi}')\geq U'_{\tilde{M},t}(\tilde{\xi}')\geq V^*_{\tilde{M},t}(\tilde{\xi}'). \tag{61}$$

We first show that $\forall\tilde{\xi}'\in\tilde{\triangle}_t\colon U_{\tilde{M},t}(\tilde{\xi}')\geq U'_{\tilde{M},t}(\tilde{\xi}')$. Using the sawtooth interpolation approach[3], we obtain successively:

$$\begin{aligned}
U'_{\tilde{M},t}(\tilde{\xi}') &\overset{\text{def}}{=} \min\left\{v^*_{\tilde{\xi}'}, v^{\tilde{\xi}''}_{\tilde{\xi}'} \mid (\tilde{\xi}''\mapsto v^{\tilde{\xi}''})\in\tilde{\Psi}_t\cup\{\tilde{\xi}\mapsto v^{\tilde{\xi}}\}\right\}, \qquad \text{(see sawtooth definition)} && (62)\\
&= \min\left\{v^{\tilde{\xi}}_{\tilde{\xi}'}, U_{\tilde{M},t}(\tilde{\xi}')\right\}, && (63)\\
&\leq U_{\tilde{M},t}(\tilde{\xi}'), && (64)
\end{aligned}$$

which proves the first part of expression (Eq. 61).

Now, we show $\forall\tilde{\xi}'\in\tilde{\triangle}_t\colon U'_{\tilde{M},t}(\tilde{\xi}')\geq V^*_{\tilde{M},t}(\tilde{\xi}')$. To this end, we distinguish between before and after the update of the $U_{\tilde{M},t}$.

3. Here, we adapted the sawtooth interpolation to replace full occupancy states by compact occupancy states.

Before the update, the following holds:

$$\forall \tilde{\xi}' \in \tilde{\triangle}_t: \qquad U'_{\tilde{M},t}(\tilde{\xi}') \geq V^*_{\tilde{M},t}(\tilde{\xi}'), \tag{65}$$

by the inductive hypothesis.

After the update, we obtain two important results. On the one hand, we have that the resulting value $v^{\tilde{\xi}}$ is an upper bound at $\tilde{\xi}$:

$$v^{\tilde{\xi}} \stackrel{\text{def}}{=} \max_{\tilde{d}_\xi \in \tilde{\boldsymbol{A}}} \tilde{\boldsymbol{R}}(\tilde{\xi}, \tilde{d}_\xi) + U_{\tilde{M},t+1}(\tilde{\boldsymbol{P}}(\tilde{\xi}, \tilde{d}_\xi)), \tag{66}$$

$$\geq \max_{\tilde{d}_\xi \in \tilde{\boldsymbol{A}}} \tilde{\boldsymbol{R}}(\tilde{\xi}, \tilde{d}_\xi) + V^*_{\tilde{M},t+1}(\tilde{\boldsymbol{P}}(\tilde{\xi}, \tilde{d}_\xi)), \tag{67}$$

$$\stackrel{\text{def}}{=} V^*_{\tilde{M},t}(\tilde{\xi}). \tag{68}$$

On the other hand, we show that from $v^{\tilde{\xi}}$ one can extrapolate an upper bound value $v^{\tilde{\xi}}_{\tilde{\xi}'}$ for any other compact occupancy state $\tilde{\xi}'$. This is mainly thanks to Theorem 6.*a*, in which we demonstrate: if expression $V^*_{\tilde{M},t}(\tilde{\xi}) \leq v^{\tilde{\xi}'}$ holds, then for any arbitrary $\tilde{\xi}' \in \tilde{\triangle}_t$, expression $V^*_{\tilde{M},t}(F_{\Phi_{\xi'}}(\tilde{\xi})) \leq v^{\tilde{\xi}}$ holds as well. The sawtooth interpolation method concludes this argument as follows:

$$v^{\tilde{\xi}'}_{\tilde{\xi}''} \stackrel{\text{def}}{=} v^*_{\tilde{\xi}'} + (v^{\tilde{\xi}'} - v^*_{\tilde{\xi}}) \cdot D(F_{\Phi_{\xi'}}(\tilde{\xi}), \tilde{\xi}'), \tag{69}$$

$$\geq V^*_{\tilde{M},t}(\tilde{\xi}'). \tag{70}$$

In fact, the sawtooth interpolation can always generate an upper-bound for one compact occupancy state from the upper bound of any compact occupancy state as long as both are expressed into the same feature set. □

A.3 Proof of Theorem 8

The proof proceeds by induction as well. Heuristic function $(L_{\tilde{M},t})_{t\in\{0,1,\ldots,T\}}$ initially lower bounds the optimal value function since we initialize it using the value function associated with the worst separable joint policy. The one that prescribes the agents the joint action that yields the minimum reward, and this over all time steps, *i.e.,* $L_{\tilde{M},t}(\cdot) \stackrel{\text{def}}{=} (T-t)\min_{s,a} r^a(s)$, for all $t \in \{0, 1, \ldots, T\}$.

