EFP and PG-EFP: Epistemic Forward Search Planners in Multi-Agent Domains

Tiep Le
Computer Science
New Mexico State University
Las Cruces, NM 88003, USA
tile@cs.nmsu.edu

Francesco Fabiano
Computer Science
New Mexico State University
Las Cruces, NM 88003, USA
ffabiano@cs.nmsu.edu

Tran Cao Son
Computer Science
New Mexico State
Las Cruces, NM 88003, USA
tson@cs.nmsu.edu

Enrico Pontelli
Computer Science
New Mexico State University
Las Cruces, NM 88003, USA
epontell@cs.nmsu.edu

Abstract

This paper presents two prototypical epistemic forward planners, called EFP and PG-EFP, for generating plans in multi-agent environments. These planners differ from recently developed epistemic planners in that they can deal with unlimited nested beliefs, common knowledge, and capable of generating plans with both knowledge and belief goals. EFP is simply a breadth first search planner while PG-EFP is a heuristic search based system. To generate heuristics in PG-EFP, the paper introduces the notion of an *epistemic planning graph*. The paper includes an evaluation of the planners using benchmarks collected from the literature and discusses the issues that affect their scalability and efficiency, thus identifies potentially directions for future work. It also includes experimental evaluation that proves the usefulness of epistemic planning graphs.

Motivation

Epistemic planning in multi-agent domains has recently gained momentum in several research communities. Its complexity has been studied in (Aucher and Bolander 2013; Bolander, Jensen, and Schwarzentruber 2015; Charrier, Maubert, and Schwarzentruber 2016). Studies of epistemic planning can be found in (Bolander and Andersen 2011; Crosby, Jonsson, and Rovatsos 2014; Engesser et al. 2017; van der Hoek and Wooldridge 2002; Huang et al. 2017; Löwe, Pacuit, and Witzel 2011; Muise et al. 2015; Kominis and Geffner 2015; 2017; van Eijck 2004; Wan et al. 2015). Due to its complexity, the majority of search based epistemic planners (e.g., (Crosby, Jonsson, and Rovatsos 2014; Engesser et al. 2017; Kominis and Geffner 2015; 2017; Huang et al. 2017; Muise et al. 2015; Wan et al. 2015)) impose certain restrictions, such as the finiteness of the levels of nested beliefs. In some approaches (e.g., (Bolander and Andersen 2011; van der Hoek and Wooldridge 2002; Löwe, Pacuit, and Witzel 2011)), the initial epistemic state is assumed to be known and finite or recursively enumerable. Issues and research directions related to epistemic planning have been summarized nicely in the report of the recent Dagstuhl's meeting (Baral et al. 2017).

In this paper, we describe an epistemic forward search planner, EFP, derived from recent research developments in reasoning about actions in multi-agent domains (Baral et al. 2012; Son et al. 2014). The primary goal is to be able to solve planning problems similar to that given in Example 1 below, a slightly modified version of the first example in (Baral et al. 2012) and given in (Baral et al. 2015).

Example 1 (The Coin in the Box Domain) *Three agents A, B, and C are in a room. In the middle of the room there is a box containing a coin. It is common knowledge that:*

- *Nobody knows which face of the coin is showing;*
- *The box is locked and one needs a key to open it; agent A is the only one with a key to the box;*
- *To determine the face of the coin, one can peek into the box if the box is open;*
- *An agent, observing another agent peeking into the box, will be able to conclude that the agent who peeked knows which face of the coin is showing—but without knowing which is showing himself;*
- *Distracting an agent causes s/he to not look at the box;*
- *Signaling an agent causes the agent to look at the box;*
- *Announcing that the coin is showing heads or tails will cause everyone to know this fact.*
- *A and C are looking, while B is not looking at the box.*

We assume that in reality the coin lies tail up. Agent A wishes to know which face of the coin is up, and s/he would like agent B to become aware of the fact that A knows the state of the coin, while keeping C in the dark.

It is easy to see that agent A could achieve such goal by: **(a)** *distracting C, thus keeping him from looking at the box;* **(b)** *signaling B to look at the box;* **(c)** *opening the box; and* **(d)** *peeking into the box.*

To the best of our knowledge, there is no system that can deal with this example. We observe that this problem cannot be expressed by the formulation given in several epistemic planners (e.g., (Kominis and Geffner 2015; Huang et al. 2017; Muise et al. 2015; Wan et al. 2015)) since it requires the representation and reasoning about common knowledge.

The key contributions of this paper are: **(i)** two epistemic planners EFP and PG-EFP that can work with common knowledge; **(ii)** the notion of epistemic planning graph, which generalizes the notion of planning graph in the literature to the case of epistemic planning; **(iii)** an experimental evaluation showing that heuristic derived from epistemic

planning graphs is useful; and **(iv)** an experimental evaluation showing that EFP and PG-EFP can be potentially useful as a tool for epistemic planning.

The paper is organized as follows. The next section reviews the background of epistemic planning and a specification language for epistemic planning. Next, we provide a description of the architecture and components of EFP and PG-EFP. This is followed by the notion of epistemic planning graph and the algorithm for computing epistemic planning graphs. The paper then presents an experimental evaluation of EFP and PG-EFP, a comparison with some other planning systems and a discussion of their strengths and weaknesses. The source code of EFP and PG-EFP and input files to reproduce results in our experiments is downloadable from `https://github.com/tiep/EpistemicPlanning`. The Supplementary Document can be downloaded from `https://www.cs.nmsu.edu/~tson/ICAPS2018/supplementary.pdf`.

Background

Epistemic Planning

An epistemic planning problem (Bolander and Andersen 2011) is defined over a multi-agent epistemic logic language $\mathcal{L}(\mathcal{F}, \mathcal{AG})$, with a set of agents $\mathcal{AG}$ and a set of *fluents* $\mathcal{F}$. $\mathcal{L}(\mathcal{F}, \mathcal{AG})$ allows formulae constructed using the BNF

$$\varphi \stackrel{def}{=} \top \mid \bot \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid K_i\varphi \mid C_X\varphi$$

where $p \in \mathcal{F}$, $i \in \mathcal{AG}$, and $X \subseteq \mathcal{AG}$. The intended interpretation of $K_i\varphi$ is "agent i knows φ" and $C_X\varphi$ is "the agents in X share the knowledge about φ", i.e., every one in the group knows that φ is true. A fluent literal is either f or $\neg f$ for some $f \in \mathcal{F}$. A consistent set of fluent literals I is an interpretation of $\mathcal{F}$; I is a complete interpretation of $\mathcal{F}$ if for each $f \in F$, $\{f, \neg f\} \cap I \neq \emptyset$. $\mathcal{F}^I$ denotes the set of complete interpretations of $\mathcal{F}$. The semantics of $\mathcal{L}(\mathcal{F}, \mathcal{AG})$ is defined using *Kripke structures* (Fagin et al. 1995), also referred to as *epistemic models.*

Definition 1 *An* epistemic model *(or* e-model*) of the language $\mathcal{L}(\mathcal{F}, \mathcal{AG})$ is a triple $\mathcal{M} = (W, R, \pi)$, where:* (i) *$W$ is a finite set of worlds (the domain);* (ii) *$R : \mathcal{AG} \to 2^{W \times W}$ assigns an accessibility relation $R(i) = R_i$ to each agent $i \in \mathcal{AG}$.*[1] (iii) *$\pi : W \to \mathcal{F}^I$ assigns a complete interpretation of $\mathcal{F}$ to each world. A* pointed epistemic model *(or* pe-model*) is a pair $(\mathcal{M}, w)$, where $\mathcal{M} = (W, R, \pi)$ is an epistemic model and $w \in W$.*

We write wR_iv for $(w, v) \in R(i)$. The satisfaction of a formula of the language $\mathcal{L}(\mathcal{F}, \mathcal{AG})$ is expressed with respect to pe-models. Given a pointed epistemic model $(\mathcal{M}, w)$ with $\mathcal{M} = (M, R, \pi)$ and a formula φ, the satisfaction relation between φ and $(\mathcal{M}, w)$ is defined as follows: **(i)** $(\mathcal{M}, w) \models p$ iff p is true in w; **(ii)** $(\mathcal{M}, w) \models \neg\varphi$ iff $(\mathcal{M}, w) \not\models \varphi$; **(iii)** $(\mathcal{M}, w) \models \varphi_1 \wedge \varphi_2$ iff $(\mathcal{M}, w) \models \varphi_1$ and $(\mathcal{M}, w) \models \varphi_2$; **(iv)** $(\mathcal{M}, w) \models K_i\varphi$ if for all $v \in W$, if wR_iv then $(\mathcal{M}, v) \models \varphi$; and **(v)** $(\mathcal{M}, w) \models C_X\varphi$ if for all $v \in W$, if $w(\bigcup_{j \in X} R_j)^*v$ then $(\mathcal{M}, v) \models \varphi$ where $(\bigcup_{j \in X} R_j)^*$ is the transitive closure of $\bigcup_{j \in X} R_j$.

