...

What you get is what you see: Decomposing Epistemic Planning using Functional STRIPS

Guang Hu, Tim Miller, and Nir Lipovetzky

School of Computing and Information Systems
The University of Melbourne, Australia
ghu1@student.unimelb.edu.au, {tmiller,nir.lipovetzky}@unimelb.edu.au

Abstract

Epistemic planning — planning with knowledge and belief — is essential in many multi-agent and human-agent interaction domains. Most state-of-the-art epistemic planners solve this problem by compiling to propositional classical planning, for example, generating all possible knowledge atoms, or compiling epistemic formula to normal forms. However, these methods become computationally infeasible as problems grow. In this paper, we decompose epistemic planning by delegating reasoning about epistemic formula to an external solver. We do this by modelling the problem using *functional STRIPS*, which is more expressive than standard STRIPS and supports the use of external, black-box functions within action models. Exploiting recent work that demonstrates the relationship between what an agent 'sees' and what it knows, we allow modellers to provide new implementations of externals functions. These define what agents see in their environment, allowing new epistemic logics to be defined without changing the planner. As a result, it increases the capability and flexibility of the epistemic model itself, and avoids the exponential pre-compilation step. We ran evaluations on well-known epistemic planning benchmarks to compare with an existing state-of-the-art planner, and on new scenarios based on different external functions. The results show that our planner scales significantly better than the state-of-the-art planner against which we compared, and can express problems more succinctly.

Keywords: Epistemic Planning, Functional STRIPS, Perspective Models

1. Introduction

Automated planning is a model-based approach to study sequential decision problems in AI [16]. Planning models describe the environment and the agents with concise planning languages, such as STRIPS. It then submits the description of the model to a general problem solver in order to find a sequence of actions to achieve a certain desired goal state. The description of the problem, in general, tracks the changes in the state of the environment. However, in many scenarios, an agent needs to reason about the knowledge or belief of other agents in the environment. This concept is known as *epistemic planning* [4], a research topic that brings together the knowledge reasoning and planning communities.

Epistemic logic is a formal account to perform inferences and updates about an agent's own knowledge and belief, including group and common knowledge in the presence of multiple agents [22]. Epistemic planning is concerned about action theories that can reason not only about variables representing the state of the world, but also the belief and knowledge that other agents have about those variables. Thus, epistemic planning intends to find the best course of action taking into account practical performance considerations when reasoning about knowledge and beliefs [4]. Bolander and Andersen (2011) first used event-based models to study epistemic planning in both single and multi

agent environments, and gave a formal definition of epistemic planning problems using Dynamic Epistemic Logic (DEL) [3].

There are typically two frameworks in which epistemic planning are studied. First, is to use DEL. This line of research investigates the decidability and complexity of epistemic planning and studies what type of problems it can solve [39, 23, 40]. The second is to extend existing planning languages and solvers to epistemic tasks [33, 34, 24, 25, 26]. In this paper, we take the latter approach.

The complexity of epistemic planning is undecidable in the general case. Thus, one of the main challenges of epistemic planning concerns computational efficiency. The dominant approach in this area it to rely on compilations. These solutions pre-compile epistemic planning problems into classical planning problems, using off-the-shelf classical planners to find solutions [33, 34, 24, 25]; or pre-compile the epistemic formulae into specific normal forms for better performance during search [23, 40]. Such approaches have shown to be fast at planning, but the compilation is computationally expensive.

This paper departs from previous approaches in two significant ways. First, we propose a model that exploits recent insights defining what an agent knows as a function of what it 'sees' [6, 14]. Cooper et al. [6] define 'seeing relations' as modal operators that 'see' whether a proposition is true, and then define knowledge of a proposition p as $K_i p \leftrightarrow p \wedge S_i p$; that is, if p is true and agent sees p, then it knows p. Thus, the seeing modal operator is equivalent to the 'knowing whether' operator [10, 32]. We generalise the notion of seeing relations to *perspective functions*, which are functions that determine which variables an agent sees. The domain of variables can be discrete or continuous, not just propositional. The basic implementation of perspective functions is just the same as seeing relations, however, we show that by changing the definition of perspective functions, we can establish new epistemic logics tailored to specific domains, such as Big Brother Logic [14]: a logic about visibility and knowledge in two-dimensional Euclidean planes.

Second, we show how to integrate perspective functions within functional STRIPS models as *external functions* [13]. External functions are black-box functions implanted in a programming language (in our case, C++), that can be called within action models. Epistemic reasoning is delegated to external functions, where epistemic formulae are evaluated lazily, avoiding the exponential blow-up from epistemic formulae present in other compilation-based approaches. This delegation effectively decomposes epistemic reasoning from search, and allows us to implement our approach in any functional STRIPS planner that supports external functions. Further, the modeller can implement new perspective functions that are tied to specific domains, effectively defining new external solvers.

In our experiments we use a width-based functional STRIPS planner [13] that is able to evaluate the truth value of epistemic fluents with external solvers, and solve a wide range of epistemic problems efficiently, included but not limited to, nested knowledge, distributed knowledge and common knowledge. We also show how modellers can implement different perspective functions as external functions in the functional STRIPS language, enabling the use of domain-dependent epistemic logics. Departing from propositional logic give us flexibility to encode expressive epistemic formulae concisely. We compare our approach to a state-of-the-art epistemic planner that relies on a compilation to classical planning [33]. The results show that, unlike in the compilation based approaches, the depth of nesting and the number of agents does not affect our performance, avoiding the exponential blow up due to our lazy evaluation of epistemic formulae.

In the following sections we give a brief background on both epistemic logic and epistemic planning (Section 2). We then introduce a new model, the *agent perspective model* (Section 3). We discuss implementation details using a functional STRIPS planner (Section 4), and report experiments on several well-known benchmarks, along with two new scenarios to demonstrate the expressiveness of the proposed approach (Section 5).

2. Background

In this section, we briefly introduce the three main areas related to this work: (1) classical planning; (2) epistemic logic; and (3) epistemic planning.

2.1. Classical Planning

Planning is the model based approach to action selection in AI, where the model is used to reason about which action an agent should do next [16]. Models vary depending on the assumptions imposed on the dynamics of the world, from classical where all actions have deterministic instantaneous effects and the world is fully known, up

to temporal or POMDP models, where actions have durations or belief distributions about the state of the world. Models are described concisely through declarative languages such as STRIPS and PDDL [11, 31], general enough to allow the encoding of different problems, while at the same time revealing important structural information that allow planners to scale up. In fact, most planners rely on exploiting the structure revealed in the action theory to guide the search of solutions, from the very first general problem solver [37] up to the latest computational approaches based on SAT, and heuristic search [36, 35]. On the other hand, declarative languages have limited the scope of planning, as certain environments representing planning models are difficult to encode declaratively, but are easily defined through simulators such as the Atari video games [2]. Consequently, a new family of width-based planners [27, 28, 30] have been proposed, broadening the scope of planning and have been shown to scale-up even when the planning model is described through simulators, only requiring the exposure of the state variables, but not imposing any syntax restriction on the action theory [13], where the denotation of some symbols can be given procedurally through simulators or by external theories.

In this paper we focus on epistemic planning, which considers the *classical planning model* as a tuple $S = \langle S, s_0, S_G, Act, A, f, c \rangle$ where S is a set of states, $s_0 \in S$ is the initial state, $S_G \subseteq S$ is the set of goal states, Act is the set of actions, $A(s)$ is the subset actions applicable in state s, f is the transition function so that $f(a, s)$ represents the state s' that results from doing action a in the state s, and $c(a, s)$ is a cost function. The solution to a classical planning model S, called a plan, is a sequence of actions that maps the initial state into a goal state, i.e., $\pi = \langle a_0, \ldots, a_n \rangle$ is a plan if there is a sequence of states $s_0, \ldots, s_{n+1}$ such that $a_i \in A(s_i)$ and $s_{i+1} = f(a_i, s_i)$ for $i = 0, \ldots, n$ such that $s_{n+1} \in S_G$. The cost of plan π is given by the sum of action costs $c(a_i, s_i)$ and a plan is optimal if there is no plan with smaller cost.

A classical planning model can be represented in STRIPS by a tuple $P = \langle F, O, I, G \rangle$ where: F is the set of all possible facts or propositions, O the set of all actions, $I \subseteq F$ a set of all true facts in the initial situation, and $G \subseteq F$ a set of facts that needs to be true as the goal conditions.

Besides the model, a solver, which is also called a planner, plays another important role in planning. One of the most successful computational approaches to planning is heuristic search. Besides the choice of search algorithm, the key feature which distinguishes planners is the heuristic function chosen [20]. To achieve good performance, the heuristic funstions should be as informed as possible. For example, one of the current best-performing planner, LAMA, uses a landmark-based heuristic derived from the model [35] along with other delete-relaxation heuristics [16]. The downside is that most heuristics require the model to be encoded in STRIPS or PDDL, but not all the problem can be easy to modeled declaratively.

The standard planning languages and solvers do not support the use of procedures or external theories. One exception is the FS planner [12], that supports the Functional STRIPS language [15], where the denotation of (non-fluent) function symbols can be given extensionally by means of atoms, or intensionally by means of procedures. Procedures also appear as an extension of PDDL under the name of semantic attachments [7]. The reason why procedures are not "first-class citizens" in planning languages is that there was no clear way to deal with them that is both general and effective. Recently, a new family of algorithms have been proposed in classical planning known as Width-based planning [27]. The latest planner known as Best-First Width Search (BFWS) [29], has been shown to scale up even in the presence of functional symbols defined procedurally [13]. *BFWS* exploits a concept called *novelty* in order to prune or guide the search. The novelty of a new state is determined by the presence of values in its state variables that are seen for the first time during the search. It keeps checking when generating search nodes whether there is something novel about them. The novelty of the node is the size of the minimal tuple of state variables that are seen for the fist time during the search.

The f-STRIPS planner using $BWFS$ has been compared over 380 problems with respect of the performance of FF* [19], LAMA-11 [35], and BFWS(f_5) [29]. The results show that the f-STRIPS BFWS planner performances is as good as BFWS(f_5), which relies on a STRIPS model, and slightly better than LAMA-11. It is worth to mention that FF and LAMA have been the top-performing planners for last 15 years, and BFWS(f_5) has been shown to win the agile track in the last International Planning Competition 2018, and is a state-of-art planner for classical planning. The f-STRIPS BFWS planner thus can cope with externally defined functional symbols while performing well with respect to other planners. Therefore, we choose their planner as our developing tool.[1]

---

[1]The planner is available through `https://github.com/aig-upf/2017-planning-with-simulators`.

2.2. Epistemic Logic

In this section, we give the necessary preliminaries for epistemic logic – the logic of knowledge. Knowledge in a multi-agent system is not only about the environment, but also about the agents' knowledge about the environment, and of agents' knowledge of others' knowledge about the environment, and so on.

Fagin et al. [9] provides a formal definition of epistemic logic as follows. Given a set of countable set of all primitive propositions $Prop = \{p_1, p_2, ...\}$ and a finite set of agents $Agt = \{a_1, a_2, ...\}$, the syntax for epistemic logic is defined as:

$$\varphi ::= p \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid K_i\varphi,$$

in which $p \in Prop$ and $i \in Agt$.