For the induction step, we assume heuristic function $(L_{\tilde{M},t})_{t\in\{0,1,\ldots,T\}}$ represented using compact vector sets $(\tilde{\Lambda}_t)_{t\in\{0,1,\ldots,T\}}$ lower bounds the optimal value function. Next, we show that for any arbitrary time step $t \in \{0, 1, \ldots, T\}$, heuristic function $L'_{\tilde{M},t}$, which results from the update of lower bound $L_{\tilde{M},t}$ and produces compact vector $\tilde{\beta}_\xi$ along feature set Φ_ξ, is also a lower bound on $V^*_{\tilde{M},t}$ over the entire compact occupancy space $\tilde{\triangle}_t$. That is,

$$\forall \tilde{\xi}' \in \tilde{\triangle}_t: \qquad L_{\tilde{M},t}(\tilde{\xi}') \leq L'_{\tilde{M},t}(\tilde{\xi}') \leq V^*_{\tilde{M},t}(\tilde{\xi}'). \tag{71}$$

We first show that $\forall \tilde{\xi}' \in \tilde{\triangle}_t$: $L_{\tilde{M},t}(\tilde{\xi}') \leq L'_{\tilde{M},t}(\tilde{\xi}')$. In fact,

$$L_{\tilde{M},t}(\tilde{\xi}') \stackrel{\text{def}}{=} \max_{\tilde{\beta}\in\tilde{\Lambda}_t} \langle \tilde{\xi}', G_{\Phi_{\xi'}}(\tilde{\beta})\rangle, \tag{72}$$

$$\leq \max_{\tilde{\beta}\in\tilde{\Lambda}_t\cup\{\tilde{\beta}_\xi\}} \langle \tilde{\xi}', G_{\Phi_{\xi'}}(\tilde{\beta})\rangle, \tag{73}$$

$$\stackrel{\text{def}}{=} L'_{\tilde{M},t}(\tilde{\xi}'), \tag{74}$$

which proves the first part.

Now, we show $\forall \tilde{\xi}' \in \tilde{\triangle}_t$: $L'_{\tilde{M},t}(\tilde{\xi}') \leq V^*_{\tilde{M},t}(\tilde{\xi}')$. To this end, we distinguish between before and after the update of $L_{\tilde{M},t}$.

Before the update, by the induction hypothesis, we have:

$$\forall \tilde{\xi}' \in \tilde{\triangle}_t: \qquad L'_{\tilde{M},t}(\tilde{\xi}') \leq V^*_{\tilde{M},t}(\tilde{\xi}'). \tag{75}$$