$\mathcal{M} \models \varphi$ if $(\mathcal{M}, w) \models \varphi$ for each $w \in W$.

An *epistemic state* (or *e-state*) is a pair $(\mathcal{M}, W_d)$ where $\mathcal{M} = (M, R, \pi)$ is an epistemic model and $W_d \subseteq W$. A truth value of a formula φ with respect to an epistemic state $(\mathcal{M}, W_d)$ is defined by

$$(\mathcal{M}, W_d) \models \varphi \qquad \text{iff} \qquad \forall w \in W_d.[(\mathcal{M}, w) \models \varphi]$$

Epistemic planning relies on the notion of an *event model* (also called *update model* in the literature) for modeling changes to epistemic states. We use the formalization of update model introduced in (Baltag and Moss 2004; van Benthem, van Eijck, and Kooi 2006). Let us start with some preliminary definitions. An $\mathcal{L}(\mathcal{F}, \mathcal{AG})$-substitution is a set $\{p_1 \to \varphi_1, \ldots, p_k \to \varphi_k\}$, where each p_i is a distinct proposition in $\mathcal{F}$ and each $\varphi_i \in \mathcal{L}(\mathcal{F}, \mathcal{AG})$. We will implicitly assume that for each $p \in \mathcal{F} \setminus \{p_1, \ldots, p_k\}$, the substitution contains $p \to p$. $SUB_{\mathcal{F},\mathcal{AG}}$ denotes the set of all $\mathcal{L}(\mathcal{F}, \mathcal{AG})$-substitutions.

Definition 2 (Event Model) *Given a set $\mathcal{AG}$ of n agents, an* event model *$\mathbf{\Sigma}$ is a tuple $\langle E, Q, pre, sub\rangle$ where (i) E is a set, whose elements are called* events*; (ii) $Q : \mathcal{AG} \to 2^{E \times E}$ assigns an accessibility relation to each agent $i \in \mathcal{AG}$; (iii) $pre : E \to \mathcal{L}(\mathcal{F}, \mathcal{AG})$ is a function mapping each event $e \in E$ to a formula in $\mathcal{L}(\mathcal{F}, \mathcal{AG})$; and (iv) $sub : E \to SUB_{\mathcal{F},\mathcal{AG}}$ is a function mapping each event $e \in E$ to a substitution in $SUB_{\mathcal{F},\mathcal{AG}}$.*

A pair $(\mathcal{E}, E_d)$, consisting of an event model $\mathcal{E} = (E, Q, pre, sub)$ and a non-empty set of designated events $E_d \subseteq E$, is called an epistemic action*.*

Given an epistemic action $(\mathcal{E}, E_d)$ and an epistemic state $(\mathcal{M}, W_d)$, we say that $(\mathcal{E}, E_d)$ is executable in $(\mathcal{M}, W_d)$ if, for each $w \in W_d$, there exists at least one $e \in E_d$ such that $(\mathcal{M}, w) \models pre(e)$. The execution of $(\mathcal{E}, E_d)$ in $(\mathcal{M}, W_d)$ results in an epistemic state $(\mathcal{M}, W_d) \otimes (\mathcal{E}, E_d) = ((W', R', \pi'), W'_d)$ where

- $W' = \{(w, e) \in W \times E \mid (\mathcal{M}, w) \models pre(e)\}$
- $R'_i = \{((w, e), (v, f)) \in W' \times W' \mid wR_iv \wedge eQ_if\}$
- For each $(w, e) \in W'$ and $p \in \mathcal{F}$, $\pi'((w, e))(p)$ is true iff $p \to \varphi \in sub(e)$ and $(\mathcal{M}, w) \models \varphi$
- $W'_d = \{(w, e) \in W' \mid w \in W_d \text{ and } e \in E_d\}$

An *epistemic planning domain* is then defined as a state-transition system $\Sigma = (S, A, \gamma)$, where S is a set of epistemic states of $\mathcal{L}(\mathcal{F}, \mathcal{AG})$, A is a finite set of epistemic actions of $\mathcal{L}(\mathcal{F}, \mathcal{AG})$, and $\gamma(s, a) = s \otimes a$ if $s \otimes a$ is defined and $\otimes$ is the above operation. An *epistemic planning problem* is a triple (Σ, s_0, ϕ_g) where $\Sigma = (S, A, \gamma)$ is an epistemic planning domain on $(\mathcal{F}, \mathcal{AG})$, $s_0 \in S$ is the initial state, and ϕ_g is a formula in $\mathcal{L}(\mathcal{F}, \mathcal{AG})$. An action sequence $a_1, \ldots, a_n$ where $s_0 \otimes a_1 \otimes \ldots \otimes a_n \models \phi_g$ is a *solution* of the problem.

Epistemic Planning Problem Specification

The above formalization of epistemic planning problem leaves an open question of *how to compactly*[2] *specify an*

[1] (Bolander and Andersen 2011) assumes that all accessibility relations are equivalence relations; we will relax this, as it is too restrictive to effectively represent beliefs.

[2] By a "compact specification" we mean a specification that *does not explicitly list* all possible e-states of the problem.

epistemic planning problem?, i.e., how to compactly specify an epistemic planning domain $\Sigma = (S, A, \gamma)$ on $(\mathcal{F}, \mathcal{AG})$ and how to specify s_0. In this paper, we will use finitary **S5**-theory (Son et al. 2014) and the language $m\mathcal{A}$ (Baral et al. 2012) to specify the initial state and a planning domain, respectively. As our focus in this paper is about the development of the planner, we defer to the Supplementary Document of the paper the precise definitions of $m\mathcal{A}$ and of finitary **S5**-theories. We will only briefly review their features needed for the development of EFP below.

Specifying s_0: A set of formulae s_0 in $\mathcal{L}(\mathcal{F}, \mathcal{AG})$ is said to be a finitary **S5**-theory if it a **S5**-theory and contains only formulae of the following form: (i) φ; (ii) $C(K_i\varphi)$; (iii) $C(K_i\varphi \vee K_i\neg\varphi)$; (iv) $C(\neg K_i\varphi \wedge \neg K_i\neg\varphi)$ where φ is a fluent formula. Intuitively, formulae of type (i) indicate formulas that are true in the actual world; (ii)-(iii) indicate that all agents know that i knows the truth value of φ; formulae (iv) say that all agents know that i does not know whether φ is true or false. It is shown in (Baral et al. 2012) that if s_0 is a finitary **S5**-theory then there exists a unique e-model $\mathcal{M} = (W, R, \pi)$ such that W is finite and (i) for every $w \in W$, $(\mathcal{M}, w) \models s_0$; and (ii) every pe-model $(\mathcal{M}', w')$ such that $(\mathcal{M}', w') \models s_0$ can be reduced by bisimulation to some $(\mathcal{M}, w)$. This means that $(\mathcal{M}, W)$ is a unique e-state satisfying s_0. There are two reasons supporting the choice of finitary **S5**-theory as the specification of initial state. First, it is computable. Second, the majority of benchmarks in epistemic planning have this property. The initial state of the problem in Example 1, for instance, is given in the Supplementary Document.

