Heuristic Strategies for Accelerating Multi-Agent Epistemic Planning

Biqing Fang , Fangzhen Lin
The Hong Kong University of Science and Technology, Hong Kong, China
biqing.fang@connect.ust.hk, flin@cs.ust.hk

Abstract

Multi-agent epistemic planning (MEP) is about achieving an epistemic goal in a multi-agent environment using agents' actions that have epistemic preconditions and effects. Recently, MEP has received interest from both the dynamic logic and planning communities, leading to the development of several innovative planners. One such state of the art planner is MEPK. In this paper, we propose two novel strategies to enhance the search methods within MEPK. Our first strategy, the enhancement strategy, dynamically updates the heuristic based on the search path to the first goal-reachable node, potentially reducing the number of nodes that need to be explored to find a solution. Our second, the belief lock strategy, prevents the planner from continuing to search a particular state that cannot progress to a goal state due to the possession by an agent of a certain belief. Our experiments on existing benchmarks show that the new strategies can indeed accelerate the problem solving. We also construct new harder instances and demonstrate that our strategies significantly improve the performance on these hard benchmarks. Overall, we consider our new planner a significant improvement over the existing one in terms of computational efficiency.

1 Introduction

Multi-agent epistemic planning (MEP) is *automated planning* that involves epistemic reasoning for multiple agents (Baral et al. 2017; Belle et al. 2023). The actions of the agents in this domain usually include epistemic preconditions and effects, the goal is also epistemic, and the reasoning algorithms need to deal with nested beliefs of agents. For example, a goal could be that agent a and c want to know a secret p, and the current state is the situation in which "agent b does not know that p, or even that b does not know that they (a and c) know p". MEP has promising potential in various domains involving autonomous agents, such as multi-agent coordination and cooperation (Thielscher 2017) and real-world planning with social intelligence (Dissing and Bolander 2020; Bolander, Dissing, and Herrmann 2021).

In recent years, multi-agent epistemic planning has received attention from both dynamic logic and automated planning communities. There are many theoretic studies on MEP. Bolander and Andersen (2011) and Löwe, Pacuit, and Witzel (2011) initially conceptualized MEP using dynamic epistemic logic (DEL) (van Ditmarsch, van der Hoek, and Kooi 2007). The DEL framework models states as Kripke models, actions as action models (also called event models). These action models describe agents' capacities to distinguish different events, and utilize product update operation to update Kripke models. Aucher and Bolander showed that solving MEP problems is undecidable in general. This complexity is further highlighted by the discovery that MEP remains undecidable when restricted to purely epistemic actions (Aucher and Bolander 2013). Furthermore, Charrier, Maubert, and Schwarzentruber (2016) have demonstrated that this undecidability continues even with action preconditions constrained to a modal depth of two. Recent work by Cong, Pinchinat, and Schwarzentruber (2018) has also shown the undecidability of MEP in two-agent S5 models with fixed actions and goals. However, there are some decidable fragments of MEP (Yu, Wen, and Liu 2013; Cooper et al. 2016a), and Cooper et al. (2016b) examined the "gossip problem" within the context of MEP, revealing its polynomial-time solvability but also its escalation to NP-completeness when negative goals are introduced.

For the implementation of multi-agent epistemic planning. Kominis and Geffner (2015) and Muise et al. (2015; 2022) exploit classical planning to solve restricted versions of MEP problems. By translating restricted MEP problems to classical planning problems, both works generate linear plans by performing conformant planning. Similarly, the EL-O framework (Cooper et al. 2021) takes a syntactic restriction and compiles the problem of epistemic planning into classical ones. Engesser et al. (2017) studied decentralized planning with implicit coordination. Based on earlier works on the action language $m\mathcal{A}$ and finitary S5-theories, Le et al. (2018) presented the planner EFP, where the initial state is specified by a finitary S5-theory. Then Fabiano et al. (2020) introduced EFP 2.0 using alternative state representations that deviate from the commonly used Kripke structures, improving the performance of EFP. Hu, Miller, and Lipovetzky (2022) decompose epistemic planning by delegating reasoning about epistemic formulas to an external solver. Recently, Wan, Fang, and Liu (2021) proposed a local MEP planner in a syntactic approach called MEPK. In their framework, states are described by syntactically restricted formulas, called alternating cover disjunctive formulas (ACDFs). Because ACDFs have the full expressive power of epistemic logic, MEPK can describe a wide range of MEP problems and produce conditional plans (policies) in a *contingent planning* framework.