After the update, we obtain: $\tilde{\xi}' \in \tilde{\triangle}_t$,

$$\begin{aligned}
V^*_{\tilde{M},t}(\tilde{\xi}') &\stackrel{\text{def}}{=} \max_{\tilde{d}_{\xi'} \in \tilde{\boldsymbol{A}}} \tilde{\boldsymbol{R}}(\tilde{\xi}', \tilde{d}_{\xi'}) + V^*_{\tilde{M},t+1}(\tilde{\boldsymbol{P}}(\tilde{\xi}', \tilde{d}_{\xi'})), && (76)\\
&= \max_{\tilde{d}_{\xi'} \in \tilde{\boldsymbol{A}}} \langle \tilde{\xi}', r^{\tilde{d}_{\xi'}} \rangle + \max_{\tilde{\alpha} \in \tilde{\Lambda}^*_{t+1}} \langle \tilde{\xi}' p^{\tilde{d}_{\xi'}}, G_{\Phi_{\tilde{\xi}' p^{\tilde{d}_{\xi'}}}}(\tilde{\alpha}) \rangle, && (77)\\
&= \max_{\tilde{d}_{\xi'} \in \tilde{\boldsymbol{A}}, \tilde{\alpha} \in \tilde{\Lambda}^*_{t+1}} \langle \tilde{\xi}', r^{\tilde{d}_{\xi'}} + G_{\Phi_{\tilde{\xi}' p^{\tilde{d}_{\xi'}}}}(\tilde{\alpha}) \cdot (p^{\tilde{d}_{\xi'}})^\top \rangle, && \text{(re-arranging terms)} \quad (78)\\
&\geq \max_{\tilde{d}_{\xi'} \in \tilde{\boldsymbol{A}}, \tilde{\alpha} \in \tilde{\Lambda}_{t+1}} \langle \tilde{\xi}', r^{\tilde{d}_{\xi'}} + G_{\Phi_{\tilde{\xi}' p^{\tilde{d}_{\xi'}}}}(\tilde{\alpha}) \cdot (p^{\tilde{d}_{\xi'}})^\top \rangle, && \text{(replace } \tilde{\Lambda}^*_{t+1} \text{ by } \tilde{\Lambda}_{t+1}\text{)} \quad (79)\\
&\geq \max_{\tilde{d}_{\xi} \in \tilde{\boldsymbol{A}}, \tilde{\alpha} \in \tilde{\Lambda}_{t+1}} \langle \tilde{\xi}', G_{\Phi_{\tilde{\xi}'}}(r^{\tilde{d}_{\xi}} + G_{\Phi_{\tilde{\xi} p^{\tilde{d}_{\xi}}}}(\tilde{\alpha}) \cdot (p^{\tilde{d}_{\xi}})^\top)) \rangle, && \text{(Lemma 6)} \quad (80)\\
&\geq \langle \tilde{\xi}', G_{\Phi_{\tilde{\xi}'}}(\widetilde{backup}(\tilde{\xi}, \tilde{\Lambda}_{t+1})) \rangle, && \text{(retain one element).} \quad (81)
\end{aligned}$$

Merging together arguments before and after the update, *i.e.,* $L'_{\tilde{M},t}(\tilde{\xi}') \leq V^*_{\tilde{M},t}(\tilde{\xi}')$, we prove the second part of the proof. This ends the proof. □

Appendix B. Subroutines

This section gives subroutines that are required to compute feature-based compact occupancy states using either local or truncation probabilistic equivalence relations (Algorithm 3).

Algorithm 3: Compact feature-based occupancy state through LPE.

```
function Compact-LPE(ξ_t)
    𝒮 ← {(s, θ) ∈ S × Θ_t : ξ_t(s, θ) > 0}
    foreach (s, θ) ∈ 𝒮 do
        𝒮 ← 𝒮\{(s, θ)} and ξ̃_t(s, θ) ← ξ_t(s, θ)
        foreach (ṡ, θ') ∈ 𝒮 do
            if AreStateJointHistoryPairsLPE((s, θ), (ṡ, θ'), ξ_t) then
                𝒮 ← 𝒮\{(ṡ, θ')} and ξ̃_t(s, θ) ← ξ̃_t(s, θ) + ξ_t(ṡ, θ')
    return ξ̃_t

function Compact-TPE(ξ_t)
    m_{ξ_t} ← getTruncationParam(ξ_t) and 𝒮 ← S × Θ_t(ξ)
    foreach (s, θ) ∈ 𝒮 do
        𝒮 ← 𝒮\{(s, θ)} and ξ̃_t(s, θ) ← ξ_t(s, θ)
        foreach (ṡ, θ') ∈ 𝒮 do
            if AreStateJointHistoryPairsTPE((s, θ), (ṡ, θ'), m_{ξ_t}) then
                𝒮 ← 𝒮\{(ṡ, θ')} and ξ̃_t(s, θ) ← ξ̃_t(s, θ) + ξ_t(ṡ, θ')
    return ξ̃_t
```