$K_i\varphi$ represents that agent i knows proposition φ, $\neg$ means negation and $\wedge$ means conjunction. Other operators such as disjunction and implication can be defined in the usual way.

Fagin et al. [9] define the semantics of epistemic logic using Kripke structures, as standard in modal logic. A Kripke structure is a tuple $M = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$ where:

- S is a non-empty set of states;

- $\pi(s)$ is an interpretation function: $Prop \mapsto \{true \mid false\}$; and

- $\mathcal{K}_1 \ldots \mathcal{K}_n$ represents the *accessibility relations* over states for each of the n agents in Agt.

Given a state s and a proposition p, the evaluation of p over s is $\pi(s)(p)$. If p is true in s, then $\pi(s)(p)$ must be *true*, and vice-versa. $\mathcal{K}_i$ for agent i is a binary relation over states, which is the key to reason about knowledge. For any pair of states s and t, if $(s, t) \in \mathcal{K}_i$, then we can say agent i cannot distinguish between t and s when in state s. In other words, if agent i is in s, the agent can consider t as the current state based on all the information it can obtain from state s. With this definition of Kripke structure, we can define the semantics of knowledge.

Given a state s, a proposition p, a propositional formula φ and a Kripke structure M, the truth of two basic formulae are defined as follows:

- $(M, s) \vDash p$ iff $\pi(s)(p) = true$

- $(M, s) \vDash K_i\varphi$ iff $(M, t) \vDash \varphi$ for all such that $(s, t) \in \mathcal{K}_i$

$(M, s) \vDash p$ represents p is true at (M, s), which means, the evaluation of p in state s, $\pi(s)(p)$ must be true. Standard propositional logic rules define conjunction and negation. $(M, s) \vDash K_i\varphi$ is defined by agent φ being true at all worlds t reachable from s via the accessibility relation $\mathcal{K}_i$. This allows knowledge to be nested; for example, K_aK_bp represents that agent a knows that agent b knows p, which means p is true at all worlds reachable by applying accessibility relation $\mathcal{K}_a$ followed by $\mathcal{K}_b$.

From these basic operators, the concept of group knowledge can be defined. For this, the grammar above is extended to:

$$\varphi ::= p \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid K_i\varphi \mid E_G\varphi \mid D_G\varphi \mid C_G\varphi,$$

in which $p \in Prop$, $i \in Agt$, and G is a non-empty set of agents such that $G \subseteq Agt$.

$E_G\varphi$ represents that everyone in group G knows φ and $C_G\varphi$ represents that it is *commonly known* in group G that φ is true, which means that everyone knows φ, and everyone knows that everyone knows φ, *ad infinitum*. $D_G\varphi$ represents *distributed* knowledge, which means we combine the knowledge of the set of agents G such that G knows φ, even though it may be that no individual in the group knows φ.

The semantics for these group operators are defined as follows:

- $(M, s) \vDash E_G\varphi$ iff $(M, s) \vDash K_i\varphi$ for all $i \in G$;

- $(M, s) \vDash C_G\varphi$ iff $(M, t) \vDash \varphi$ for all t that are G-reachable from s

- $(M, s) \vDash D_G\varphi$ iff $(M, t) \vDash \varphi$ for all t such that $(s, t) \in \cap_{i \in G}\mathcal{K}_i$

By definition, $(M, s) \vDash E_G\varphi$ will be true, if and only if, φ is known by all agents in G. Common knowledge in world s, $(M, s) \vDash C_G\varphi$ is defined as: in all worlds t, which are reachable by following the accessibility relations of all agents in G, φ is true. For distributed knowledge, $(M, s) \vDash D_G\varphi$ is true, if and only if, in all the possible worlds that any one agent from G consider possible, φ is true; in other words, we eliminate worlds that any agent in G knows to be impossible.
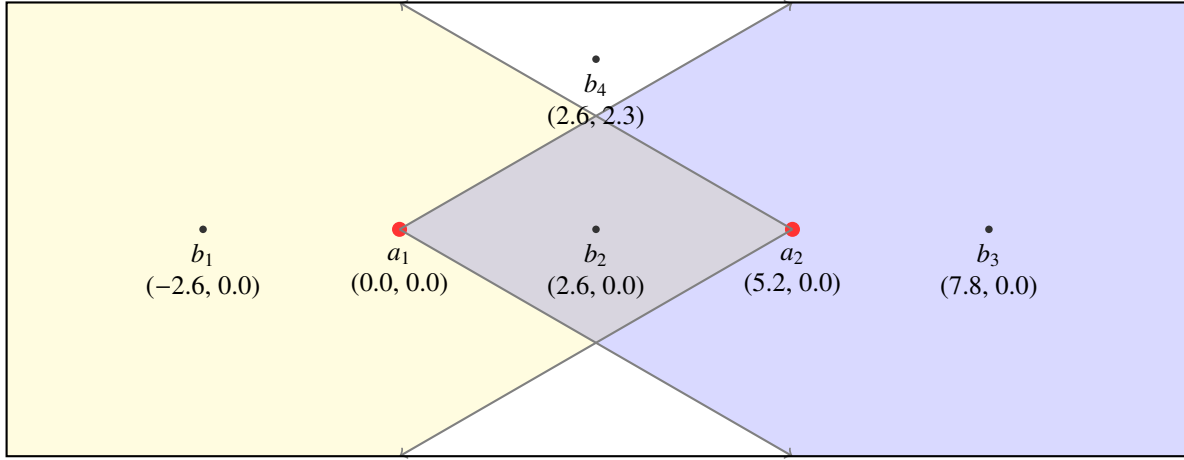
Figure 1. Example for big brother logic

2.2.1. Seeing and Knowledge

Recently Gasquet et al.[14] noted the relationship between what an agent sees and what it knows. They define a more specific task of logically modeling and reasoning about cooperating tasks of vision-based agents, which they called *Big Brother Logic* (BBL). Their framework models multi-agent knowledge in a continuous environment of vision, which has great potential applications such as reasoning over cameras inputs, autonomous robots and vehicles. They introduced the semantics of their model and its extensions on natural geometric models.

In their scenario, agents are stationary cameras in a Euclidean plane $\mathbb{R}^2$, and they set the assumptions that those cameras can see anything in their sight range, and they do not have volume, which means they would not block others' sight. They extend Fagin et al.'s logic [9] by noting that, at any point in time, what an agent knows, including nested knowledge, can be derived directly from what it can see in the current state. In brief, instead of Kripke frames, they define a *geometric model* as (pos, dir, ang), in which:

- $pos : Agt \rightarrow \mathbb{R}^2$

- $dir : Agt \rightarrow U$

- $ang : Agt \rightarrow [0, 2\pi)$

in which U is the set of unit vectors of $\mathbb{R}^2$, the pos function gives the position of each agent, the dir function gives the direction that each agent is facing, and the function ang gives the angle of view for each agent.

A model is defined as (pos, ang, D, R), in which pos and ang are as above, D is the set of dir functions and R the set of equivlance relations, one for each agent a, defined as:

$$R_a = \{(dir, dir') \in D^2 \mid \text{for all } b \neq a, dir(b) = dir'(b)\}$$

In this context, standard propositional logic is extended with the binary operator $a \triangleright b$, which represents that "a sees b". This is defined as:

- $(pos, ang, D, R), dir \models a \triangleright b$ iff $pos(b) \in C_{pos(a), dir(a), ang(a)}$,

in which $C_{pos(a), dir(a), ang(a)}$ is the field of vision that begins at $pos(a)$ from direction $dir(a)$ and goes $ang(a)$ degrees in a counter-clockwise direction.

Figure 1 shows an example with two agents, a_1 and a_2, and model $((0.0, 0.0), 60°, D, R)$ and $((5.2, 0.0), 60°, D, R)$ respectively, along with four objects, b_1, b_2, b_3 and b_4. Based on the current state, for agent a_1, we have:
$(pos, ang, D, R), dir \models a_1 \triangleright a_2$; $(pos, ang, D, R), dir \models a_1 \triangleright b_2$; and, $(pos, ang, D, R), dir \models a_1 \triangleright b_3$.

From this, they show the relationship between seeing and knowing. For example, $K_a(b \triangleright c)$ is defined as $a \triangleright b \wedge a \triangleright c \wedge b \triangleright c$.

Gasquet et al. [14] also define a common knowledge operator, defined in a similar manner to that of Fagin et al.'s definition based on G-reachable worlds [9]. In Figure 1, the formula $C_{\{a_1,a_2\}}loc(b_2)=(2.6,0.0)$ holds because a_1 and a_2 can both see b_2, and can both see that each other sees b_2, etc.

Cooper et al. [6] adopted Gasquet et al. [14]'s idea of modelling an agent's knowledge based on what it sees, and generalised it to seeing propositions, rather than to just to seeing other agents in a Euclidean plane. They extended Fagin et al.'s definition by adding an extra formula α, which represents formula about what can be seen:

$$\alpha \quad ::= \quad p \mid S_i p \mid S_i \alpha$$
$$\varphi \quad ::= \quad \alpha \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid K_i\varphi,$$

in which $p \in PVar$ (the set of propositional *variables*) and $i \in Agt$. The grammar for α defines how to represent visibility relations. $S_i\alpha$ can be read as "agent i sees α". Note the syntactic restriction that agents can only see atomic propositions or nestings of seeing relationships that see atomic propositions.

From this, they define knowledge using the equivalences $K_i p \leftrightarrow p \wedge S_i p$ and $K_i \neg p \leftrightarrow \neg p \wedge S_i p$. This tight correspondence between seeing and knowledge is intuitive: an agent knows p is true if p is true and the agent can see the variable p. Such a relationship is the same as the relationship between knowing something is true and *knowing whether* something is true [32, 10]. In fact, in early drafts of the work available online, Cooper et al. [6], S_i was written KW_i and was called the "knowing whether" operator.

Comparing these two bodies of work, Gasquet et al. [14] use a geometric model to represent the environment and derive knowledge from this by checking the agents' line of sight. Their idea is literally matching the phrase "Seeing is believing". However, their logic is constrained only to vision in physical spaces. While in Cooper et al. [6]'s world, the seeing operator applies to propositional variables, and thus visibility can be interpreted more abstractly; for example, 'seeing' (hearing) a message over a telephone.

The current paper generalises seeing relations to perspective functions, which are domain-dependent functions defining what agents see in particular states. The result is more flexible than seeing relations, and allows Big Brother Logic to be defined with a simple perspective function, as well as new perspective functions for other domains; for example, Big Brother Logic in three-dimensional planes, or visibility of messages on a social network.

2.3. Epistemic Planning

Bolander and Andersen [4] introduce the concept of epistemic planning, and define it in both single agent and multi-agent domain. Their planning framework is defined in *dynamic epistemic logic* (DEL) [38], which has been shown to be undecidable in general, but with decidable fragements [5]. In addition, they also provided a solution to PSPACE-hardness of the plan verification problem. This formalism has been used to explore theoretical properties of epistemic planning; for example, Engesser et al. [8] used concepts of perspective shifting to reason about other's contribution to joint goals. Along with implicit coordination actions, their model can solve some problems elegantly without communication between agents.