Specifying planning domain: Using $m\mathcal{A}$, a multi-agent planning domain D can be specified by a tuple $\langle \mathcal{AG}, \mathcal{F}, A, O\rangle$ where $\mathcal{AG}$ is the set of agents and $\mathcal{F}$ is the set of fluents; A is the set of actions, where each action is either an *ontic action* (or *world-changing action*), a *sensing action*, or an *announcement action*. It is assumed that each action $a \in A$ is of a unique type. A also contains the description of preconditions and effects of actions (see below); and O are observability statements, describing the frames of reference of agents with respect to action occurrences. A and O consist of statements of form (1)–(4) and (5)-(6), respectively.

executable a **if** ψ (1)

a **causes** ℓ **if** ψ (2)

a **determines** f (3)

a **announces** ℓ (4)

α **observes** a **if** φ (5)

α **aware_of** a **if** φ (6)

where a is an action, f is a fluent, ℓ is a fluent literal, ψ is a belief formula, α is a set of agents, and φ is a fluent formula.

The preconditions ψ of an action a, denoted by $pre(a)$, is specified by (1). Without loss of generality, it is assumed that for each a, there exists exactly one statement of the form (1) in the domain. The action a in (2), (3), and (4) is an ontic, sensing, or announcement action, respectively. (2) states that a is an ontic action and its execution will cause ℓ to be true in the actual world state if ψ is true. (3) indicates that a is a sensing action and executing a will reveal the truth value of f. (4) says that a is an announcement action and its execution will inform aware agents that ℓ is true.

(5) (resp. (6)) indicates that if a occurs and φ is true, then α is the set of agents who are fully (resp. partially) aware of the action occurrence and its consequences. If an agent does not occur in a statement of the forms (5) or (6), then she is oblivious of the action occurrence and its consequences. In writing the observability statements, we often write x instead of the singleton $\{x\}$.

The semantics of a multi-agent domain is defined by a transition function Φ that maps pairs of actions and e-states to sets of e-states. Let a be an action in A and $(\mathcal{M}, w)$ be an e-model. a is executable in $(\mathcal{M}, w)$ if $(\mathcal{M}, w) \models pre(a)$. Let

$$\begin{aligned} F(a, \mathcal{M}, w) &= \{X \in \mathcal{AG} \mid [Y \textbf{ observes } a \textbf{ if } \varphi] \in A \\ &\qquad \text{such that } (\mathcal{M}, w) \models \varphi \wedge X \in Y\} \\ P(a, \mathcal{M}, w) &= \{X \in \mathcal{AG} \mid [Y \textbf{ aware\_of } a \textbf{ if } \varphi] \in A \\ &\qquad \text{such that } (\mathcal{M}, w) \models \varphi \wedge X \in Y\} \\ O(a, \mathcal{M}, w) &= \mathcal{AG} \setminus (F(a, \mathcal{M}, w) \cup P(a, \mathcal{M}, w)) \end{aligned}$$

These sets represent the set of agents who are fully aware, partially aware, and oblivious of the execution of a in $(\mathcal{M}, w)$, respectively. $m\mathcal{A}$ specifies how the corresponding epistemic action $(\mathcal{E}_{(a,\mathcal{M},w)}, E_{(a,\mathcal{M},w)})$ of an occurrence of a in $(\mathcal{M}, w)$ is defined given a and $(\mathcal{M}, w)$. The precise definition of $(\mathcal{E}_{(a,\mathcal{M},w)}, E_{(a,\mathcal{M},w)})$ is rather long, we refer the reader to the Supplementary Document or (Baral et al. 2012) for the detailed definition. Note that if a is executable in $(\mathcal{M}, w)$, the result of executing a in $(\mathcal{M}, w)$ is an e-model.

Figure 1 shows the epistemic actions (on the right) correspond to the occurrence of $open(A)$ in two e-models (left); in the e-model at the top, B is not looking and in the e-model at the bottom, B is looking. We use the conventional representation of e-models in the figure: nodes represent worlds, labeled links represents the accessibility relation, and π is given implicit (listing only important fluent literals in the nodes). In both models, we omit fluent literals encoding the facts that A and C are looking, A has the key, and the box is locked. Likewise, rectangles represent events, double lines rectangles represent designated events, etc.

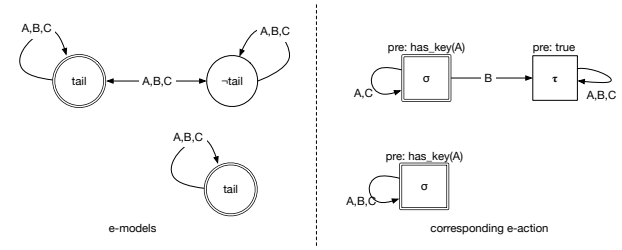


Figure 1: Epistemic action of $open(A)$

We note that the idea of constructing e-actions from the action description and action occurrences has recently been used in (Engesser et al. 2017).

For each e-model $(\mathcal{M}, w)$ and an action a, $\Phi(a, (\mathcal{M}, w))$ denotes the result of executing a in $(\mathcal{M}, w)$. $\Phi(a, (\mathcal{M}, w)) = \emptyset$ if a is not executable in $(\mathcal{M}, w)$, i.e., $(\mathcal{M}, w) \not\models pre(a)$.

$$\Phi(a, (\mathcal{M}, w)) = (\mathcal{M}, w) \otimes (\mathcal{E}_{(a,\mathcal{M},w)}, E_{(a,\mathcal{M},w)})$$

If $(\mathcal{M}, W_d) \models pre(a)$, $\Phi(a, (\mathcal{M}, W_d)) = \bigcup_{w \in W_d} \Phi(a, (\mathcal{M}, w))$; otherwise, $\Phi(a, (\mathcal{M}, W_d)) = \emptyset$. A domain $\langle \mathcal{F}, \mathcal{AG}, A, O\rangle$

Algorithm 1: EFP($\langle \mathcal{F}, \mathcal{AG}, A, O, s_0, \phi_g \rangle$)

Input : A planning problem $P = \langle \mathcal{F}, \mathcal{AG}, A, O, s_0, \phi_g \rangle$
Output: A solution for P if exists; failed otherwise

Compute the initial e-state given s_0: $(\mathcal{M}_i, W_i)$
Initialize a priority queue $q = [(\{(\mathcal{M}_i, W_i)\}, [])]$
while *q is not empty* **do**
  $(\Omega, plan) = dequeue(q)$
  if $(\mathcal{M}, W_d) \models \phi_g$ *for every* $(\mathcal{M}, W_d) \in \Omega$ **then**
    return $plan$
  end
  for *action a executable in every* $(\mathcal{M}, W_d)$ *in* Ω **do**
    Compute $\Omega' = \bigcup_{(\mathcal{M}, W_d) \in \Omega} \Phi(a, (\mathcal{M}, W_d))$
    Compute heuristics and insert $(\Omega', plan \circ a)$ into q
  end
end
return failed

represents a planning domain $\Sigma = (S, Act, \gamma)$ where S is the set of e-states of $\mathcal{L}(\mathcal{F}, \mathcal{AG})$, Act is the set of epistemic actions produced by e-models, actions, and observations in A and O, and γ is defined by Φ. A planning problem $\mathcal{P}$ is a triple (D, s_0, ϕ_g) where D is a domain, s_0 is a finitary **S5**-theory, and ϕ_g is formula. Often, we also write $\langle \mathcal{F}, \mathcal{AG}, A, O, s_0, \phi_g \rangle$ to denote a planning problem.

It is worth to note that it has been discussed in (Baral et al. 2015) that, for certain situations, the semantics of $m\mathcal{A}$ as defined in (Baral et al. 2015) is not intuitive (e.g., it does not allow agents to correct their knowledge). Fortunately, a recent proposal (van Eijck 2017) provides a way to fix this. It involves two steps. The first step corrects the false beliefs of agents by creating $(\mathcal{M}', w)$ from $(\mathcal{M}, w)$ and the action occurrence. The second step is to apply the operation $\otimes$ on $(\mathcal{M}', w)$ and $(\mathcal{E}_{(a,\mathcal{M},w)}, E_{(a,\mathcal{M},w)})$. EFP and PG-EFP implement this modified semantics.

The EFP and PG-EFP Systems

Overall Architecture

The overall architecture of EFP and PG-EFP is given in Algorithm 1. The key modules of EFP and PG-EFP are: **(i)** a pre-processor; **(ii)** initial e-state computation; and **(ii)** a search engine.