However, Wan, Fang, and Liu (2021) only evaluated the viability of MEPK in rather easy domains collected from the literature. In this paper, we revisit MEPK and propose two new strategies based on our observations of the previous results of MEPK. Firstly, we propose the enhancement strategy by utilizing the information on the path from the initial node to the first goal-reachable node (for contingent planning, an action tree becomes a feasible solution only if each branch of it achieves the goal). The basic idea is to update the explore order of the nodes by changing their attached enhancement value during the search. Then, we identify a scenario of "belief lock" during planning in some specific domains and propose a strategy that prevents the planner from exploring the nodes that are valueless under the scenario of "belief lock". By incorporating the two novel strategies, the performance of MEPK is improved. Then, we build hard instances based on the existing benchmarks to evaluate the performance of the new strategies on hard benchmarks. Finally, the experimental results affirm the effectiveness of the new strategies, marking a significant leap in computational efficiency for MEP.

2 Preliminaries

In this section, we review the background of multi-agent modal logic KD45$_n$ and the MEPK modeling framework.

2.1 Multi-agent Modal Logic KD45$_n$

Consider a finite set of agents $\mathcal{A}$ and a finite set of atoms $\mathcal{P}$. We use ϕ and ψ for formulas, Φ and Ψ for sets of formulas.

Definition 1. The language $\mathcal{L}_{KC}$ of multi-agent modal logic with common knowledge is generated by the BNF:

$$\phi ::= p \mid \neg\phi \mid (\phi \wedge \phi) \mid B_a\phi \mid C\phi, \text{ where}$$

$p \in \mathcal{P}$, $a \in \mathcal{A}$, $\phi \in \mathcal{L}_{KC}$. $\mathcal{L}_K$ is used for the language without the C operator, and $\mathcal{L}_0$ for the propositional language.

Intuitively, $B_a\phi$ means that agent a believes ϕ holds, and $C\phi$ means ϕ is *common knowledge* among all agents. The original MEPK does not support arbitrary common knowledge, so we also restrict our attention to the case of propositional common knowledge *i.e.*, $C\phi$ where $\phi \in \mathcal{L}_0$, and the propositional formula ϕ is called a *constraint*.

We let $\top$ and $\bot$ represent *true* and *false* respectively. We let $\bigvee \Phi$ (resp. $\bigwedge \Phi$) denote the disjunction (resp. conjunction) of members of Φ. The *modal depth* of a formula ϕ in $\mathcal{L}_K$ is the depth of nesting of modal operators in ϕ.

Definition 2. A *frame* is a pair (W, R), where W is a nonempty set of possible worlds; for each agent $a \in \mathcal{A}$, $R : \mathcal{A} \to 2^{W\times W}$ assigns to each $a \in \mathcal{A}$ an accessibility relation R_a.

We say R_a is *serial* if for any $w \in W$, there is $w' \in W$ s.t. wR_aw'; we say R_a is *transitive* if whenever wR_aw_1 and $w_1R_aw_2$, we get wR_aw_2; we say R_a is *Euclidean* if whenever wR_aw_1 and wR_aw_2, we get $w_1R_aw_2$. A KD45$_n$ frame is a frame whose accessibility relations are serial, transitive, and Euclidean.

Definition 3. A Kripke model is a triple $M = (W, R, V)$, where (W, R) is a frame, and $V : W \to 2^{\mathcal{P}}$ is a valuation map. A pointed Kripke model is a pair $s = (M, w)$, where M is a Kripke model and w is a world of M, called the *actual world*.

Definition 4. Let $s = (M, w)$ be a pointed Kripke model where $M = (W, R, V)$. We interpret formulas in $\mathcal{L}_{KC}$ by induction:

- $M, w \models p$ iff $p \in V(w)$;
- $M, w \models \neg\phi$ iff $M, w \not\models \phi$;
- $M, w \models \phi \wedge \psi$ iff $M, w \models \phi$ and $M, w \models \psi$;
- $M, w \models B_a\phi$ iff for all v s.t. wR_av, $M, v \models \phi$;
- $M, w \models C\phi$ iff for all v s.t. $wR_{\mathcal{A}}v$, $M, v \models \phi$, where $R_{\mathcal{A}}$ is the transitive closure of the union of R_a for $a \in \mathcal{A}$.