Since epistemic planning was formalized in DEL, there has been substantial work on DEL-based planning. However, in this paper, our focus is on the design, implementation, and evaluation of planning tools, rather than on logic-based models of planning. In this section, we focus on research in epistemic planning that is focused on the design, implementation, and evaluation of planning algorithms.

A handful of researchers in the planning field focus on leveraging existing planners to solve epistemic problems. Muise et al. [33] proposed an approach to multi-agent epistemic planning with nested belief, non-homogeneous agents, co-present observation, and the ability for one agent to reason as if it were the other. Generally, compiled an epistemic logic problem into a classical planning problem by grounding epistemic fluents into propositional fluents and using additional conditional effects of actions to enforce desirable properties of belief. They evaluated their approach on the *Corridor* [24] problem and the *Grapevine* problem, which is a combination of *Corridor* and *Gossip* [21]. Their results show that their approach is able to solve the planning task within a typically short time, but the compilation time to generate fluents and conditional effects is exponential in the size of the original problem.

Simultaneously, Kominis and Geffner [24] adopted methods from partially-observable planning for representing beliefs of a single agent, and converted that method to handle multi-agent setting. They define three kinds of actions sets O, N and U. O represents all physical actions, which is the same O as in classical planning. The actions set N denotes a set of sensing actions, which can be used to infer knowledge. The last action set U is used to update

any of the fact φ. They are able to use their model to encode epistemic planning problems, and can convert their model to a classical planning model using standard compilation techniques for partially-observable planning. They evaluated their model on *Muddy Children*, *Sum* and *Word Rooms* [24] domains. Their results show that their model is able to handle solve all cases presented, however, the performance as measured by time and depth of the optimal plan is indirectly related to the problem scope and planning algorithm. For example, by using two different planners, $A^*(h_{max})$ and BFS(h_{add}), the *Muddy Children* test case with seven children, the time consumed by BFS(h_{add}) is only half of $A^*(h_{max})$, while for other problems, there is little difference between the two.

Since Kominis and Geffner, and Muise et al.'s approach the problem in a similar way (compilation to classical planning) their results are similar in a general way. However, the methods they used are different, so, their work and results have diverse limitations and strengths. For Muise et al.'s work, they managed to modeling nested beliefs without explicit or implicit Kripke structures, which means they can only represent literals, while Kominis and Geffner's work is able to handle arbitrary formulae. Furthermore, Muise et al.'s model does not have the strict common initial knowledge setting found in Kominis and Geffner, and does not have the constraint that all action effects are commonly known to all the agents. Therefore, Muise et al.'s model allows them to model beliefs, which might be not what is actual true in the world state, rather just model knowledge. In other words, they can handle different agents having different belief about the same fluent.

More recently, rather than compiling epistemic planning problems into classic planning, Huang et al. built a native multi-agent epistemic planner, and proposed a general representation framework for multi-agent epistemic problems [23]. They considered the whole multi-agent epistemic planning task from a third person point of view. In addition, based on a well-established concepts of believe change algorithms (both revision and update algorithms), they design and implemented an algorithm to encode belief change as the result of planning actions. Catering to their ideas and algorithms, they developed a planner called MEPK. They evaluated their approach with *Grapevine*, *Hexa Game* and *Gossip*, among others. From their results, it is clear that their approach can handle a variety of problems, and performance on some problems is better than other approaches. While this approach is different from Kominis and Geffner and Muise et al., it still requires a compilation phase before planning to re-write epistemic formula into a specific normal form called *alternating cover disjunctive formulas* (ACDF) [18]. The ACDF formula is worst-case exponentially longer than the original formula. The results show that this step is of similar computational burden as either Kominis and Geffner or Muise et al.. In addition, building a native planner to solve an epistemic planning problem makes it more difficult to take advantage of recent advances in other areas of planning.

3. A Model of Epistemic Planning using Perspectives

In this section, we define a syntax and semantics of our *agent perspective model*. Our idea is based on that of Big Brother Logic [14] and Cooper et al.'s seeing operators [6]. The syntax and semantics of our model is introduced, including distributed knowledge and common knowledge.

3.1. Language

Extending Cooper et al.'s [6] idea of seeing propositional variables, our model is based on a model of functional STRIPS (F-STRIPS) [13], which uses variables and domains, rather than just propositions. We allow agents to see variables with discrete and continuous domains.

3.1.1. Model

We define an epistemic planning problem as a tuple (Agt, V, D, O, I, G), in which Agt is a set of agents, V is a set of variables, D stands for domains for each variable, in which domains can be discrete or continuous, I and G are the initial state and goal states respectively, and both of them are also bounded by V and D. Specifically, they should be assignments for some or all variables in V, or relations over these. O is the set of operators, with argument in the terms of variables from V.

3.1.2. Epistemic Formulae

DEFINITION 3.1.1:

Goals, actions preconditions, and conditions on conditional effects, are epistemic formulae, defined by the following grammar:

$$\varphi ::= R(v_1, \ldots, v_k) \mid \neg\varphi \mid \varphi \wedge \varphi \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid S_i v \mid S_i\varphi \mid K_i\varphi,$$

in which R is k-arity 'domain-dependent' relation formula, $v \in V$, $S_i v$ and $S_i\varphi$ are both are visibility formulae, and $K_i\varphi$ is a knowledge formula.

3.1.3. Domain Dependent Formulae

Domain-dependent formula not only including basic mathematical relations, but also relational terms defined by the underlying planning language can have the relation between variables. For example, based on the scenario from Figure 1, $loc(a_1) = (0, 0)$ is a true formula expression the location of agent a_1, while $loc(a_1) = loc(a)$ is a false. In Section 4, we discuss the use of *external functions* in F-STRIPS in our implemented planning, which allow more complex relations, or even customized relations, as long as they have been defined in the external functions. For example, we can define a domain dependent relation in external function to compare distance between objects, called @$far\_away(loc(i), loc(j), loc(k))$. This external function takes three coordinates as input, and returns a Boolean value, whether distance between i and j is longer than i and k. In the scenario displayed in Figure 1, the relation @$far\_away(loc(a_1), loc(b_4), loc(b_2))$ would be true, while relation @$far\_away(loc(a_1), loc(b_1), loc(b_2))$ would be false, since b_1 and b_2 are equally close to a_1. The definition of this function is delegated to a function implemented in a programming language such as C++, and the planner is unaware of its semantics. However, for the remainder of this section, we will ignore the existence of external functions, and return to them in our implementation section.

3.1.4. Visibility Formula

An important concept adapted from Cooper et al. [6] is "seeing a proposition". Let p be a proposition, "agent i knows whether p" can be represented as "agent i sees p". Their interpretation on this is: either p is true and i knows that; or, p is false and i knows that. With higher-order observation added, it gives us a way to reason about others' epistemic viewpoint about a proposition without actually knowing whether it is true. Building on this concept, our 'seeing' operator allows us to write formulae about visibility: $S_i v$ and, $S_i\varphi$.

The seeing formulae represent two related interpretations: seeing a variable; or seeing a formula. The formula $S_i v$ can be understood as variable v has some value, and no matter what value it has, agent i can see the variable and knows its value. On the other hand, seeing a relation is trickier. $S_v\varphi$ can be interpreted as: one relation ψ is a formula and no matter whether is true or false, i knows whether it is true or not. To make sure i knows whether φ is true or not, the evaluation for this seeing formula is simply that agent i sees the variables in that relation.

For example, in Figure 1, using the notation defined in Section 3.1.1, $S_{a_1}loc(b_1)$ can be read as "agent a_1 sees variable b_1". In the case of seeing a domain-dependent formula, $S_{a_1}(far\_away, loc(a_1), loc(b_4), loc(b_2))$ can be read as "agent a_1 sees whether the relation $far\_away(loc(a_1), loc(b_4), loc(b_2))$ is true or not", which is: "agent a_1 sees whether b_4 is farther away from a_1 than b_2."

3.1.5. Knowledge Formula

In addition to the visibility operator, our language supports the standard knowing operator K_i. Following Cooper et al.'s idea on defining knowledge based on visibility [6], we define knowledge as: $K_i\varphi \leftrightarrow \varphi \wedge S_i\varphi$. That is, for i to know φ is true, it needs to be able to see φ, and φ needs to be true. In other words, if you can see something and it is true, then, you know it is true.

3.2. Semantics

We now formally define the semantics of our model.

3.2.1. Knowledge Model

Our model decomposes the planning model from the knowledge model, and as we will see in our implementation, our planner delegates the knowledge model to an external solver. Therefore, in this section, we define the semantics of that knowledge solver. The novel part of this model is the use of perspective functions, which are functions that define which variables an agent can see, instead of Kripke structures. From this, a rich knowledge model can be built up independent of the planning domain.

DEFINITION 3.2.1:

A model M is defined as $M = (V, D, \pi, f_1, \ldots, f_n)$, in which V is a set of variables, D is a function mapping domains for each variable, π is the evaluation function, which evaluates variables and formulae based on the given state, and $f_1, \ldots, f_n : S \rightarrow S$ are the agents' *perspective functions* that given a state s, will return the local state from agents' perspectives.

A state s is a tuple of variable assignments, denoted $[v_1 = e_1, \ldots, v_k = e_k]$, in which $v_i \in V$ and $e_i \in D(V)$ for each i. The *global state* is a complete assignment for all variables in V. A *local state*, which represents an individual agent's perspective of the global state, can be a partial assignment. The set of all states is denoted S.

A perspective function, $f_i : S \rightarrow S$ is a function that takes a state and returns a subset of that state, which represents the part of that state that is visible to agent i. These functions can be nested, such that $f_j(f_i(s))$ represents agent i's perspective of agent j's perspective, which can be just a subset of agent j's actual perspective. The following properties must hold on $f_1, \ldots, f_n$ for all $i \in Agt$ and $s \in S$:

1. $f_i(s) \subseteq s$
2. $f_i(s) = f_i(f_i(s))$

First, we give the definition for propositional formulae. Let any variable be denoted as v_i, and R any k-ary domain dependent formula:

- $(M, s) \vDash R(v_1, \ldots, v_k)$ iff $\pi(s)(R(\pi(s)(v_1), \ldots, \pi(s)(v_k))) = true$

Relations are handled by evaluation function $\pi(s)$. The relation R is evaluated by getting the value for each variable in s, and checking whether R stands or not. Other propositional operators are defined in the standard way.

Then, we have the following formal semantics for visibility:

- $(M, s) \vDash S_i v$ iff $\exists (v = x) \in f_i(s)$ for any value x

- $(M, s) \vDash S_i R(v_1, \ldots, v_k)$ iff $\forall v \in \{v_1, \ldots, v_k\}, (M, s) \vDash S_i v$

- $(M, s) \vDash S_i \neg \varphi$ iff $(M, s) \vDash S_i \varphi$

- $(M, s) \vDash S_i (\varphi \land \psi)$ iff $(M, s) \vDash S_i \varphi$ and $(M, s) \vDash S_i \psi$

- $(M, s) \vDash S_i S_j \varphi$ iff $(M, f_i(s)) \vDash S_j \varphi$

- $(M, s) \vDash S_i K_j \varphi$ iff $(M, f_i(s)) \vDash K_j \varphi$

$S_i v$, read "Agent i sees variable v", is true if and only if v is visible in the state $f_i(s)$. That is, an agent sees a variable if and only if that variable is in its perspective of the state. Similarly, an agent knows whether a domain-dependent formula is true or false if and only if it can see every variable of that formula. For example, in Figure 1, $S_{a_1} b_1$ is false and $S_{a_1} b_2$ is true, which is because b_2 is in a_1's perspective (blue area), while b_1 is not. The remainder of the definitions simply deal with logical operations. However, note the definition of $S_i \neg \phi$, which is equivalent to $S_i \phi$, because all variables are the same. This effectively just defines that 'seeing' a formula means seeing its variables.