- *Pre-processor*: this module is responsible for parsing the planning problem description, setting up the planning domain, that includes the list of agents, the list of actions, the rules for computing frames of reference, and the list of fluents. This module is also responsible for the initialization of necessary data structures (e.g., e-state) and executes some transformations (e.g., the transformation of general planning problems to d-observable problems[3]).
- *Initial e-state computation*: EFP implements the algorithm given in (Son et al. 2014) for computing the initial e-state. As an example, Figure 2 shows the e-state representing the initial state of the problem in Example 1. The fluent literals in the gray box of the figure are true in all worlds in the e-state.

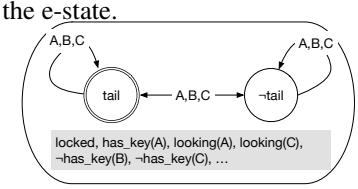


Figure 2: Initial e-state

- *Search engine*: this module is responsible for computing a solution. EFP and PG-EFP implement a best-first search (Algorithm 1). EFP is a breadth-first search planner and PG-EFP is a heuristic search planner, using a heuristic that is derived from epistemic planning graph, described in the next section.

EFP and PG-EFP are modularly organized (e.g., the function Φ is implemented in a separate module) which will facilitate future modifications and extensions.

Epistemic Planning Graph

In this section, we develop a means for deriving heuristics in the implementation of EFP. We note that epistemic planners in the literature do not focus on this issue because the majority of them translate an epistemic planning problem into a classical planning problem and uses classical planners for computing the solutions. The exceptions to this trend are presented in (Huang et al. 2017; Wan et al. 2015) where the planning systems described in these papers use a heuristic similar to the number of satisfaction subgoals in single agent planning. In this paper, we approach this problem from a different angle. Specifically, we start with the fact that we search for plans using forward-search. It is easy to see that a generalization of the notion of a planning graph in single-agent to epistemic planning is a reasonable choice because it allows us to compactly represent the search tree using one single data structure and to extract different ways to extract heuristics. We therefore introduce the notion of an *Epistemic Planning Graph (EPG)* for epistemic planning.

As in single-agent domains, we expect that an epistemic planning graph will also consist of alternating *state levels* and *action levels*. While the content of action levels is obvious (i.e., it should be actions), it is not trivially clear *what should the state levels contain?* There are at least two possible answers: formulae in $\mathcal{L}(\mathcal{F}, \mathcal{AG})$ or epistemic states. The former would require the computation of all effects of an action, which could be an infinite set of formulae, and it is not desirable; this set could be approximated by finding an approximation of common knowledge but this not trivial (e.g., a public announcement of a fluent f makes f a common knowledge). For this reason, we use e-states in state levels. Intuitively, each e-state in a state level represents the updated knowledge/belief of the agents about some fluents after an action is executed. Before we present the formal definition of this notion, we illustrate this idea in the following figure.

In Figure 3, level 0 contains the unique e-state of the planning problem. In this e-state, the actions $open(A)$, $signal(A, B)$, $signal(C, B)$, etc. are executable. The exe-

[3]d-observable problems is defined later in Epistemic Planning Graph: Formal Definition

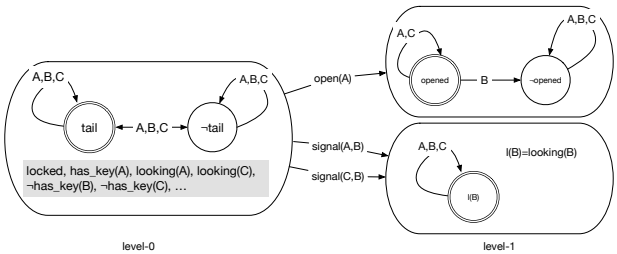


Figure 3: Epistemic planning graph: an illustration

cution of $open(A)$ in the pe-model of the initial e-state will result in A and C know that the box is open and B oblivious about this fact. This creates the top-right e-state of the figure (top of level-1). The execution of either $signal(A, B)$ or $signal(C, B)$ will result in B looking at the box and everyone knows that everyone is looking at the box; this creates the bottom-right e-state of the figure (bottom of level-1).

As it is shown in the application of planning graph for single-agent domains, planning graphs with mutexes provide better heuristics. Therefore, it is natural to expect that EPG should also include mutexes. By definition, a mutex represents a pair of actions with conflicting effects. This means that computing mutexes requires checking whether two actions produce conflicting effects. As it turns out, this is computational cheap and straightforward in single-agent domains under PDDL-specification, computing effects of actions under $m\mathcal{A}$ is not straightforward. For these reasons, we leave the mutex definition for future work.

Epistemic Planning Graph: Formal Definitions

We will next formally define the notion of an epistemic planning graph. Given a planning problem $P = \langle \mathcal{F}, \mathcal{AG}, A, O, s_0, \phi_g \rangle$, we say that P is *deterministically observable* (or *d-observable*) if O contains only statements of the form "α **observes** a" and "α **aware_of** a." It is easy to see that every planning problem P can be translated[4] into an equivalent d-observable problem P', i.e., a solution to P is equivalent to a solution to P' and vice versa. Intuitively, in d-observable problems, $F(a, \mathcal{M}, w)$ (or $P(.)$ and $O(.)$) is independent from $(\mathcal{M}, w)$. For this reason, we will use F_a, P_a, and O_a to denote the sets of agents $F(a, \mathcal{M}, w)$, $P(a, \mathcal{M}, w)$, and $O(a, \mathcal{M}, w)$, respectively. This simplified the definition of an epistemic planning graph. For simplicity of the presentation, we will present the notion of an epistemic planning graph only for d-observable planning problems. Observe that, from now on, we assume that $P = \langle \mathcal{F}, \mathcal{AG}, A, O, s_0, \phi_g \rangle$ is given and is d-observable.

In the following, we will consider *incomplete* e-model (W, R, π) where W, R, and π are defined similarly to the components of an e-model with the exception that $\pi(w)$ can be a partial interpretation of $\mathcal{F}$. An incomplete pe-model (or e-state) is of the form $(\mathcal{M}, w)$ (or $(\mathcal{M}, W)$) where $\mathcal{M}$ is an incomplete e-model. From now on, whenever we refer to an (p)e-model (or e-state), we mean a (possibly incomplete) (p)e-model (or e-state).

For a set of e-states $\mathcal{P} = \{(\mathcal{M}_1, W_1), \ldots, (\mathcal{M}_n, W_n)\}$ and a conjunction of fluent literals $\psi = \ell_1 \wedge \ell_2 \ldots \wedge \ell_k$, we say $\mathcal{P} \mid\sim \psi$ if for every $1 \leq i \leq k$, there exists some $1 \leq j \leq n$ such that $(\mathcal{M}_j, W_j) \models \ell_i$. For a fluent formula φ, let $\bigvee_{i=1}^{k} \varphi_k$ be its DNF, i.e., each φ_i is a conjunction of literals. We say $\mathcal{P} \mid\sim \varphi$ iff $\mathcal{P} \mid\sim \varphi_i$ for some $1 \leq i \leq k$. Intuitively, if $\mathcal{P}$ represents a state level, $\mathcal{P} \mid\sim \varphi$ means that the level, $\mathcal{P}$, "*possibly entails*" φ. $\mid\sim$ is used for checking whether or not some formula could be entailed from a state level. This is similar to the verification of the presence of a literal in a state level in a planning graph. $\mid\sim$ is generalized to arbitrary formula as follows.