Algorithm 4: Subroutines for compact feature-based occupancy state through LPE and TPE.

```
function ArePrivateHistoriesLPE(θ_t^i, θ'_t^i, ξ_t)
    foreach θ_t^{-i} ∈ Θ_t^{-i}(ξ_t) and s ∈ S do
        if Pr(s, θ_t^{-i}|ξ_0, θ_t) ≠ Pr(s, θ_t^{-i}|ξ_0, θ'_t) then return False
    return True

function AreStateJointHistoryPairsLPE((s, ⟨θ_t^i⟩_{i∈I}), (ṡ, ⟨θ'_t^i⟩_{i∈I}), ξ_t)
    if s ≠ ṡ then return False
    foreach i ∈ I do
        if ¬ArePrivateHistoryLPE(θ_t^i, θ'_t^i, ξ_t) then return False
    return True

function getTruncationParam(ξ_t)
    m ← 0
    foreach i ∈ I do
        foreach θ_t^i, θ'_t^i ∈ Θ_t^i(ξ_t) do
            if Suffix(θ_t^i, m) = Suffix(θ'_t^i, m) then
                if ¬ArePrivateHistoriesLPE(θ_t^i, θ'_t^i, ξ_t) then m ← m + 1;
    return m

function ArePrivateHistoriesTPE(θ_t^i, θ'_t^i, m_{ξ_t})
    return Suffix(θ_t^i, m_{ξ_t}) = Suffix(θ'_t^i, m_{ξ_t})

function AreStateJointHistoryPairsTPE((s, ⟨θ_t^i⟩_{i∈I}), (ṡ, ⟨θ'_t^i⟩_{i∈I}), m_{ξ_t})
    if s ≠ ṡ then return False
    foreach i ∈ I do
        if ¬ArePrivateHistoryLPE(θ_t^i, θ'_t^i, m_{ξ_t}) then return False
    return True
```

References

Amato, C., Bernstein, D. S., & Zilberstein, S. (2010). Optimizing fixed-size stochastic controllers for POMDPs and decentralized POMDPs. *Journal of Autonomous Agents and Multi-Agent Systems*, *21*(3), 293–320.

Amato, C., Chowdhary, G., Geramifard, A., Ure, N. K., & Kochenderfer, M. J. (2013). Decentralized control of partially observable Markov decision processes. In *54th IEEE Conference on Decision and Control*.

Amato, C., Dibangoye, J. S., & Zilberstein, S. (2009). Incremental policy generation for finite-horizon DEC-POMDPs. In *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling*.

Amato, C., Konidaris, G. D., Anders, A., Cruz, G., How, J. P., & Kaelbling, L. P. (2015). Policy search for multi-robot coordination under uncertainty. In *Proceedings of the Robotics: Science and Systems Conference*.

Amato, C., Konidaris, G. D., & Kaelbling, L. P. (2014). Planning with macro-actions in decentralized POMDPs. In *Proceedings of the Thirteenth International Conference on Autonomous Agents and Multiagent Systems*.

Aras, R., & Dutech, A. (2010). An investigation into mathematical programming for finite horizon decentralized POMDPs. *Journal of Artificial Intelligence Research*, *37*, 329–396.

Banerjee, B., Lyle, J., Kraemer, L., & Yellamraju, R. (2012). Sample bounded distributed reinforcement learning for decentralized POMDPs. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pp. 1256–1262, Toronto, Canada.

Barto, A. G., Bradtke, S. J., & Singh, S. P. (1995). Learning to act using real-time dynamic programming. *Artificial Intelligence*, *72*(1-2), 81–138.

Becker, R., Zilberstein, S., Lesser, V. R., & Goldman, C. V. (2004). Solving transition independent decentralized Markov decision processes. *Journal of Artificial Intelligence Research*, *22*, 423–455.

Bellman, R. E. (1957). *Dynamic Programming*. Dover Publications, Incorporated.

Bernstein, D. S., Givan, R., Immerman, N., & Zilberstein, S. (2002). The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, *27*(4).

Boularias, A., & Chaib-draa, B. (2008). Exact dynamic programming for decentralized POMDPs with lossless policy compression. In *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling*, pp. 20–27.

Canu, A., & Mouaddib, A.-I. (2011). Collective decision under partial observability - a dynamic local interaction model. In *IJCCI (ECTA-FCTA)*, pp. 146–155.

Carlin, A., & Zilberstein, S. (2008). Value-based observation compression for DEC-POMDPs. In *Proceedings of the Seventh International Conference on Autonomous Agents and Multiagent Systems*.

De Farias, D. P., & Van Roy, B. (2003). The linear programming approach to approximate dynamic programming. *Operations Research*, *51*(6), 850–865.

Dibangoye, J. S., Amato, C., Buffet, O., & Charpillet, F. (2014). Exploiting separability in multiagent planning with continuous-state MDPs. In *Proceedings of the Thirteenth International Conference on Autonomous Agents and Multiagent Systems*, pp. 1281–1288.

Dibangoye, J. S., Amato, C., Buffet, O., & Charpillet, F. (2015). Exploiting separability in multiagent planning with continuous-state MDPs (extended abstract). In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pp. 4254–4260.