Now, we define knowledge as:

- $(M, s) \vDash K_i \varphi$ iff $(M, s) \vDash \varphi$ and $(M, s) \vDash S_i \varphi$

This definition follows Cooper et al.'s definition [6]: agent i knows φ if and only if the formula is true at (M, s) and agent i sees it. Using the same example as previously, $K_{a_1} @far\_away(loc(a_1), loc(b_2), loc(b_3))$ is false, while $K_{a_1} @far\_away(loc(a_1), loc(b_3), loc(b_2))$ is true.

Theorem 3.1. The S5 axioms of epistemic logic hold in this language. That is, the following axioms hold:

(K)	$K_i(\varphi \to \psi)$	$\to$	$K_i\varphi \to K_i\psi$
(T)	$K_i\varphi$	$\to$	φ
(4)	$K_i\varphi$	$\to$	$K_iK_i\varphi$
(5)	$\neg K_i\varphi$	$\to$	$K_i\neg K_i\varphi$

Proof. We first consider axiom (T). By our semantics, $(M, s) \vDash K_i \varphi$ is true, if and only if, both $(M, s) \vDash \varphi$ and $(M, s) \vDash S_i\varphi$ are true. Therefore, it is trivial that axiom (T) holds.

For (K), based on our definition of knowledge, we have $(M, s) \vDash K_i(\varphi \to \psi)$ is equivalent to $(M, s) \vDash S_i(\varphi \to \psi)$ and $(M, s) \vDash (\varphi \to \psi)$. Then, by our semantics, we have that $(M, s) \vDash S_i(\varphi \to \psi)$ is equivalent to $(M, s) \vDash S_i\neg\varphi$ or $(M, s) \vDash S_i\psi$. From propositional logic, $\varphi \implies \psi$ is equivalent to $\neg\varphi \lor \psi$. We combine $(M, s) \vDash \neg\varphi$ with $(M, s) \vDash S_i\varphi$ to get $(M, s) \vDash \neg K_i\varphi$ and similarly for ψ to get $(M, s) \vDash K_i\psi$, which is equivalent to $(M, s) \vDash K_i\varphi \to K_i\psi$ from propositional logic.

To prove (4) and (5), we use the properties of the perspective function f_i. The second property shows, a perspective function for agent i on state s converges after the first nested iteration, which means $(M, f_i(s)) \equiv (M, f_i(f_i(s)))$. Therefore, whenever $(M, f_i(s)) \vDash \psi$, then ψ also holds in $(M, f_i(f_i(s)))$, implying that $K_i\psi$ holds too. This is the case when ψ is $K_i\varphi$ or $\neg K_i\varphi$, so (4) and (5) hold. $\square$

3.3. Group Knowledge

From the basic visibility and knowledge definitions, in this section, we define group operators, including distributed and common visibility/knowledge.

3.3.1. Syntax

We extend the syntax of our language with group operators:

$$\varphi ::= \psi \mid \neg\varphi_1 \mid \varphi_2 \land \varphi \mid ES_G\alpha \mid EK_G\varphi \mid DS_G\alpha \mid DK_G\varphi \mid CS_G\alpha \mid CK_G\varphi,$$

in which G is a set of agents and α is a variable v or formula φ.

Group formula $ES_G\alpha$ is read as: everyone in group G sees variable/formula α, and $EK_G\varphi$ represents that everyone in group G knows φ. DK_G is the distributed knowledge operator, equivalent to D_G in Section 2.2, while DS_G is its visibility counterpart: someone in group G sees. Finally, CK_G is common knowledge and CS_G common visibility: "it is commonly seen".

3.3.2. Semantics

Let G be a set of agents, φ a formula, and α either a formula or a variable, then we can define the semantics of these group formula as follows:

- $(M, s) \vDash ES_G \alpha$ iff $\forall i \in G, (M, s) \vDash S_i \alpha$

- $(M, s) \vDash EK_G \varphi$ iff $\forall i \in G, (M, s) \vDash K_i \varphi$

- $(M, s) \vDash DS_G \alpha$ iff $(M, s') \vDash \alpha$, where $s' = \bigcup_{i \in G} f_i(s)$

- $(M, s) \vDash DK_G \varphi$ iff $(M, s) \vDash \varphi$ and $(M, s) \vDash DS_G \varphi$

- $(M, s) \vDash CS_G \alpha$ iff $(M, s') \vDash \alpha$, where $s' = fc(G, s)$

- $(M, s) \vDash CK_G \varphi$ iff $(M, s) \vDash \varphi$ and $(M, s) \vDash CS_G \varphi$,

in which $fc(G, s)$ is state reached by applying composite function $\bigcap_{i \in G} f_i$ until it reaches its fixed point. That is, the fixed point s' such that $fc(G, s') = fc(G, \bigcap_{i \in G} f_i(s'))$.

Reasoning about common knowledge and visibility is more complex than other modalities. Common knowledge among a group is not only everyone in the group shares this knowledge, but also everyone knows other knows this

knowledge, and so on, *ad infinitum*. The infinite nature of this definition leads to some definitions that are untractable in some models.

However, due to our restriction on the definition of states as variable assignments and our use of perspective functions, common knowledge is much simpler. Our intuition is based on the fact that each time we apply composite perspective function $\bigcap_{i \in G} f_i(s)$, the resulting state is either a proper subset of s (smaller) or is s. By this intuition, we can limit common formula in finite steps.

The fixed point is a recursive definition. However, the following theorem shows that this fixed point always exists (even if it is empty), and the number of iterations is bound by the size of $|s|$, the state to which it is applied.

Theorem 3.2. Function $fc(G, s)$ converges on a fixed point $s' = fc(G, s')$ within $|s|$ iterations.

Proof. First, we prove convergence is finite; that is, when $s' = fc(G, s')$, further 'iterations' will result in s'; that is, $s' = fc(G, fc(G, s'))$. Let $s_i = fc(G, s_i)$, where i is the number of iterations. Then, we have $s_{i+1} = fc(G, s_i) = s_i$. Since $s_{i+1} = s_i$, we have $s_{i+2} = fc(G, s_i)$, which means $s_{i+2} = s_i$. Via induction, we have that for all $k \geq i$, $s_k = s_i$. Therefore, once we reach convergence, it remains.

Next, we prove convergence within $|s|$ iterations. By the intuition and definition of the perspective functions, $f_k(s) \subseteq s$, and $s_{i+1} = (\bigcap_{k \in G} f_k(s_i))$, we have $s_{i+1} \subseteq s_i$. Then, as we prove for the first point, if $s_i = s_{i+1}$, then we have reached a fixed point and no further iterations are necessary. Therefore, the worst case is when $s_{i+1} \subsetneq s_i$ and $|s_i| - |s_{i+1}| = 1$. There are most $|s|$ such worst case iterations until s_i converges on an empty set. Therefore, the maximum number of iterations is $|s|$. □

For each of the iteration, there are $|G|$ local states in group G that need to be applied in the generalised union calculation, which can be done in polynomial time, and there there are at most $|s|$ steps. So, a poly-time algorithm for function fc exists.

3.4. A brief note on expressiveness

The intuitive idea about perspective functions is based on what agents can see, as determined by the current state. The relation between $t = f_i(s)$ and s corresponds roughly to accessibly relations $(s, t) \in \mathcal{K}_i$ in Kripke semantics. However, only focusing on what agent exactly knows/sees means overlooking those variables that agents are uncertain about. Perspective functions return one partial world that the agent i is certain about, rather than a set of worlds that i considers possible. The advantage is that applying a perspective function provides us only one state, rather than multiple states in Kripke semantics, preventing explosions in model size.

However, the reduced complexity loses information on the "unsure" variables. Theoretically, $t = f_i(s)$ from our model is a set intersection from all the t that $(s, t) \in \mathcal{K}_i$. This eliminates disjunctive knowledge about variables; the only uncertainty being that an agent does not see a variable. For example, in the well-known *Muddy Child* problem, the knowledge is not only generated by what each child can see by the others' appearance, which is modelled straightforwardly using perspective functions, but also can be derived from the statements made by their father and the response by other children. From their perspective, they would know exactly m children are dirty, which can be handled by our model, as they are certain about it. While by the k-th times the father asked and no one responds, they can use induction and get the knowledge that at least k children are dirty. By considering that there are two possible worlds, where the number of dirty children is m or $m + 1$, Kripke structures keep both possible worlds until $m + 1$ steps. Our model cannot keep this.

Therefore, although our model can handle preconditions and goals with disjunction, such as $K_i (v = e_1) \vee (v = e_2)]$, it cannot *store* such disjunction in its 'knowledge base'. Rather, agent i knows v's value is e_1 or its knows it is e_2.

Despite this, possible worlds can be *represented* using our model, using the same approach taken by Kominis and Geffner [24] of representing Kripke structures in the planning language. We can then define perspective functions that implement Kripke semantics.

To summarise Section 3, we have defined a new model of epistemic logic, including group operators, in which states in the model are constrained to be just variable assignments. The complexity of common knowledge in this logic is bound by the size of the state and number of agents. In the following section, we show how to implement this in any F-STRIPS planner that supports external functions.

4. Implementation

To validate our model and test its capabilities, we encode it within a planner and solve on some well-known epistemic planning benchmarks. Two key aspects in planning problem solving is the planning language and solver. Since, as mentioned in section 2.1, we use BFWS(f_5)[13], the advantage of F-STRIPS with external functions allows us to decompose the planning task from the epistemic logic reasoning. In this section, we encode our model into the F-STRIPS planning language and explain our use of external functions for supporting this.

4.1. F-STRIPS encoding

Any classical F-STRIPS [13] problem can be represented by a tuple $(V, D, O, I, G, \mathbb{F})$, where V and D are variables and domains, O, I, G are operators, initial state, and goal states. $\mathbb{F}$ stands for the external functions, which allow the planner to handle problems that cannot be defined as a classical planning task, such as, the effects are too complex to be modelled by propositional fluents, or even the actions and effects has some unrevealed corresponding relations. In our implementation, external functions are implemented in C++, allowing any C++ function to be executed during planning.

In our epistemic planning model (Agt, V, D, O, I, G) defined in Section 3, Agt is a set of agent identifiers, and V and D are exactly same as in F-STRIPS. Operators O, initial states I and goal states G differ to F-STRIPS only in that they contain epistemic formulae in preconditions, conditions, and goals.

4.1.1. Epistemic Formula in Planning Actions

There are two major ways to include epistemic formula in planning, using the formula as preconditions and conditions (on conditional effects) in operators, or using as epistemic goals.