Definition 3 *Given a set of e-states $\mathcal{P} = \{(\mathcal{M}_1, W_1), \ldots, (\mathcal{M}_n, W_n)\}$ where $\mathcal{M}_j = (W_j, R_j, \pi_j)$ for $1 \leq j \leq n$, and a belief formula φ, we say $\mathcal{P} \mid\sim \varphi$ if*

- *φ is a fluent formula and $\mathcal{P} \mid\sim \varphi$;*
- *$\varphi = K_i\psi$ and $\{(\mathcal{M}_1, W_1'), \ldots, (\mathcal{M}_n, W_n')\} \mid\sim \psi$ where for $1 \leq j \leq n$, $W_j' = \{w_j' \mid \exists w_j \in W_j$ such that $(w_j, w_j') \in R_j(i)\}$;*
- *$\varphi = \neg\psi$ and ψ is not a fluent formula and $\mathcal{P} \not\mid\sim \psi$;*
- *$\varphi = \varphi_1 \vee \varphi_2$ and ($\mathcal{P} \mid\sim \varphi_1$ or $\mathcal{P} \mid\sim \varphi_2$);*
- *$\varphi = \varphi_1 \wedge \varphi_2$ and ($\mathcal{P} \mid\sim \varphi_1$ and $\mathcal{P} \mid\sim \varphi_2$);*
- *$\varphi = E_\alpha\psi$ and $\mathcal{P} \mid\sim K_i\psi$ for every $i \in \alpha$; and*
- *$\varphi = C_\alpha\varphi$ and $\mathcal{P} \mid\sim E_\alpha^k\varphi$ for every $k \geq 1$, where $E_\alpha^1\varphi = E_\alpha\varphi$ and $E_\alpha^{k+1}\varphi = E_\alpha(E_\alpha^k\varphi)$.*

This allows us to define the notion of an action *potentially applicable* from a set of e-states, which allows us to compute the action level of an epistemic planning graph, as follows.

Definition 4 *An action $a \in A$ is* potentially applicable *in a state level $\mathcal{K}$ if there exists a set of e-states $\mathcal{PS} \subseteq \mathcal{K}$ such that $\mathcal{PS} \mid\sim pre(a)$.*

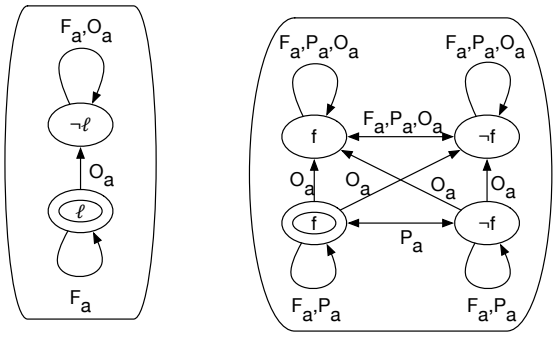
Let $\mathcal{K}$ be a set of e-states and a be an action potentially applicable in $\mathcal{K}$. To complete the definition of an epistemic planning graph, we need to define the set of e-states that could be obtained given that a is executed. We first define a set of e-states $E(\mathcal{K}, a)$ as follows.

- *a is an ontic action*: For each [a **causes** ℓ **if** φ] in A such that $\mathcal{K} \mid\sim \varphi$, let $\mathcal{M} = (W, R, \pi)$ where
  - $W = \{u, v\}$;
  - $R : \mathcal{AG} \to 2^{W \times W}$ is defined as:
    - $R(i) = \{(u, u), (v, v)\}$ for all $i \in F_a$; and
    - $R(i) = \{(u, v), (v, v)\}$ for all $i \in O_a$.
  - $\pi(u) = \{\ell\}$ and $\pi(v) = \{\neg\ell\}$.

  $E(\mathcal{K}, a) = \{(\mathcal{M}, \{u\}) \mid [a$ **causes** ℓ **if** $\varphi] \in A$ such that $\mathcal{K} \mid\sim \varphi\}$.
- *a is a sensing action*: $E(\mathcal{K}, a) = \{(\mathcal{M}, \{u^1\}) \mid [a$ **determines** $f] \in A$ *and* $\mathcal{K} \mid\sim f\} \cup \{(\mathcal{M}, \{u^2\}) \mid [a$ **determines** $f] \in A$ *and* $\mathcal{K} \mid\sim \neg f\}$ where $\mathcal{M} = (W, R, \pi)$ and
  - $W = \{u^1, u^2, v^1, v^2\}$;
  - $R : \mathcal{AG} \to 2^{W \times W}$ is defined as:
    - For $i \in F_a$: $R(i) = \{(u^1, u^1), (u^2, u^2)\} \cup \{(v^1, v^1), (v^2, v^2), (v^1, v^2), (v^2, v^1)\}$;
    - For $i \in P_a$: $R(i) = \{(u^1, u^1), (u^2, u^2), (u^1, u^2), (u^2, u^1)\} \cup \{(v^1, v^1), (v^2, v^2), (v^1, v^2), (v^2, v^1)\}$;

[4] This process is similar to the process of converting an action with conditional effects to a set of actions without conditional effects in classical planning.

- For $i \in O_a$: $R(i) = \{(u^1, v^1), (u^1, v^2), (u^2, v^1), (u^2, v^2)\} \cup \{(v^1, v^1), (v^2, v^2), (v^1, v^2), (v^2, v^1)\}$;
  - $\pi(u^1)$=$\pi(v^1)$=$\{f\}$ and $\pi(u^2)$=$\pi(v^2)$=$\{\neg f\}$.
- *a is an announcement action*: assume that $[a$ **announces** $\varphi]$ is in A. Let W^1 and W^2 be two disjoint sets, each represents the set of all complete interpretations of the set of fluents occurring in φ. We define $E(\mathcal{K}, a) = \{(\mathcal{M}, \{w \mid w \in W^1, w \models \varphi\})\}$ where $\mathcal{M} = (W^1 \cup W^2)$ and
  - For $i \in F_a$: $R(i) = \{(w, w') \mid w, w' \in W^1 \wedge w \models \varphi \wedge w' \models \varphi\} \cup \{(w, w') \mid w, w' \in W^1 \wedge w \not\models \varphi \wedge w' \not\models \varphi\} \cup \{(w, w') \mid w, w' \in W^2\}$;
  - For $i \in P_a$: $R(i) = \{(w, w') \mid w, w' \in W^1\} \cup \{(w, w') \mid w, w' \in W^2\}$;
  - For $i \in O_a$: $R(i) = \{(w, w') \mid w, w' \in W^2\} \cup \{(w, w') \mid w \in W^1 \wedge w' \in W^2\}$.



(a) ontic action (b) sensing or announcement action

Figure 4: Examples of resulting e-states in $E(\mathcal{K}, a)$

Intuitively, $E(\mathcal{K}, a)$ is the set of e-states that partially represents the updated knowledge/belief of the agents after action a is fired. Each e-state in $E(\mathcal{K}, a)$ encodes that agents in F_a know the effects of a, agents in O_a are oblivious, and agents in P_a do not know the effects of a but are aware that those in F_a know the effects of a. Figure 4(a) presents the e-state resulting from the firing of a with the statement $[a$ **causes** ℓ **if** $\top] \in A$, and Figure 4(b) illustrates an e-state of firing a sensing action (resp. an announcement action) a where $[a$ **determines** $f] \in A \wedge \mathcal{K} \mid\sim f$ (resp. $[a$ **announces** $f] \in A$).

Definition 5 *Given a planning problem $P = \langle \mathcal{F}, \mathcal{AG}, A, O, s_0, \phi_g \rangle$, the* epistemic planning graph *of P is an alternative sequence of state levels and action levels $\mathcal{K}_0, A_0, \ldots, \mathcal{K}_k, A_k, \ldots$ where*

- *$\mathcal{K}_0$ is the set consisting of the unique initial e-state of P;*
- *for $i \geq 0$,*
  - *A_i is the set of actions potentially applicable in $\mathcal{K}_i$; and*
  - *$\mathcal{K}_{i+1} = \mathcal{K}_i \cup \big(\bigcup_{a \in A_i} E(\mathcal{K}_i, a)\big)$.*

Algorithm 2 shows the computation of an epistemic planning graph of a planning problem given the e-state in the first state-level. At each iteration, the set of potentially applicable actions are computed and then the set of potential effects of these actions is added to the next state level. The following property guarantees that Algorithm 2 terminates.