Dibangoye, J. S., Amato, C., & Doniec, A. (2012). Scaling up decentralized MDPs through heuristic search. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, pp. 217–226.

Dibangoye, J. S., Amato, C., Doniec, A., & Charpillet, F. (2013). Producing efficient error-bounded solutions for transition independent decentralized MDPs. In *Proceedings of the Twelfth International Conference on Autonomous Agents and Multiagent Systems*, pp. 539–546.

Dibangoye, J. S., Buffet, O., & Simonin, O. (2015). Structural results for cooperative decentralized control models. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pp. 46–52.

Dibangoye, J. S., Mouaddib, A.-I., & Chaib-draa, B. (2009). Point-based incremental pruning heuristic for solving finite-horizon DEC-POMDPs. In *Proceedings of the Eighth International Conference on Autonomous Agents and Multiagent Systems*, pp. 569–576.

Dibangoye, J. S., Mouaddib, A.-I., & Chaib-draa, B. (2011). Toward error-bounded algorithms for infinite-horizon Dec-POMDPs. In *Proceedings of the Tenth International Conference on Autonomous Agents and Multiagent Systems*, pp. 947–954.

Dibangoye, J. S., Shani, G., Chaib-Draa, B., & Mouaddib, A.-I. (2009). Topological order planner for POMDPs. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, pp. 1684–1689.

Dibangoye, J. S., Amato, C., Buffet, O., & Charpillet, F. (2013). Optimally solving Dec-POMDPs as continuous-state MDPs. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Dibangoye, J. S., Buffet, O., & Charpillet, F. (2014). Error-bounded approximations for infinite-horizon discounted decentralized POMDPs. In *Proceedings of the Twenty-Fourth European Conference on Machine Learning*, pp. 338–353.

Dibangoye, J. S., Chaib-draa, B., & Mouaddib, A.-I. (2008). A novel prioritization technique for solving Markov decision processes. In *Proceedings of the 21th International Conference of the Florida Artificial Intelligence Research Society*, pp. 537–542.

Grizzle, J. W., Marcus, S. I., & Hsu, K. (1981). Decentralized control of a multiaccess broadcast network. In *20th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes*, Vol. 20, pp. 390–391.

Hansen, E. A., Bernstein, D. S., & Zilberstein, S. (2004). Dynamic programming for partially observable stochastic games. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, pp. 709–715.

Hansen, E. A., & Zilberstein, S. (2001). LAO*: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, *129*(1-2), 35–62.

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Systems Science and Cybernetics*, *4*(2), 100–107.

Hauskrecht, M. (2000). Value-function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, *13*, 33–94.

Howard, R. A. (1960). *Dynamic Programming and Markov Processes*. The M.I.T. Press.

Jain, M., Taylor, M. E., Tambe, M., & Yokoo, M. (2009). DCOPs meet the real world: Exploring unknown reward matrices with applications to mobile sensor networks. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, pp. 181–186.

Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, *101*(1-2), 99–134.

Korf, R. E. (1990). Real-time heuristic search. *Artificial Intelligence*, *42*(2-3), 189–211.

Kumar, A., & Zilberstein, S. (2009). Constraint-based dynamic programming for decentralized POMDPs with structured interactions. In *Proceedings of the Eighth International Conference on Autonomous Agents and Multiagent Systems*, pp. 561–568.

Kumar, A., & Zilberstein, S. (2010). Point-based backup for decentralized POMDPs: complexity and new algorithms. In *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems*, pp. 1315–1322.

Nair, R., Tambe, M., Yokoo, M., Pynadath, D. V., & Marsella, S. (2003). Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pp. 705–711.

Nair, R., Varakantham, P., Tambe, M., & Yokoo, M. (2005). Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, pp. 133–139.

Oliehoek, F. A. (2012). Decentralized POMDPs. In Wiering, M., & van Otterlo, M. (Eds.), *Reinforcement Learning: State of the Art*, Vol. 12, pp. 471–503. Springer Berlin Heidelberg, Berlin, Germany.

Oliehoek, F. A. (2013). Sufficient plan-time statistics for decentralized POMDPs. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Oliehoek, F. A., Spaan, M. T. J., Amato, C., & Whiteson, S. (2013). Incremental clustering and expansion for faster optimal planning in Dec-POMDPs. *Journal of Artificial Intelligence Research*, *46*, 449–509.

Oliehoek, F. A., Spaan, M. T. J., & Vlassis, N. A. (2008). Optimal and approximate Q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research*, *32*, 289–353.