A model of ϕ is a KD45$_n$ Kripke model (M, w) s.t. $M, w \models \phi$. We say ϕ is *satisfiable* if ϕ has a model. We say ϕ *entails* ψ, written $\phi \models \psi$, if any model of ϕ is also a model of ψ. We use $C^*\phi$ to denote $\phi \wedge C\phi$, and say that ϕ is satisfiable w.r.t. constraint $\gamma \in \mathcal{L}_0$ if $\phi \wedge C^*\gamma$ is satisfiable; we say that ϕ entails ψ w.r.t. constraint γ, written $\phi \models_\gamma \psi$, if $\phi \wedge C^*\gamma \models \psi \wedge C^*\gamma$.

2.2 Modeling Framework of MEPK

We briefly introduce the MEPK modeling framework. Please refer to (Wan, Fang, and Liu 2021) for more details.

Definition 5. A multi-agent epistemic planning problem $\mathcal{Q}$ is a tuple $\langle \mathcal{A}, \mathcal{P}, \mathcal{D}, \mathcal{S}, \mathcal{I}, \mathcal{G}, \gamma \rangle$, where $\mathcal{A}$ is a set of agents; $\mathcal{P}$ is a set of atoms; $\mathcal{D}$ is a set of deterministic actions; $\mathcal{S}$ is a set of sensing actions; $\mathcal{I} \in \mathcal{L}_K$ is the initial knowledge base; $\mathcal{G} \in \mathcal{L}_K$ is the goal; and $\gamma \in \mathcal{L}_0$ is the constraint.

The propositional constraint is similar to the domain closure axiom in classical planning domains (Lin 2004). For example, in a domain with a single box and two rooms, the constraint $\gamma = in(box, room_1) \wedge (\neg in(box, room_2)) \vee (\neg in(box, room_1)) \wedge in(box, room_2)$ indicates that the box must be located in exactly one room.

There are three types of actions: ontic, communication, and sensing actions. The first two types share the same representation, called deterministic actions.

Definition 6. A deterministic action is a pair $\langle pre, eff \rangle$, where $pre \in \mathcal{L}_K$ is the precondition; *eff* is a set of conditional effects, each of which is a pair $\langle c, e \rangle$, where c, $e \in \mathcal{L}_K$ indicate the condition and effect, respectively.

Definition 7. A sensing action is a triple $\langle pre, pos, neg \rangle$ of $\mathcal{L}_K$ formulas, where pre, pos, and neg indicate the precondition, the positive result, and the negative result, respectively.

Sensing actions are used to gather information from the environment. For example, agents can sense the status of the lighting in a room, which leads to two possible results: the light is on (pos) or off (neg).

An action a is *executable* w.r.t. a KB $\phi \in \mathcal{L}_K$ if $\phi \models_\gamma pre(a)$. This means that a is executable in each model of ϕ. Suppose a is executable w.r.t. ϕ. The progression of ϕ w.r.t. a is defined by resorting to a higher-order revision operator $\circ_\gamma$ and a higher-order update operator $\diamond_\gamma$, where γ is a constraint (Wan, Fang, and Liu 2021).

The basic idea of these two operators is to reduce the change of higher-order epistemic formulas to that of lower-order epistemic formulas, and as a basis, it resorts to a change of propositional formulas.

The following is the progression for the sensing action.

Definition 8. Let $\phi \in \mathcal{L}_K$, and a a sensing action. If $\phi \wedge pos(a)$ is propositionally unsatisfiable w.r.t. γ, then the progression of ϕ w.r.t. a with positive result is $\bot$. Otherwise, the result is $\phi \circ_\gamma pos(a)$. The progression of ϕ w.r.t. a with negative result $neg(a)$ is defined similarly.

The progression for deterministic action is more complex. In short, it is done by splitting KB ϕ based on the satisfiability of conditions c_i for each effect e_i, and considering all possible splittings to determine the overall progression.

We now introduce the concepts of *action tree*, which is essential for defining a solution.

Definition 9. The progression of ϕ w.r.t. a sequence of actions is inductively defined as follows: $prog(\phi, \epsilon) = \phi$, where ϵ represents the empty sequence; $prog(\phi, (a;\sigma)) = prog(prog(\phi, a), \sigma)$ if $\phi \models pre(a)$, and undefined otherwise.