Algorithm 2: EpistemicPlanningGraph(P, $(\mathcal{M}_0, W_0)$)

Input : a planning problem $P = \langle \mathcal{F}, \mathcal{AG}, A, O, s_0, \phi_g \rangle$ and an e-state $(\mathcal{M}_0, W_0)$
Output : An epistemic planning graph whose first state level is $\{(\mathcal{M}_0, W_0)\}$

14 Let $\mathcal{K}_0 = \{(\mathcal{M}_0, W_0)\}$
15 $i = 0$
16 **while** *true* **do**
17 Compute $A_i = \{a \in A \mid a \text{ is potentially applicable given } \mathcal{K}_i\}$
18 $i = i + 1$
19 Let $\mathcal{K}_i = \mathcal{K}_{i-1} \cup \Big(\bigcup_{a \in A_i} E(\mathcal{K}_{i-1}, a)\Big)$
20 **if** $\mathcal{K}_i \equiv \mathcal{K}_{i-1}$ *and* $A_i \equiv A_{i-1}$ **then**
21 **return** $\langle \mathcal{K}_0, A_0, \ldots, \mathcal{K}_i, A_i \rangle$
22 **end**
23 **end**

Proposition 1 *For a planning problem with finite sets of actions A, Algorithm 2 terminates.*

The proof of this proposition relies on the following observations: (i) $\mathcal{K}_{i-1} \subseteq \mathcal{K}_i$ (Line 19); and thus (ii) if an action is potentially applicable at level i then it is also potentially applicable at level $i + 1$ (Definition 4); (iii) $E(\mathcal{K}_i, a) \subseteq E(\mathcal{K}_{i+1}, a)$ and $E(\mathcal{K}, a)$ is finite.

It is also easy to see that since $\mid\sim$ stands for "possibly entails," the following property holds.

Proposition 2 *Let $\langle \mathcal{K}_0, A_0, \ldots, \mathcal{K}_i, A_i \rangle$ be the epistemic planning graph returned by Algorithm 2 with the planning problem $P = \langle \mathcal{F}, \mathcal{AG}, A, O, s_0, \phi_g \rangle$ and $(\mathcal{M}, W)$ as input. Furthermore, let φ be a formula and j be the smallest index such that $\mathcal{K}_j \mid\sim \varphi$. Then, the shortest solution of the planning problem $\langle \mathcal{F}, \mathcal{AG}, A, O, \{(\mathcal{M}, W)\}, \varphi \rangle$ has at least j actions.*

In the following, we denote with $level(\varphi)$ the smallest state level index in the epistemic planning graph such that $\mathcal{K}_{level(\varphi)} \mid\sim \varphi$.

Heuristics from Epistemic Planning Graphs

It is well-known that there are several possible ways to extract heuristics from a planning graph in planning in single-agent domain (Nguyen, Kambhampati, and Nigenda 2002). Studying different heuristics from epistemic planning graphs is an interesting topic of research of its own right, but it is not the focus of this paper. For this reason, we will describe the heuristics that we used in the experiments in the next section.

We experimented with two heuristics derived from the epistemic planning graphs. We assume that ϕ_g is a conjunction of formulas $\phi_1 \wedge \ldots \wedge \phi_n$ where ϕ_i is either a fluent formula or a formula of the form $K_i\varphi$ or $C_X\varphi$. We define

$$h^{max}(\phi_g) = \max\{level(\phi_i) \mid i = 1, \ldots, n\} \quad (7)$$

and

$$h^{sum}(\phi_g) = \sum_{i=1}^{n} level(\phi_i) \quad (8)$$

Experimental Evaluation and Discussion

Experimental Evaluation.

As we have mentioned earlier, we implement two planners– one that uses Epistemic Planning Graph as heuristics, which we call "PG-EFP", and one that does not use heuristic but performs *breadth first search* to search for the plan, which we call "EFP". We note that the heuristic used in PG-EFP is the h^{sum} heuristic defined in 8. We note that we did experiment with h^{max}, but its performance is generally worse than EFP with h^{sum}. For the sake of space, we omit such comparisons form this paper.

In our experiments, we compare EFP and PG-EFP against the two systems proposed in (Muise et al. 2015; Huang et al. 2017). The first system, which is called RP-MEP, supports the notion of planning with nested belief, and demonstrates how to automatically convert such planning problems into problems that can be solved using classical planning technology. The limitation of this system is that it limits the number of nested belief modality. The second system, which is called MEPK, supports efficient reasoning in the multi-agent KD45 logic by using alternating cover disjunctive formulas (ACDFs). Such ACDFs are subjected to proposed belief-revision and update algorithms that adapt the PrAO algorithm originally developed for contingent planning. We use publicly-available implementations of RP-MEP and MEPK in our experiments.[5] We did not compare our system with the system in (Kominis and Geffner 2017) as it is an online planner. We also did not compare our system with the system in (Kominis and Geffner 2015) as this has been already compared with the system in (Huang et al. 2017) and its performance is not as good as that of MEPK.

We evaluate EFP and PG-EFP using the *Selective-communication* (SC), and *Collaboration-and-communication* domains that are adapted from (Kominis and Geffner 2015),[6] as well as *Assemble Line* introduced in (Huang et al. 2017), and *Logistics* (LO) that was used in the Competition of Distributed and Multiagent Planners 2015 (CoDMAP'15).[7]

In the *Selective-communication* (SC(n,m)), there are n agents, each of them initially is in one of m rooms in a corridor. An agent, says a, can move from a room to a neighboring room by actions *left* and *right*. When the agent tells some information—which is the truth value of a fluent q—in a room i (i.e., executing an announcement action "shout_i"), all the other agents in the same room i or in a neighboring room of i can hear what was told. The goal is that some agents get to know q while some other agents do not. In this experiment, we will vary n, m, the depth of the knowledge d, and the length of the plan by varying the goal.

In the *Collaboration-and-communication* (CC(n,m,k)), there is a corridor of $k \geq 2$ rooms. m boxes are located in some rooms. n agents can move back and forth along this corridor. When an agent gets into a room, he can see if a box is in the room. An agent can communicate information to another agent. Initially, we set all agents are in room 2 and the boxes are not there. The goals are varied in which some agents know the position of some boxes.

The *Grapevine* (GR(n)) problem is a modification of a similar domain from (Kominis and Geffner 2015)): n agents are located in two rooms, and they share secrets with agents in the same room. The domain supports different goals, from sharing secrets with other agents to having misconceptions about agents' beliefs.

Assembly-line: AL(d). There are two agents, each responsible for processing a part of a product. It is possible that an agent fails in processing his part. An agent can inform the other agent of the status of his task. Two agents decide to assemble the product or restart, depending on their knowledge of the status of the agents' tasks. In order to compare with RP-MEP and MEPK, we vary the depth of the knowledge d in this experiment.

Logistics: LO. The logistics domain is recommended by one of the reviewers to evaluate the scalability of EFP and PG-EFP since it consists of a large number of fluents and actions. This benchmark is not originally developed for epistemic planning. As such, we use its slightly-modified version by modeling the public/privacy separation of fluents (Brafman and Domshlak 2008) using the "observes" statement of the form (5). In our experiment, the Logistics problem has 3 agents (i.e., airplane, truck1, and truck2), 2 packages, and 4 locations (i.e., "airport1", "airport2", "pos1", and "pos2"). The packages can be moved from one location to another location by being loaded/unloaded onto agents and then moving agents. We vary the length L of the optimal plan by changing the desired locations of packages as goals.

All experiments are performed on a 2.8 GHz Intel Core i7 machine with 16GB of memory. We report the runtimes in second for all experiments. We set the timeout to 25 minutes. "TO" means one algorithm fails to solve one problem due to timeout. The results of our experiments are summarized in Tables 1–5. In the tables, L denotes the length of the shortest plan (optimal plan) and d the depth of knowledge. We make the following observations:

- EFP performs reasonably well comparing to other systems in the SC domain. When depth (d) of the knowledge increases, EFP's performance does not decrease but the performance of other systems (i.e., MEPK and RP-MEP) gets significantly worse. The reason is that the representation of the problem in EFP remains unchanged when d increases. In contrast, the size of the problem representation for MEPK and RP-MEP increases when d increases. This results in the worse performance of MEPK and RP-MEP when d increases. In this domain, PG-EFP cannot solve some instances which contain goals with negation (e.g., goal of the form $\neg K_1\varphi$). Our analysis shows that the heuristic h^{max} does not work well for this situation. On the other hand, if PG-EFP can solve an instance; it is almost always the fastest, due to the heuristic h^{sum}.
- EFP performs well in the Grapevine domain against RP-

[5] We retrieved RP-MEP from `https://bitbucket.org/haz/pdkb-planning` and MEPK from `https://github.com/sysulic/MEPK`.

[6] SC is called "Corridor" in (Muise et al. 2015)

[7] LO is retrieved from `http://agents.fel.cvut.cz/codmap` with *centralized* and *unfactored-privacy* setting.