Setting desirable epistemic formulae as goals is straightforward. For example, in Figure 1, if we want agent a_1 to know a_2 sees b_1, we could simply set the goal be $K_{a_1} S_{a_2} b_1$. However, there are some other scenarios that cannot be modeled by epistemic goals, including temporal constraints such as "agent a_1 sees b_2 all the time" or "target b_4 needs to secretly move to the other side without been seeing by any agents". Each of those temporal epistemic constraints can be handled by a Boolean variable and getting the external functions to determine whether the constraint holds at each state of the plan. If the external function returns false, we 'fail' that plan.

In addition, epistemic formula can be in the precondition of the operators directly. For example, in Figure 1, if the scenario is continued surveillance of b_2 over the entire plan, then the operator **turn**(a_1, d) would have that either $S_{a_1} b_2 \lor S_{a_2} b_2$ as the precondition, after a_1 turns d degree as one of the preconditions.

As for the encoding O, I and G, besides the classical F-STRIPS planning parts, all the formulae are encoded as calls to external functions.

4.2. External Functions

External functions take variables as input, and return the evaluated result based on the current search state. It is the key aspect that allows us to divide epistemic reasoning from search. Therefore, unlike compilation approaches to epistemic planning that compile new formula or normal forms that may never be required, our epistemic reasoning using lazy evaluation, which we hypothesise can significantly reduce the time complexity for most epistemic planning problems.

4.2.1. Agent perspective functions

As briefly mentioned in Section 3.2.1, the perspective function, $f_i : S \rightarrow S$, is a function that takes a state and returns the local state from the perspective of agent i. Compared to Kripke structures, the intuition is to only define which variables an agent sees. Individual and group knowledge all derive from this.

Once we have domain-specific perspective functions for each agent, or just one implementation for homogenous agents, our framework implementation takes care of the remaining parts of epistemic logic. Given a domain-specific perspective function, we have implemented a library of external functions that implement the semantics of K_i, DS_i, DK_i, CS_i, and CK_i, using the underlying domain-specific perspective functions. The modeller simply needs to provide the perspective function for their domain, if a suitable one is not already present in our library.

For example, in Figure 1, global state s covers the whole flat field. $f_{a_1}(s)$ is the blue area, and $f_{a_2}(s)$ is the yellow area, which means in agent a_1's perspective, the world is the blue part, and for agent a_2, the world is only the yellow

part. Further more, in agent a_1's perspective, what agent a_2 sees, can be represented by the intersection between those two coloured areas, which is actually $f_{a_2}(f_{a_1}(s))$. The interpretation is that, agent a_1 only considers state $l = f_{a_1}(s)$ is the "global state" for him, and inside that state, agent a_2's perspective is $f_{a_2}(l)$.

To be more specific about perspective functions, assume the global state s contains all variables for $\{a_1, a_2, b_1, b_2, b_3, b_4\}$, such as locations, the directions agents are facing, etc. Based on the current set up from Figure 1, we can implement f_i for any agent i with the following Euclidean geometric calculation:

$$[|\arctan(\frac{|y(a_1) - y(a_2)|}{x(a_1) - x(a_2)}) - dir(a_1)| \leq \frac{range(a_1)}{2}] \bigoplus [|\arctan(\frac{|y(a_1) - y(a_2)|}{x(a_1) - x(a_2)}) - dir(a_1)| \geq \frac{360 - range(a_1)}{2}] \quad (1)$$

The perspective function takes all agents' locations, directions and vision angles, along with target location as input, and returns those agents and objects who fall instead these regions. On each variable for both agents, $f_{a_1}(s) = \{a_1, a_2, b_2, b_3\}$ and $f_{a_2}(s) = \{a_1, a_2, b_1, b_2\}$. Our library can then reason about knowledge operator.

While such an approach could be directly encoded using proposition in classical planning, we assert that the resulting encoding would be tediously difficult and error prone.

Their distributed knowledge would be derived from the union over their perspectives, which is the whole world s except all variables for b_4. Both agents would know every variables in the intersection of their perspectives, $\{a_1, a_2, b_2\}$, while it is also the same as what they commonly knows.

However, if we alter the scenario a bit by turning a_1 180°, then the *EK* would work similar as one level perspective functions, while *CK* is empty. In the new scenario, $f_{a_1}(s)$ would become $\{a_1, b_1\}$, and *EK* becomes $\{a_1, b_1\}$. When finding the converged perspective for a_1 and a_2, $f_{a_1}(\{a_1, b_1\})$ stays the same, but the perspective for a_2 results in an empty world. The intersection over their perspective does not have a_2, which means a_2 does not exist in at least one of the agent's (a_1) perspectives from the group.

However, the above implementation of the perspective function is only useful for Euclidean planes. One of the novelties in this paper is that we define our implementation to support arbitrary perspective functions, which can be provided by the modeller as new external functions. Therefore, an agent's perspective function would be able to handle any problem with a proper set of visibility rules to be defined. Basically, in our implementation, a perspective function can take any state variable from the domain model, and converts it into its the agent's perspective state. Then, following the property that $f_i(s) \subseteq s$, the function applies the domain-specific visibility rules on all the variables to gain the agent's local state. From there, the semantics outlined in Section 3 are handled by our library.

Therefore, we can simply define our external function on different epistemic problem by providing different implementations of f_i. For example, the only difference between the external function's implementation on Corridor and Grapevine benchmarks from Section 5, is that in Corridor, the visibiility rules are that an agent can see within the current room and the adjacent room, while in Grapevine, the agent can only see within the current room. Moreover, in order to implement epistemic planning problems for e.g. a thtee-dimensional Euclidean plane, we need to modify the perspective functions based on the geometric model.

From a practical perspective, this means that modellers provide: (1) a planning model that uses epistemic formula; and (2) domain-specific implementations for $f_1, \ldots, f_n$. Linking the epistemic formula in the planning model to the perspective functions is delegated to our library.

4.2.2. Domain dependent functions

Domain dependent functions are customised relations for each set of problems, corresponding to $R(v_1, \ldots, v_k)$ in Section 3, and can be any domain specific functions that are implementable as external functions.

Overall, the implementation of our model is not as direct as other classical F-STRIPS planning problems or epistemic planning problems. However, as we demonstrate in the next section, it is scalable and flexible enough to find valid solutions to many epistemic planning problems.

5. Experiments & Results

In this section, we evaluate our approach on four problems: *Corridor* [24], *Grapevine* [33], *Big Brother Logic (BBL)* [14], and *Social-media Network* (*SN*). Corridor and Grapevine are well-known epistemic planning problems, which we use to compare actual performance of our model against an existing state-of-the-art planner. BBL is a model

of the Big Brother Logic in a two-dimensional continuous domain, which we used to demonstrate the expressiveness of our model. In addition, to demonstrate our model's capability of reasoning about group knowledge, we inherit the classical Gossip Problem, and create our own version, called Social-media Network. Hereafter, we assume any knowledge formula $K_a v$ is supplied with correct value e, which means its equivalent with $K_a v = e$, unless the value is specified.

5.1. Benchmark problems

In this section, we briefly describe the corridor and grapevine problems, which are benchmark problems that we use to compare against Muise et al.'s epistemic planner [33], which compiles to classical planning.

Corridor. The corridor problem was originally presented by Kominis and Geffner[24]. It is about selective communication among agents. The basic set up is in a corridor of rooms, in which there are several agents. An agent is able to move around adjacent rooms, sense the secret in a room, and share the secret. The rule of communication is that when an agent shares the secret, all the agents in the same room or adjacent rooms would know.

The goals in this domain are to have some agents knowing the secret and other agents not knowing the secret. Thus, the main agent needs to get to the right room and communication to avoid the secret being overheard.

Grapevine. Grapevine, proposed by Muise et al.[33], is a similar problem to Corridor. With only two rooms available for agents, the scenario makes sharing secrets while hiding from others more difficult. The basic set up is each agent has their own secret, and they can share their secret among everyone in the same room. Since there are only two rooms, the secret is only shared within the room. The basic actions for agents are moving between rooms and sharing his secret.

To evaluate computational performance of our model, we compare to Muise et al.[33]'s *PDKB* planner. They have several test cases for Corridor and Grapevine. In addition, to test how the performance is influenced by the problem, we created new problems that varied some of the parameters, such as the number of agents, the number of goal conditions and also depth of epistemic relations.

The *PDKB* planner converts epistemic planning problems into classical planning, which results in a significant number of propositions when the depth or number of agents increased. We tried to submit the converted classical planning problem to the same planner that our model used, $BFWS(R)$ planner, to maintain a fair comparison. However, since the computational cost of the novelty check in $BFWS(R)$ planner increases with size of propositions, the planning costs was prohibitively expensive. Therefore, for comparison, we use the default *ff* planner that is used by Muise et al..

We ran the problems on both planners using a Linux machine with 1 CPU and 10 gigabyte memory. We measure the number of atoms (fluents) and number of nodes generated during the search to compare the size of same problem modelled by different methods. We also measured the total time for both planners to solve the problem, and the time they taken to reasoning about the epistemic relations, which correspond to the time taken to call external functions for our solution (during planning), and the time taken to convert the epistemic planning problem into classical planning problem in the *PDKB* solution (before planning).

We show the results of the problems in Table 1, in which $|a|$ specifies the number of agents, d the maximum depth of a nested epistemic query, $|g|$ the number of goals, $|atom|$ the number of atomic fluents, $|gen|$ the number of generated nodes in the search, and $|calls|$ the number of calls made to external functions.

From the results, it is clear that the complexity of the *PDKB* approach grows exponentially on both number of the agents and depth of epistemic relations (we ran out of memory in the final Grapevine problem), while in our approach, those features do not have a large affect. However, epistemic reasoning in our approach (calls to the external solver), has a significant influence on the performance for our solution. Since the F-STRIPS planner we use checks each goal at each node in the search, the complexity is in $O(|g| * |gen|)$. While this is exponential in the size of the original problem (because $|gen|$ is exponential), the compuational cost is significantly lower than the compilation in the *PDKB* approach.

Problem	Parameters			Our Model					PDKB			
							TIME(s)				TIME(s)	
	$\|a\|$	d	$\|g\|$	$\|atom\|$	$\|gen\|$	$\|calls\|$	Calls	Total	$\|atom\|$	$\|gen\|$	Compile	Total
Corridor	3	1	2	13	15	48	0.001	0.004	54	21	0.148	0.180
	7	1	2	13	15	48	0.002	0.005	70	21	0.186	0.195
	3	3	2	13	15	48	0.003	0.007	558	21	0.635	0.693
	6	3	2	13	15	48	0.005	0.008	3810	21	5.732	6.324
	7	3	2	13	15	48	0.005	0.008	5950	21	9.990	11.13
	8	3	2	13	15	48	0.006	0.009	8778	21	14.14	15.68
	3	4	2	13	15	48	0.006	0.009	3150	21	3.354	3.752
	3	5	2	13	15	48	0.006	0.009	18702	21	25.69	29.54
Grapevine	4	1	2	346	10	48	0.002	0.006	96	22	0.429	0.469
	4	2	2	346	10	48	0.002	0.007	608	5	2.845	3.168
	4	1	4	346	23	144	0.005	0.009	96	11	0.428	0.468
	4	2	4	346	23	144	0.006	0.010	608	11	2.885	3.178
	4	1	8	346	368	1200	0.040	0.047	96	529	0.381	0.455
	4	2	8	346	368	1200	0.050	0.057	608	1234	3.450	4.409
	4	3	8	346	368	1200	0.068	0.073	4704	14	28.66	30.72
	8	1	2	546	18	48	0.003	0.010	312	5	3.025	3.321
	8	2	2	546	18	48	0.003	0.010	4408	5	54.35	58.80
	8	1	4	546	43	144	0.008	0.016	312	11	2.546	2.840
	8	2	4	546	43	144	0.008	0.016	4408	11	55.33	59.78
	8	1	8	546	1854	4528	0.238	0.268	312	2002	2.519	3.752
	8	2	8	546	1854	4528	0.322	0.294	4408	4371	54.90	228.1
	8	3	8	546	1854	4528	0.394	0.421	–	–	–	–

Table 1. Results for the Corridor and Grapevine Problems

5.2. Big Brother Logic

Big Brother Logic (BBL) is a problem that first discussed by Gasquet et al. [14]. The basic environment is on a two-dimensional space called "Flatland" without any obstacles. There are several stationary and transparent cameras; that is, cameras can only rotate, and do not have volume, so they do not block others' vision. In our scenario, we allow cameras to also move in Flatland.