Oliehoek, F. A., & Spaan, M. T. (2012). Tree-based solution methods for multiagent POMDPs with delayed communication. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Oliehoek, F. A., Whiteson, S., & Spaan, M. T. J. (2009). Lossless clustering of histories in decentralized POMDPs. In *Proceedings of the Eighth International Conference on Autonomous Agents and Multiagent Systems*, pp. 577–584.

Ooi, J. M., & Wornell, G. W. (1996). Decentralized control of a multiple access broadcast channel: Performance bounds. In *Proc. of the 35th IEEE Conference on Decision and Control*, Vol. 1, pp. 293–298. IEEE.

Pajarinen, J., Hottinen, A., & Peltonen, J. (2013). Optimizing spatial and temporal reuse in wireless networks by decentralized partially observable Markov decision processes. *IEEE Transactions on Mobile Computing*, *13*(4). Preprint.

Paquet, S., Chaib-draa, B., Dallaire, P., & Bergeron, D. (2010). Task allocation learning in a multiagent environment: Application to the robocuprescue simulation. *Multiagent and Grid Systems*, *6*(4), 293–314.

Pineau, J., Gordon, G. J., & Thrun, S. (2006). Anytime point-based approximations for large POMDPs. *Journal of Artificial Intelligence Research*, *27*, 335–380.

Powell, W. B. (2007). *Approximate Dynamic Programming: Solving the Curses of Dimensionality (Wiley Series in Probability and Statistics).* Wiley-Interscience.

Puterman, M. L. (1994). *Markov Decision Processes, Discrete Stochastic Dynamic Programming*. Wiley-Interscience, Hoboken, New Jersey.

Roy, N., Gordon, G. J., & Thrun, S. (2005). Finding approximate POMDP solutions through belief compression. *Journal of Artificial Intelligence Research*, *23*, 1–40.

Seuken, S., & Zilberstein, S. (2007). Improved memory-bounded dynamic programming for DEC-POMDPs. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*.

Shani, G., Pineau, J., & Kaplow, R. (2013). A survey of point-based POMDP solvers. *Journal of Autonomous Agents and Multi-Agent Systems*, *27*(1), 1–51.

Smallwood, R. D., & Sondik, E. J. (1973). The optimal control of partially observable Markov decision processes over a finite horizon. *Operations Research*, *21*(5), 1071–1088.

Smith, T. (2007). *Probabilistic Planning for Robotic Exploration*. Ph.D. thesis, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Smith, T., & Simmons, R. (2004). Heuristic search value iteration for POMDPs. In *Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence*, pp. 520–527, Arlington, Virginia, United States.

Smith, T., & Simmons, R. G. (2006). Focused real-time dynamic programming for MDPs: Squeezing more out of a heuristic. In *Proceedings of the Twenty-First AAAI Conference on Artificial Intelligence*, pp. 1227–1232.

Spaan, M. T. J., Oliehoek, F. A., & Amato, C. (2011). Scaling up optimal heuristic search in Dec-POMDPs via incremental expansion. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, pp. 2027–2032.

Szer, D., Charpillet, F., & Zilberstein, S. (2005). MAA*: A heuristic search algorithm for solving decentralized POMDPs. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pp. 568–576.

Tsitsiklis, J. N., & van Roy, B. (1996). Feature-based methods for large scale dynamic programming. *Machine Learning*, *22*(1-3), 59–94.

Velagapudi, P., Varakantham, P., Sycara, K., & Scerri, P. (2011). Distributed model shaping for scaling to decentralized POMDPs with hundreds of agents. In *Proceedings of the Tenth International Conference on Autonomous Agents and Multiagent Systems*, pp. 955–962.

Winstein, K., & Balakrishnan, H. (2013). TCP ex Machina: Computer-generated congestion control. In *SIGCOMM*, Hong Kong.

Wu, F., Zilberstein, S., & Chen, X. (2010). Point-based policy generation for decentralized POMDPs. In *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems*, pp. 1307–1314.

Wu, F., Zilberstein, S., & Chen, X. (2011). Online planning for multi-agent systems with bounded communication. *Artificial Intelligence*, *175*(2), 487–511.

Zilberstein, S., Washington, R., Bernstein, D. S., & Mouaddib, A.-I. (2002). Decision-theoretic control of planetary rovers. In *Revised Papers from the International Seminar on Advances in Plan-Based Control of Robotic Agents*, pp. 270–289, London, UK. Springer-Verlag.