Selective Communication: SC(3,4) $\lvert\mathcal{AG}\rvert = 3, \lvert\mathcal{F}\rvert = 5, \lvert A\rvert = 7$						Selective Communication: SC(5,6) $\lvert\mathcal{AG}\rvert = 5, \lvert\mathcal{F}\rvert = 7, \lvert A\rvert = 9$						Selective Communication: SC(7,8) $\lvert\mathcal{AG}\rvert = 7, \lvert\mathcal{F}\rvert = 9, \lvert A\rvert = 11$					
L	d	MEPK	RP-MEP	EFP	PG-EFP	L	d	MEPK	RP-MEP	EFP	PG-EFP	L	d	MEPK	RP-MEP	EFP	PG-EFP
2	1	.01	.1	.01	.02	2	1	.6	.2	.02	.04	5	1	35	.36	.22	TO
	3	.2	.5	.02	.07		3	TO	3.2	.02	.04		3	TO	10.7	.22	TO
	5	TO	28	.03	.08		4	TO	51.58	.03	.04		4	TO	292	.24	TO
3	1	.02	.1	.02	.06	4	1	.68	.2	.07	TO	7	1	35	.36	1.9	TO
	3	.2	.5	.02	.07		3	TO	3.18	.08	TO		3	TO	10.8	1.92	TO
	5	TO	30	.02	.06		4	TO	54.78	.08	TO		4	TO	300	1.9	TO
5	1	.05	.1	.08	TO	6	1	.81	.2	.51	.35	9	1	35.7	.32	23.7	1.86
	3	.21	.6	.09	TO		3	TO	3.21	.52	.36		3	TO	12.72	24	1.9
	5	TO	28	.1	TO		4	TO	51.81	.51	.34		4	TO	312	23.5	1.93

Table 1: Runtimes for Selective Communication Problems

MEP. It is not as fast as RP-MEP in the first configuration of this problem but is faster than RP-MEP when d increases in Configuration 2. We believe that this reflects the trade-off between the representation and the computation of the transition function. In general, computing Φ is more expensive than computing the transition function implemented in RP-MEP. This is the reason when the domain is small, EFP is not as good as RP-MEP. However, when the size of the problem increases (i.e., $|\mathcal{AG}|$, $|\mathcal{F}|$, $|A|$, and d increase in Configuration 2), the size of the representation used in RP-MEP increases and this affects its performance much more than the complexity of computing Φ. We were unable to run PG-EFP in this domain as our naive translation from general domain to deterministically-observable produces a significantly larger input. We believe that some optimization of the translation could help in this regard.

Grapevine - Configuration 1 $\lvert\mathcal{AG}\rvert = 3, \lvert\mathcal{F}\rvert = 9, \lvert A\rvert = 24$				Grapevine - Configuration 2 $\lvert\mathcal{AG}\rvert = 5, \lvert\mathcal{F}\rvert = 15, \lvert A\rvert = 60$			
L	d	RP-MEP	EFP	L	d	RP-MEP	EFP
2	1	0.090	0.034	2	1	0.260	0.912
	2	0.255	0.036		2	1.970	0.908
	3	1.178	0.035		3	26.110	0.974
4	1	0.088	1.130		4	1499	0.954
	2	0.256	1.125		5	TO	0.982
	3	1.222	1.141	3	1	0.271	10.879
5	1	0.090	22.687		2	1.990	10.320
	2	0.267	22.692		3	26.407	10.568
	3	1.220	22.694		4	1575	10.836
					5	TO	10.967
				4	1	0.264	88.662
					2	2.035	90.128
					3	26.510	89.135
					4	1632	87.631
					5	TO	88.164

Table 2: RP-MEP vs. EFP in Grapevine

- For the AL domain (Table 3(Left)), we were unable to create the input for RP-MEP as the instances become too large to do the translation manually. Other than that, the performances of EFP and PG-EFP are similar to their performances in the SC domain and are independent of the number of depth of knowledge. MEPK's performance decreases when the depth of knowledge increases.
- The coin-in-the-box domain (Example 1) is not solvable using other planners. Table 3(Right) shows the comparison between EFP and PG-EFP to investigate the influence of the heuristic. In the last column of Table 3(Right), (X) stands for the length X of the plan returned by PG-EFP, and (N/A) means that the respective problem has no plan. We change the goals to create different instances. We can observe that PG-EFP is often faster than EFP.

Assemble Line: AL(d) $\lvert\mathcal{AG}\rvert = 2, \lvert\mathcal{F}\rvert = 4, \lvert A\rvert = 6$			
d	MEPK	EFP	PG-EFP
2	.03	1.9	.9
5	.11		
10	5.47		
15	175		
20	TO		

Coin in the Box $\lvert\mathcal{AG}\rvert = 3, \lvert\mathcal{F}\rvert = 8, \lvert A\rvert = 31$		
L	EFP	PG-EFP
2	.04	.27(2)
3	.22	1.16(3)
5	3.58	1.44(6)
no plan	TO	.43(N/A)

Table 3: AL domain (left) and Coin-in-the-box domain (Right)

- Table 4 compares the two systems EFP and PG-EFP in the CC domain. In this domain, we did not run the experiment with MEPK as it produces solutions that are not an action sequence and only runs with $d = 1$. The experiments are done on four different configurations and different goals. We note that CC(3,3,3) has smaller number of $|\mathcal{F}|$ and $|A|$ than CC(3,2,3) because we experiment with a different representation of the problem. It is interesting to observe that PG-EFP is slower than EFP in less-complicated problems (i.e., problems with $L = 2$), but significantly faster than EFP (several magnitudes faster) in more-complicated problems (i.e., problems with $L > 2$). The reason for the former observation is that PG-EFP needs to compute epistemic planning graphs while EFP does not. However, when the problems become more complex, the search space of PG-EFP seems to reduce significantly and much smaller than that of EFP due to the heuristic derivable from the EPG. These results show that the heuristic produced by the epistemic planning graph is indeed quite useful.

Problem	L	EFP	PG-EFP
CC(2,2,3) $\lvert\mathcal{AG}\rvert = 2, \lvert\mathcal{F}\rvert = 10, \lvert A\rvert = 16$	2	.61	.81
	5	48.6	2.5
	6	278.6	4.3
CC(2,2,4) $\lvert\mathcal{AG}\rvert = 2, \lvert\mathcal{F}\rvert = 14, \lvert A\rvert = 22$	2	22.2	27.5
	4	TO	70
	7	TO	160
CC(3,2,3) $\lvert\mathcal{AG}\rvert = 3, \lvert\mathcal{F}\rvert = 13, \lvert A\rvert = 24$	2	3.3	2.3
	5	257.9	7.9
	6	TO	10.3
CC(3,3,3) $\lvert\mathcal{AG}\rvert = 3, \lvert\mathcal{F}\rvert = 12, \lvert A\rvert = 21$	2	.42	.68
	5	115.7	2.27
	6	TO	3

Table 4: EFP vs. PG-EFP in CC domain

- The logistics domain cannot be modeled using the speci-

fication language $m\mathcal{A}$ in a straightforward manner due to the fact that the observability of agents is specified at the fluent levels (e.g, when an action is executed, some agents observe one effect and others observe another one), and $m\mathcal{A}$ does not consider this situation yet. For this reason, we used a slightly-modified representation of this domain in testing EFP and PG-EFP. Table 5 displays the runtime comparison between the two systems EFP and PG-EFP in the LO domain. In Table 5, (X) stands for the length X of the plan that is returned by PG-EFP. The results showed in Table 5 are consistent with those shown earlier, and exhibit that PG-EFP can scale up to solve problems with a large number of fluents and actions. This is clearly due to the heuristic produced by the epistemic planning graph.