5.2.1. Examples

Let a_1 and a_2 be two cameras in Flatland. Camera a_1 is located at $(5, 5)$, and camera a_2 at $(15, 15)$. Both cameras have an 90° range. Camera a_1 is facing north-east, while camera a_2 is facing south-west. There are three objects with values $o_1 = 1$, $o_2 = 2$ and $o_3 = 3$, located at $(1, 1)$, $(10, 10)$ and $(19, 19)$ respectively. For simplicity, we assume only camera a_1 can move or turn freely, and camera a_2, o_1, o_2 and o_3 are fixed. Figure 2 visualises the problem set up.

This problem can be represented by the tuple $(V, D, O, I, G, \mathbb{F})$, where:

- $V = \{x, y, direction, query\}$

- $D : dom(x) = dom(y) = \{-20, \ldots, 20\}$; $dom(direction) = \{-179, \ldots, 180\}$; and, $dom(query) = \{0, 1\}$

- $O : \mathbf{move}(dx, dy)$ and $\mathbf{turn}(d)$

- $I = \{x = 5, y = 5, direction = 45\}$

- $G = \{query = 1\}$

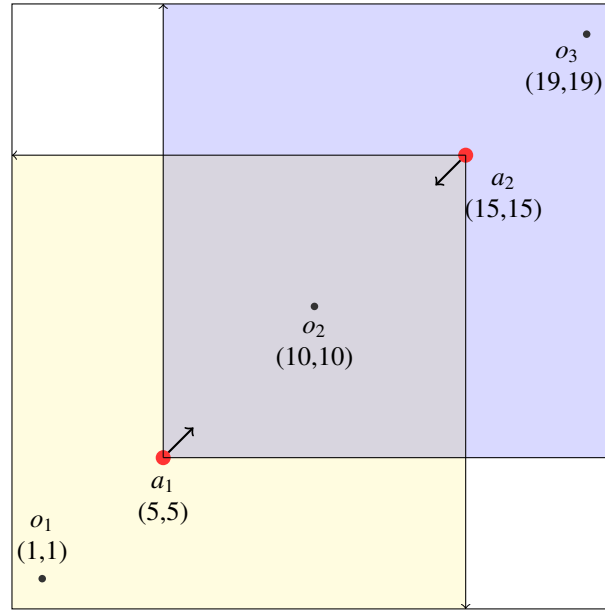- $\mathbb{F} : @\mathbf{check}: query \mapsto \{true, false\},$

Figure 2. Example for Big Brother Logic set up

in which *query* is a goal query, which we describe later.

This is just a simple example to demonstrate our model. Variables x and y represent coordinates of camera a_1, and *direction* determines which way a_1 is facing. Since a_2 and all other objects are fixed, we model them in an external state handled by the external functions, which lightens the domain and reduces the state space. However, we could also model the positions of these as part of the planning model if desired. For the domain of the variables, although the F-STRIPS planner we used does not support using real numbers, using integers is enough to show that our model can work on continuous problems.

We need to check the knowledge queries in the actions (precondition, conditions), or goals. Both action **move**(dx, dy) and action **turn**(d) can change agents' perspectives, and therefore, can influence knowledge. To simplify the problem, instead of moving 1 or turning 1 degree per action, we can set up a reasonable boundary for actions, such as turning ± 45 degree per action and move at speed of 2 for each direction on x and y.

As for the goal conditions, some queries can be achieved for the problem in Figure 2 without doing any actions, such as:

1. Single Knowledge query: $\neg K_{a_2} o_3$; $K_{a_1} o_3$
2. Nested Knowledge query: $\neg K_{a_1} S_{a_2} o_3$; $S_{a_1} S_{a_2} o_3$
3. Group Knowledge query: $K_{a_1,a_2} o_3$; $K_{a_1,a_2} o_2$
4. Distributed Knowledge query: $\neg DK_{a_1,a_2}(o_1, 3)$; $DK_{a_1,a_2}(o_1, 1)$
5. Common Knowledge query: $CK_{a_1,a_2} o_2$; $CK_{a_1,a_2} S_{a_1} o_3$

Most of them are intuitive. From goal 2, although $S_{a_1} S_{a_2} o_3$ is true because a_1 can see a_2's location, range of vision and direction, so a_1 knows whether a_2 can see o_3, the formula $K_{a_1} S_{a_2} o_3$ is false because there is no action that a_1 can do to make a_2 see o_3. For goal 5, $CK_{a,b} S_{a_1} o_3$ is true without any action, because, by calling the external function, the common local state for a_1 and a_2 would be the location of all three values, both a_1 and a_2 and the value of o_2. Then, $S_{a_1} o_3$ would be evaluated true based on the common local state.

In addition, there are some query that would be achieved through valid plans:

1. $K_{a_1,a_2} o_1$: **move**$(-2, -2)$, **move**$(-2, -2)$
2. $CK_{a_1,a_2} o_1$: **move**$(-2, -2)$, **move**$(-2, -2)$
3. $S_{a_2} S_{a_1} o_1$: **move**$(-2, 2)$, **move**$(-2, 2)$

4. $K_{a_1}o_1 \wedge \neg K_{a_2}K_{a_1}o_1$(BBL11): **move**$(-2, 1)$, **move**$(-2, 2)$, **move**$(-1, 2)$, **move**$(0, 2)$, **move**$(0, 2)$, **move**$(0, 2)$, **turn**$(-45)$, **turn**$(-44)$

5. $\neg K_{a_1}S_{a_2}S_{a_1}o_1 \wedge S_{a_1}o_1$(BBL12): **turn**$(-44)$, **turn**$(-45)$, **turn**$(-45)$, **move**$(1, 2)$, **move**$(2, 2)$, **move**$(2, 2)$, **move**$(2, 2)$, **move**$(2, 2)$, **move**$(2, 2)$, **move**$(-1, 2)$

The first one is clear. There is more than one way to let both of them know value o_1, and the planner returns the optimal solution. The second one is also intuitive: to achieve common knowledge in a BBL problem, they need to both see the item and both see each other. The difference between the last two are a bit trickier. To avoid a_2 that knows whether a_1 can see o_1, the cheapest plan returned by planner was for a_1 to move out of a_2's eye sight. The last one is the most difficult to solve. Not only should a_1 see o_1, but also a_1 should know that originally a_2 cannot see that a_1 sees o_1. This is done by decomposing into three facts: "a_1 sees o_1";"a_2 cannot see whether a_1 sees o_1"; and, "a_1 can see that whether a_2 can see whether a_1 sees o_1".

5.2.2. Results

Table 3 shows the results for our problems in the BBL domain. A plan of ∞ means that the problem is unsolvable – no plan exists. While the perspective function in BBL depends on a *geometric model* based on agent's position, direction and facing angle, the results show that with proper usage of our functional STRIPS planner, we can represent continuous domains. Our agent epistemic solver is able to reason about other the agents' epistemic states (vision) and derive plans based on these for non-trivial intricate goal that we believe would be difficult to encode propositionally, demonstrating that our model can handle important problems in vision-based domains.

Problem	Parameters				Performance					Goal
	$|a|$	d	$|g|$	$|p|$	$|gen|$	$|exp|$	$|calls|$	TIME(s) calls	Total	
BBL01	2	1	1	0	1	0	2	0.000	0.002	$K_{a_1}o_2$
BBL02	2	1	1	2	115	2	232	0.007	0.009	$K_{a_1}o_1$
BBL03	2	1	1	∞	605160	*all*	1210320	36.1	78.2	$K_{a_2}o_3$
BBL04	2	2	1	2	115	2	232	0.015	0.017	$K_{a_1}K_{a_2}o_1$
BBL05	2	1	1	0	1	0	2	0.000	0.002	$DK_{a_1,a_2}\{o_1, o_2, o_3\}$
BBL06	2	1	1	0	1	0	2	0.000	0.002	$EK_{a_1,a_2}o_2$
BBL07	2	1	1	2	115	2	232	0.019	0.021	$EK_{a_1,a_2}\{o_1, o_2\}$
BBL08	2	1	1	0	1	0	2	0.000	0.002	$CK_{a_1,a_2}o_2$
BBL09	2	1	1	2	115	2	232	0.048	0.050	$CK_{a_1,a_2}\{o_1, o_2\}$
BBL10	2	2	1	2	115	2	232	0.016	0.018	$K_{a_1}DK_{a,b}\{o_1, o_2, o_3\}$
BBL11	2	2	2	8	59260	7509	187332	7.650	8.382	$K_{a_1}o_1 \wedge \neg K_{a_2}K_{a_1}o_1$
BBL12	2	3	2	9	15842	592	47626	2.380	2.472	$S_{a_1}o_1 \wedge \neg K_{a_1}S_{a_2}S_{a_1}o_1$

Table 2. Experiments Results for the Big Brother Logic domain

5.3. Social-media Network

The *Social-media Network* (SN) domain is an abstract network which agents can befriend each other to read their updates, etc., based on typical social media models. We extend two-way one-time communication channels from a classical gossip problem [6] into two-way, all-time communication channels, and add the concepts of secret messages. By decomposing secrets into messages and posting through an agent's friendship network, we model how secrets can be shared between a group of individuals not directly connected without anyone on the network knowing the secret, and some secrets can be shared within a group excepting for some individuals. The former would be ,ike spies sharing information with each other through the resistance's personal page, and the latter would be a group arranging a surprise party for a mutual friend.
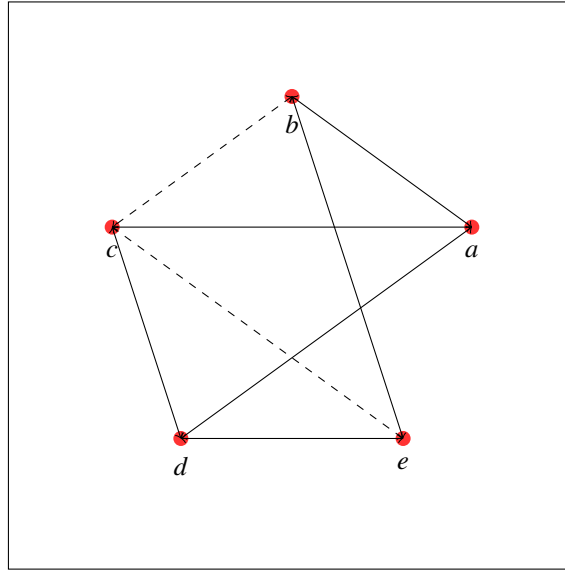
Figure 3. Example for Social-media Network

5.3.1. Examples

Let a, b, c, d, e be five agents in the SN, with friendship links shown in Figure 3. Their friend relations are represented by full lines between each agent.