Logistics $\lvert\mathcal{AG}\rvert = 3, \lvert\mathcal{F}\rvert = 66, \lvert A\rvert = 84$		
L	EFP	PG-EFP
3	10.32	16.63(3)
4	75.1	27.88(4)
30	TO	1355.6(31)
31	TO	1447.2(36)

Table 5: EFP vs. PG-EFP in LO domain

Solution quality: We now discuss about solution quality in terms of the length of the plans that are returned by EFP and PG-EFP. In our experiments, we observed that PG-EFP returns optimal plans in most problems, except for some in the coin-in-the-box domain (see the row of $L = 5$ in Table 3(Right)) and some in the logistics domain (see the rows of $L = 30, 31$ in Table 5). Thus, we report only the plan lengths for PG-EFP in the coin-in-the-box domain and the logistics domain. Theoretically, EFP returns the optimal plan whose length is smallest since it is a breadth-first search planner. In contrast, as PG-EFP is a heuristic search planner that uses heuristic derived from epistemic planning graph, its plan is not guaranteed to be optimal.

Discussions

EFP (or PG-EFP) is closely related to RP-MEP (Muise et al. 2015) or the system (called K(P) hereafter) described in (Kominis and Geffner 2015). The two planners differ from the system MEPK in (Huang et al. 2017) in that they do not use a special search algorithm why MEPK uses a special search algorithm, called PrAO, from (To, Son, and Pontelli 2011). EFP (or PG-EFP) is an alternative to the other systems in three aspects: *(i)* the methods of searching for solution in EFP is different; *(ii)* EFP and K(P) can deal with common knowledge while other systems do not.

The proposed systems represent an ideal testbed for research in the area of epistemic planning in multi-agent settings. The advantages offered by the proposed platform are:

- The system is the first of its kind, supporting planning with complex forms of knowledge and beliefs; EFP does not require restrictions on the nesting of knowledge/beliefs, thus allowing us, e.g., to reason about common knowledge.
- The platform, even in its current prototype form, can solve problems for which RP-MEP or other systems have difficulty; for some problems, even for a small number of nested beliefs (e.g., 5), the generation of the planning instance for other planners takes more than an hour.
- The system serves as a research platform; its modular organization allows researchers to experiment with different transitions, heuristics, and state representations.
- Finally, it should be noted that EFP and PG-EFP can deal with both knowledge and belief goals as (Son et al. 2015) shows that epistemic actions of the form described in this paper, which are generated from an action a and an e-state $(\mathcal{M}, W)$, can maintain the $\mathbf{KD}_{45}$ properties of an e-state. As such, a goal with both K and B-modalities can be dealt with by (a) using the equivalence $K\varphi \equiv B\varphi \wedge \varphi$ to remove the occurrences of K in the goal; and then (b) planning with the new goal.

Although EFP performs reasonably well, there are a number of issues that require further study so that it can efficiently deal with larger problems. In our experiments, we observe the following:

- If the number of unknown fluents in the actual world of the initial e-state is high, then EFP does not work well. The reason for this is the size, in terms of the number of worlds and the number of edges, of the initial e-state. For example, for the CC(2, 3, 4) problems, there are 17 fluents whose values are unknown in the initial e-state. This leads to the initial state has 512 pe-models, and 524288 edges. This raises the question of how to represent and reason with pe-models and e-states with a high-degree of connectivity.
- An e-state is essentially a graph. Checking for graph isomorphism is not a computational easy task. As such, we did not check for repeated element in the queue q of Algorithm 1. How best to check for repeated element in the queue and whether or not it improves the performance of EFP are the two questions left for the future work.

Conclusions and Future Works

We describe two forward search epistemic planners, EFP and PG-EFP, and experimentally evaluate these planners with domains collected from the literature. We also introduce the notion of an epistemic planning graph and provide algorithm for computing epistemic planning graphs. The experimental evaluation of the two planners shows that both are working reasonable well comparing to other epistemic planners; and that heuristics derived from epistemic planning graphs are indeed useful. In the near future, we plan to (i) generate more informative heuristic; (ii) develop methods for comparing e-states so that we can eliminate repeated e-states from the queue; (iii) investigate the notion of mutexes for epistemic planning graph; and (iv) investigate alternative representation to improve the performance of the planners.

Acknowledgement

The last two authors are partially supported by the NSF grant HRD-1345232. We would like to thank Christian Muise for assisting us to install RP-MEP.

References

Aucher, G., and Bolander, T. 2013. Undecidability in epistemic planning. In *Proc. of IJCAI*, 27–33.

Baltag, A., and Moss, L. S. 2004. Logics for epistemic programs. *Synthese* 139(2):165–224.

Baral, C.; Gelfond, G.; Pontelli, E.; and Son, T. C. 2012. An action language for reasoning about beliefs in multi-agent domains. In *Proc. of NMR*.

Baral, C.; Gelfond, G.; Pontelli, E.; and Son, T. C. 2015. An action language for multi-agent domains: Foundations. *CoRR* abs/1511.01960.

Baral, C.; Bolander, T.; van Ditmarsch, H.; and McIlraith, S. A. 2017. Epistemic planning. *Dagstuhl Reports* 7(6):1–47.

Bolander, T., and Andersen, M. B. 2011. Epistemic planning for single and multi-agent systems. *Journal of Applied Non-Classical Logics* 21(1):9–34.

Bolander, T.; Jensen, M. H.; and Schwarzentruber, F. 2015. Complexity results in epistemic planning. In *Proc. of IJCAI*, 2791–2797.

Brafman, R. I., and Domshlak, C. 2008. From one to many: Planning for loosely coupled multi-agent systems. In *Proc. of ICAPS*, 28–35.

Charrier, T.; Maubert, B.; and Schwarzentruber, F. 2016. On the impact of modal depth in epistemic planning. In *Proc. of IJCAI*, 1030–1036.

Crosby, M.; Jonsson, A.; and Rovatsos, M. 2014. A single-agent approach to multiagent planning. In *Proc. of ECAI*, 237–242.

Engesser, T.; Bolander, T.; Mattmüller, R.; and Nebel, B. 2017. Cooperative epistemic multi-agent planning for implicit coordination. In *Proc. of M4M@ICLA*, 75–90.

Fagin, R.; Halpern, J.; Moses, Y.; and Vardi, M. 1995. *Reasoning about Knowledge*. MIT press.

Huang, X.; Fang, B.; Wan, H.; and Liu, Y. 2017. A general multi-agent epistemic planner based on higher-order belief change. In *Proc. of IJCAI*, 1093–1101.

Kominis, F., and Geffner, H. 2015. Beliefs in multiagent planning: From one agent to many. In *Proc. ICAPS*, 147–155.

Kominis, F., and Geffner, H. 2017. Multiagent online planning with nested beliefs and dialogue. In *Proc. ICAPS*, 186–194.

Löwe, B.; Pacuit, E.; and Witzel, A. 2011. DEL planning and some tractable cases. In *Proc. of LORI*. 179–192.

Muise, C. J.; Belle, V.; Felli, P.; McIlraith, S. A.; Miller, T.; Pearce, A. R.; and Sonenberg, L. 2015. Planning over multi-agent epistemic states: A classical planning approach. In *Proc. of AAAI*, 3327–3334.

Nguyen, X.; Kambhampati, S.; and Nigenda, R. 2002. Planning graph as the basis for deriving heuristics for plan synthesis by state space and CSP search. *Artificial Intelligence* 135(1-2):73–123.

Son, T. C.; Pontelli, E.; Baral, C.; and Gelfond, G. 2014. Finitary s5-theories. In *Proc. of JELIA*, 239–252.

Son, T. C.; Pontelli, E.; Baral, C.; and Gelfond, G. 2015. Exploring the KD45 property of a kripke model after the execution of an action sequence. In *Proc. of AAAI*, 1604–1610.

To, S. T.; Son, T. C.; and Pontelli, E. 2011. Contingent planning as AND/OR forward search with disjunctive representation. In *Proc. of ICAPS*.

van Benthem, J.; van Eijck, J.; and Kooi, B. P. 2006. Logics of communication and change. *Inf. Comput.* 204(11):1620–1662.

van der Hoek, W., and Wooldridge, M. 2002. Tractable multiagent planning for epistemic goals. In *Proc. of AAMAS*, 1167–1174.

van Eijck, J. 2004. Dynamic epistemic modelling. Technical report.

van Eijck, J. 2017. Public announcements and public lies. Technical report, Lying Workshop.

Wan, H.; Yang, R.; Fang, L.; Liu, Y.; and Xu, H. 2015. A complete epistemic planner without the epistemic closed world assumption. In *Proc. of IJCAI*, 3257–3263.