Let g be a friend for all agents and g wants to share a secret. We assume the social network is in g's perspective directly, and the network is fixed for simplicity. Let all the epistemic queries that we concern of be a set Q, and p_1, p_2, p_3 as three parts of the secret P. Any problem by this set up can be represented by a tuple $(A, V, D, O, I, G, \mathbb{F})$, where:

- $A = \{a, b, c, d, e\}$

- $V = \{(friended\ i\ j), (post\ p)\ (q) \mid i, j \in A,\ p \in P,\ q \in Q\}$

- $D : dom(friended\ i\ j) = dom(q) = \{0, 1\}, dom(post\ p) = A$, where $i, j \in A,\ p \in P,\ q \in Q$

- $O : \mathbf{post}(i, p)$, where $i \in A,\ p \in P$

- $I = \{\ (friended\ a\ b) = 1,\ (friended\ a\ c) = 1,\ (friended\ a\ d) = 1,\ (friended\ b\ e) = 1,\ (friended\ c\ d) = 1,$ $(friended\ d\ e) = 1\ \}$

- G: see below

- $\mathbb{F} : (\mathbf{@check}: q) \mapsto \{true, false\}$

($friended\ i\ j$) represents whether i and j are friends with each other, which is a domain dependent relation, and q covers all the epistemic queries. ($\mathbf{post}\ i\ p$) specifies that the message p is posted on agent i's page. The I covers the friendship relations in Figure 3, and no message has been posted yet. Similarly, the action $\mathbf{post}$ is the only source for epistemic relation changes. Therefore, we update all desired queries Q as condition effects.

Goals that we have tested are shown in Table 3. For some epistemic formulae between a and b, since they are friends, simply posting the message in any of their personal page is sufficient to establish common knowledge about the information in that post. But for the knowledge between a and e, for example, $EK_{a,e}p_1$, the message needs to be posted on the page of a mutual friend, such as agent b. In addition, since a and e are not friends, in each of their perspectives of the world, there is no information (variables) describing others. Therefore, both $EK_{a,e}EK_{a,e}p_1$ and $CK_{a,e}p_1$ are not possible without changing the network structure.

Below are some of the sample goals that we have tested, with the identifying name from Table 3 and the plan that achieves that goal:

- $K_a p_1$ (SN01), $K_a K_b p_1$ (SN02), $EK_{a,b} p_1$ (SN03) and $CK_{a,b} p_1$ (SN06): **post**(a,p_1)

- $CK_{a,e} p_1$ (SN07): ∞

- $K_a\{p_1, p_2, p_3\}$ (SN08), $EK_{a,b}\{p_1, p_2, p_3\}$ (SN04) and $DK_{a,b}\{p_1, p_2, p_3\}$ (SN05):
  post(a,p_1), **post**(a,p_2), **post**(a,p_3)

A plan of ∞ means that the problem is unsolvable.
The other types of goals are secretive:

- $K_a\{p_1, p_2, p_3\} \wedge \neg K_b\{p_1, p_2, p_3\}$ (SN09): **post**(a,p_1), **post**(a,p_2), **post**(c,p_3)

- $K_a\{p_1, p_2, p_3\} \wedge \neg K_b\{p_1, p_2, p_3\} \wedge \neg K_c\{p_1, p_2, p_3\}$ (SN10): **post**(a,p_1), **post**(b,p_2), **post**(c,p_3)

The aims are to share the whole secret with a while b must not know the whole secret, but can know some of it. Some parts of it, such as p_3, needs to be shared in the page that b does not have access to. Then, c must also not know the secret, the secret now needs to be posted in the way that b and c do not see some parts respectively, while a sees all the parts.

Finally, we look on those two desired scenarios in the introduction of SN:

- Sharing with a spy:
  $K_a\{p_1, p_2, p_3\} \wedge \neg K_b\{p_1, p_2, p_3\} \wedge \neg K_c\{p_1, p_2, p_3\} \wedge \neg K_d\{p_1, p_2, p_3\} \wedge \neg K_e\{p_1, p_2, p_3\}$ (SN11):
  post(a,p_1), **post**(b,p_2), **post**(c,p_3)

- Surprise party:
  $\neg K_a\{p_1, p_2, p_3\} \wedge K_b\{p_1, p_2, p_3\} \wedge K_c\{p_1, p_2, p_3\} \wedge K_d\{p_1, p_2, p_3\} \wedge K_e\{p_1, p_2, p_3\}$ (SN13): ∞

Sharing a secret to some specific individual without anyone else knowing the secret can be done with the current network. However, if we alter the problem a bit by adding a friend relation between b and c (SN12), and apply the same goal conditions as SN11, no plan would be found by the planner, because c sees everything a can see, and there is no way to share some information to a without c seeing it.

For sharing a secret surprise party for agent a among all the agents without a knowing it, the messages needs to shared in such a way that a is not able to get a complete picture of the secrets. In the current set up of the problem (SN13), since a sees everything seen by c, there is no way to held a surprise party without a knowing it. However, by adding a friend relation between e and c (SN14), the planner returns with the plan: **post**(e,p_1), **post**(e,p_2), **post**(e,p_3).

Table 3 shows the results for our problems in the social-media network domain. The results show that is able to reason about nested knowledge and also group knowledge to achieve an intricate goal, which means our model can handle variety of knowledge relations at same time within reasonable time complexity.

5.4. Discussion

Overall, computationally, our solution outperforms *PDKB*, which is state-of-the-art for epistemic planning problems. The number of agents and depth of epistemic relations do not not increase the computation time as rapidly as the *PDKB* planner. In the terms of expressiveness, our solution demonstrates its capability to handle variety of complex epistemic relations, such as, nested knowledge, distributed knowledge and common knowledge.

The results show that the computational time depends heavily on how many times have the external functions are called, which is actually determined by the number of generated nodes and expanded nodes. Moreover, the number of nodes involved in the search is impacted by some factors, such as, the length of the plan, the algorithm that the planner uses, and also the scale of the problem itself.

The results also show that the external solver takes up a large part of the execution time. This is a prototype implementation and this represents an opportunity for performance optimisation of our code base.

Problem	Parameters				Performance			TIME(s)		Goal
	$\lvert a \rvert$	d	$\lvert g \rvert$	$\lvert p \rvert$	$\lvert gen \rvert$	$\lvert exp \rvert$	$\lvert calls \rvert$	calls	Total	
SN01	5	1	1	1	16	2	42	0.002	0.004	$K_i p_1$
SN02	5	2	1	1	16	2	42	0.003	0.005	$K_i K_j p_1$
SN03	5	1	1	1	16	2	42	0.003	0.004	$EK_{i,j} p_1$
SN04	5	1	1	3	216	92	3286	0.484	0.489	$EK_{i,j}\{p_1, p_2, p_3\}$
SN05	5	1	1	3	216	92	3286	0.566	0.571	$DK_{i,j}\{p_1, p_2, p_3\}$
SN06	5	1	1	1	16	2	42	0.006	0.007	$CK_{i,j} p_1$
SN07	5	1	1	∞	216	all	7776	0.825	0.838	$CK_{i,k} p_1$
SN08	5	1	1	3	216	92	3286	0.216	0.221	$K_i\{p_1, p_2, p_3\}$
SN09	5	1	2	3	306	107	7652	0.759	0.767	$K_i\{p_1, p_2, p_3\} \wedge \neg K_j\{p_1, p_2, p_3\}$
SN10	5	1	3	3	614	189	20334	2.049	2.069	$K_i\{p_1, p_2, p_3\} \wedge \neg K_{j \wedge k}\{p_1, p_2, p_3\}$
SN11	5	1	5	3	901	265	47570	4.841	4.886	$K_i\{p_1, p_2, p_3\} \wedge \neg K_{other}\{p_1, p_2, p_3\}$
SN12[1]	5	1	5	∞	2808	all	505400	57.5	58.0	$K_i\{p_1, p_2, p_3\} \wedge \neg K_{other}\{p_1, p_2, p_3\}$
SN13	5	1	2	∞	432	all	31104	5.589	5.629	$\neg K_i\{p_1, p_2, p_3\} \wedge K_{other}\{p_1, p_2, p_3\}$
SN14[2]	5	1	2	3	418	278	19964	4.049	4.073	$\neg K_i\{p_1, p_2, p_3\} \wedge K_{other}\{p_1, p_2, p_3\}$

Table 3. Experiments Results for the Social-media Network domain

6. Conclusions

In this work, we introduced a new epistemic planning model called the agent perspective model, driven from the intuition: "What you know is what you see". This perspective based model allows us to evaluate epistemic formula, even nested, distributed or common epistemic relations, based on the simple concept of defining an agent's local state. Then, by separating the planning task from epistemic reasoning with functional STRIPS, we proposed an expressive and flexible solution for most of the epistemic planning problems without an expensive pre-compilation step. We implemented our model well-known epistemic planning benchmarks and two new scenarios based on different perspective functions. The results not only show our model can solve the epistemic benchmarks efficiently, but also demonstrate a variety types of epistemic relation can be handled. Our work is the first to delegate epistemic reasoning to an external solver.

For future work, there are three ways to extend our model. First, extending the model to belief rather than knowledge. The success of our model is dependent on the property $f_i(s) \subseteq s$ for perspective functions, which implies beliefs cannot be false. Extending to belief would be an important step. Second, we can improve our model by allowing simplified disjunctive knowledge relations, such as that proposed by Miller et al. [32]. Finally, investigating event-based epistemic planning domains within our model would be another potential research direction. Our intuition is that those problems would challenge our model, however, it would be possible to including seeing events as well as seeing variables or agents.

References

[1] Laura Barbulescu, Jeremy Frank, Mausam, and Stephen F. Smith, editors. *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling, ICAPS 2017, Pittsburgh, Pennsylvania, USA, June 18-23, 2017*, 2017. AAAI Press. URL `http://www.aaai.org/Library/ICAPS/icaps17contents.php`.

[2] M. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *JAIR*, 47, 2013.

[3] Thomas Bolander. A gentle introduction to epistemic planning: The DEL approach. In Ghosh and Ramanujam [17], pages 1–22. doi: 10.4204/EPTCS.243.1. URL `https://doi.org/10.4204/EPTCS.243.1`.

[4] Thomas Bolander and Mikkel Birkegaard Andersen. Epistemic planning for single and multi-agent systems. *Journal of Applied Non-Classical Logics*, 21(1):9–34, 2011. doi: 10.3166/jancl.21.9-34. URL `https://doi.org/10.3166/jancl.21.9-34`.

[5] Thomas Bolander, Martin Holm Jensen, and François Schwarzentruber. Complexity results in epistemic planning. In Yang and Wooldridge [41], pages 2791–2797. ISBN 978-1-57735-738-4. URL `http://ijcai.org/Abstract/15/395`.

[6] Martin C. Cooper, Andreas Herzig, Faustine Maffre, Frédéric Maris, and Pierre Régnier. A simple account of multi-agent epistemic planning. In Gal A. Kaminka, Maria Fox, Paolo Bouquet, Eyke Hüllermeier, Virginia Dignum, Frank Dignum, and Frank van Harmelen, editors, *ECAI 2016 - 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands - Including Prestigious Applications of Artificial Intelligence (PAIS 2016)*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, pages 193–201. IOS Press, 2016. ISBN 978-1-61499-671-2. doi: 10.3233/978-1-61499-672-9-193. URL `https://doi.org/10.3233/978-1-61499-672-9-193`.

[7] C. Dornhege, P. Eyerich, T. Keller, S. Trüg, M. Brenner, and B. Nebel. Semantic attachments for domain-independent planning systems. In *Proc. ICAPS*, 2009.

[8] Thorsten Engesser, Thomas Bolander, Robert Mattmüller, and Bernhard Nebel. Cooperative epistemic multi-agent planning for implicit coordination. In Ghosh and Ramanujam [17], pages 75–90. doi: 10.4204/EPTCS.243.6. URL `https://doi.org/10.4204/EPTCS.243.6`.

[9] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge*. MIT Press, Cambridge, MA, USA, 2003. ISBN 0262562006.

[10] Jie Fan, Yanjing Wang, and Hans van Ditmarsch. Contingency and knowing whether. *Rew. Symb. Logic*, 8(1):75–107, 2015. doi: 10.1017/S1755020314000343. URL `https://doi.org/10.1017/S1755020314000343`.

[11] R. Fikes and N. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 1: 27–120, 1971.

[12] G. Francès and H. Geffner. Modeling and computation in planning: Better heuristics for more expressive languages. In *Proc. ICAPS*, 2015.

[13] Guillem Francès, Miquel Ramírez, Nir Lipovetzky, and Hector Geffner. Purely declarative action descriptions are overrated: Classical planning with simulators. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 4294–4301, 2017. doi: 10.24963/ijcai.2017/600. URL `https://doi.org/10.24963/ijcai.2017/600`.

[14] Olivier Gasquet, Valentin Goranko, and François Schwarzentruber. Big brother logic: logical modeling and reasoning about agents equipped with surveillance cameras in the plane. In Ana L. C. Bazzan, Michael N. Huhns, Alessio Lomuscio, and Paul Scerri, editors, *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14, Paris, France, May 5-9, 2014*, pages 325–332. IFAAMAS/ACM, 2014. ISBN 978-1-4503-2738-1. URL `http://dl.acm.org/citation.cfm?id=2615786`.

[15] Héctor Geffner. *Functional Strips: A More Flexible Language for Planning and Problem Solving*, pages 187–209. Springer US, Boston, MA, 2000. ISBN 978-1-4615-1567-8. doi: 10.1007/978-1-4615-1567-8_9. URL `https://doi.org/10.1007/978-1-4615-1567-8_9`.

[16] Hector Geffner and Blai Bonet. *A Concise Introduction to Models and Methods for Automated Planning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2013. ISBN 9781608459698. doi: 10.2200/S00513ED1V01Y201306AIM022. URL `https://doi.org/10.2200/S00513ED1V01Y201306AIM022`.

[17] Sujata Ghosh and R. Ramanujam, editors. *Proceedings of the Ninth Workshop on Methods for Modalities, M4M@ICLA 2017, Indian Institute of Technology, Kanpur, India, 8th to 10th January 2017*, volume 243 of *EPTCS*, 2017. doi: 10.4204/EPTCS.243. URL `https://doi.org/10.4204/EPTCS.243`.

[18] James Hales, Tim French, and Rowan Davies. Refinement quantified logics of knowledge and belief for multiple agents. *Advances in Modal Logic*, 9:317–338, 2012.

[19] Malte Helmert. The fast downward planning system. *J. Artif. Intell. Res.*, 26:191–246, 2006. doi: 10.1613/jair.1705. URL `https://doi.org/10.1613/jair.1705`.

[20] Malte Helmert and Carmel Domshlak. Landmarks, critical paths and abstractions: What's the difference anyway? In Alfonso Gerevini, Adele E. Howe, Amedeo Cesta, and Ioannis Refanidis, editors, *Proceedings of the 19th International Conference on Automated Planning and Scheduling, ICAPS 2009, Thessaloniki, Greece, September 19-23, 2009*. AAAI, 2009. ISBN 978-1-57735-406-2. URL `http://aaai.org/ocs/index.php/ICAPS/ICAPS09/paper/view/735`.

[21] Andreas Herzig and Faustine Maffre. How to share knowledge by gossiping. In Michael Rovatsos, George A. Vouros, and Vicente Julián, editors, *Multi-Agent Systems and Agreement Technologies - 13th European Conference, EUMAS 2015, and Third International Conference, AT 2015, Athens, Greece, December 17-18, 2015, Revised Selected Papers*, volume 9571 of *Lecture Notes in Computer Science*, pages 249–263. Springer, 2015. ISBN 978-3-319-33508-7. doi: 10.1007/978-3-319-33509-4_20. URL `https://doi.org/10.1007/978-3-319-33509-4_20`.

[22] Jaakko Hintikka. *Knowledge and Belief*. Ithaca: Cornell University Press, 1962.

[23] Xiao Huang, Biqing Fang, Hai Wan, and Yongmei Liu. A general multi-agent epistemic planner based on higher-order belief change. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 1093–1101, 2017. doi: 10.24963/ijcai.2017/152. URL `https://doi.org/10.24963/ijcai.2017/152`.

[24] Filippos Kominis and Hector Geffner. Beliefs in multiagent planning: From one agent to many. In Ronen I. Brafman, Carmel Domshlak, Patrik Haslum, and Shlomo Zilberstein, editors, *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling, ICAPS 2015, Jerusalem, Israel, June 7-11, 2015.*, pages 147–155. AAAI Press, 2015. ISBN 978-1-57735-731-5. URL `http:`

//www.aaai.org/ocs/index.php/ICAPS/ICAPS15/paper/view/10617.

[25] Filippos Kominis and Hector Geffner. Multiagent online planning with nested beliefs and dialogue. In Barbulescu et al. [1], pages 186–194. URL https://aaai.org/ocs/index.php/ICAPS/ICAPS17/paper/view/15748.

[26] Tiep Le, Francesco Fabiano, Tran Cao Son, and Enrico Pontelli. EFP and PG-EFP: epistemic forward search planners in multi-agent domains. In Mathijs de Weerdt, Sven Koenig, Gabriele Röger, and Matthijs T. J. Spaan, editors, *Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling, ICAPS 2018, Delft, The Netherlands, June 24-29, 2018.*, pages 161–170. AAAI Press, 2018. ISBN 978-1-57735-797-1. URL https://aaai.org/ocs/index.php/ICAPS/ICAPS18/paper/view/17733.

[27] Nir Lipovetzky and Hector Geffner. Width and serialization of classical planning problems. In Luc De Raedt, Christian Bessière, Didier Dubois, Patrick Doherty, Paolo Frasconi, Fredrik Heintz, and Peter J. F. Lucas, editors, *ECAI 2012 - 20th European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track, Montpellier, France, August 27-31 , 2012*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 540–545. IOS Press, 2012. ISBN 978-1-61499-097-0. doi: 10.3233/978-1-61499-098-7-540. URL https://doi.org/10.3233/978-1-61499-098-7-540.

[28] Nir Lipovetzky and Hector Geffner. Width-based algorithms for classical planning: New results. In Torsten Schaub, Gerhard Friedrich, and Barry O'Sullivan, editors, *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 1059–1060. IOS Press, 2014. ISBN 978-1-61499-418-3. doi: 10.3233/978-1-61499-419-0-1059. URL https://doi.org/10.3233/978-1-61499-419-0-1059.

[29] Nir Lipovetzky and Hector Geffner. Best-first width search: Exploration and exploitation in classical planning. In Satinder P. Singh and Shaul Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3590–3596. AAAI Press, 2017. URL http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14862.

[30] Nir Lipovetzky and Hector Geffner. A polynomial planning algorithm that beats LAMA and FF. In Barbulescu et al. [1], pages 195–199. URL https://aaai.org/ocs/index.php/ICAPS/ICAPS17/paper/view/15740.

[31] D. McDermott. The 1998 AI Planning Systems Competition. *Artificial Intelligence Magazine*, 21(2):35–56, 2000.

[32] Tim Miller, Paolo Felli, Christian J. Muise, Adrian R. Pearce, and Liz Sonenberg. 'knowing whether' in proper epistemic knowledge bases. In Dale Schuurmans and Michael P. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 1044–1050. AAAI Press, 2016. ISBN 978-1-57735-760-5. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12291.

[33] Christian J. Muise, Vaishak Belle, Paolo Felli, Sheila A. McIlraith, Tim Miller, Adrian R. Pearce, and Liz Sonenberg. Planning over multi-agent epistemic states: A classical planning approach. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 3327–3334, 2015. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9974.

[34] Christian J. Muise, Tim Miller, Paolo Felli, Adrian R. Pearce, and Liz Sonenberg. Efficient reasoning with consistent proper epistemic knowledge bases. In Gerhard Weiss, Pinar Yolum, Rafael H. Bordini, and Edith Elkind, editors, *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015*, pages 1461–1469. ACM, 2015. ISBN 978-1-4503-3413-6. URL http://dl.acm.org/citation.cfm?id=2773339.

[35] Silvia Richter and Matthias Westphal. The LAMA planner: Guiding cost-based anytime planning with landmarks. *J. Artif. Intell. Res.*, 39:127–177, 2010. doi: 10.1613/jair.2972. URL https://doi.org/10.1613/jair.2972.

[36] J. Rintanen. Planning as satisfiability: Heuristics. *Artificial Intelligence*, 193:45–86, 2012.

[37] HA Simon and Allen Newell. Gps, a program that simulates human thought. *Computers and Thought*, pages 279–293, 1963.

[38] Hans Van Ditmarsch, Wiebe van Der Hoek, and Barteld Kooi. *Dynamic epistemic logic*, volume 337. Springer Science & Business Media, 2007.

[39] Hai Wan, Rui Yang, Liangda Fang, Yongmei Liu, and Huada Xu. A complete epistemic planner without the epistemic closed world assumption. In Yang and Wooldridge [41], pages 3257–3263. ISBN 978-1-57735-738-4. URL http://ijcai.org/Abstract/15/459.

[40] Zhongbin Wu. An improved multi-agent epistemic planner via higher-order belief change based on heuristic search. In Weiru Liu, Fausto Giunchiglia, and Bo Yang, editors, *Knowledge Science, Engineering and Management - 11th International Conference, KSEM 2018, Changchun, China, August 17-19, 2018, Proceedings, Part II*, volume 11062 of *Lecture Notes in Computer Science*, pages 102–116. Springer, 2018. ISBN 978-3-319-99246-4. doi: 10.1007/978-3-319-99247-1_10. URL https://doi.org/10.1007/978-3-319-99247-1_10.

[41] Qiang Yang and Michael Wooldridge, editors. *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, 2015. AAAI Press. ISBN 978-1-57735-738-4. URL http://ijcai.org/proceedings/2015.