Definition 10. Let $\mathcal{Q}$ be an MEP problem $\langle \mathcal{A}, \mathcal{P}, \mathcal{D}, \mathcal{S}, \mathcal{I}, \mathcal{G}, \gamma \rangle$. The set $\mathcal{T}$ of action trees is inductively defined:

1. ϵ is in $\mathcal{T}$, here ϵ represents the empty tree;
2. if $a_d \in \mathcal{D}$ and $T \in \mathcal{T}$, then $a_d; T$ is in $\mathcal{T}$;
3. if $a_s \in \mathcal{S}$, $T^+, T^- \in \mathcal{T}$, then $a_s; (T^+ \mid T^-)$ is in $\mathcal{T}$.

Definition 11. Let $\mathcal{Q}$ be an MEP problem $\langle \mathcal{A}, \mathcal{P}, \mathcal{D}, \mathcal{S}, \mathcal{I}, \mathcal{G}, \gamma \rangle$. Let T be an action tree. We say a branch σ of T achieves the goal if $prog(\mathcal{I}, \sigma)$ is defined, and $prog(\mathcal{I}, \sigma) \models_\gamma \mathcal{G}$; and if $prog(\mathcal{I}, \sigma)$ is not $\bot$, we say σ properly achieves the goal. We say T is a solution of $\mathcal{Q}$ if each branch of T achieves the goal, and at least one branch properly achieves the goal.

We explain the concept of solution using an example. Selective-communication (Kominis and Geffner 2015) is a domain of MEP: There are several rooms in a corridor. The agents can move from one room to a neighboring room. When agent i gives some information, all other agents in the same room or in a neighboring room can hear what was said. Initially, each agent is in one of the rooms. The goal is for some agents to get to know some information while some other agents do not.

Example 1. *Consider an instance with four rooms and three agents a, b, and c. Only agent a has the ability to move, sense and tell the information p. Initially, the agents a, b and c are in room 1, 2, and 3, respectively, while the information p is located in room 2. The goal is to ensure that agents a and c know p while b not. Figure 1 is an example of feasible solutions. The action tree is a solution if both the sequences of $(right; sense{+}; right; right; tell)$ and $(right; sense{-}; right; right; tell)$ achieve the goal. Node 1 is the initial node. The sensing action, sense, can yield two possible results: p and $\neg p$. In the planning process, nodes 6 and 10 are found to entail the goal.*

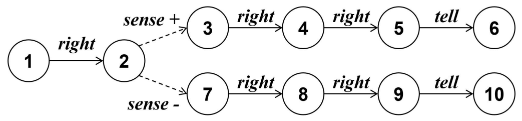

Figure 1: A solution to an instance of Selective-communication.

Algorithm 1: Planning

```
Input: An MEP problem $\mathcal{Q} = \langle \mathcal{A}, \mathcal{P}, \mathcal{D}, \mathcal{S}, \mathcal{I}, \mathcal{G}, \gamma \rangle$.
Output: A solution or null.
1  if $\mathcal{I} \models \mathcal{G}$ then
2      return an empty tree
3  else
4      Let $n_0 = \mathcal{I}$, $state(n_0) = unexplored$,
5      $connected(n_0) =$ true, $\mathcal{N} = \{n_0\}$, $\mathcal{T} = \emptyset$
6  while there are unexplored nodes do
7      Choose the next node $n$ with some method
8      explore($n$)
9      if initial node is goal-reachable then
10         return build_plan($\mathcal{T}$)
11     if initial state is dead then
12         return null
13 return null
```

The planner uses a knowledge base (KB) to model an epistemic state. When an action is performed, it *revises* or *updates* the current KB with the action's effects. MEPK adopts the PrAO (Pruning AND/OR search) algorithm for contingent planning (To, Son, and Pontelli 2011) as the planning algorithm, which allows the planner to significantly prune the search space. Algorithm 1 presents the framework. The algorithm takes an MEP problem as input and aims to find a solution or return "null" if no solution is found.

Each node n is a knowledge base. We use $state(n)$ to indicate the state of n, which is one of the following values: $goal$ meaning goal-reachable if there is a path (sequence of actions) leading to the goal; $dead$ meaning that the node is not goal-reachable and there is no outgoing edges from it; $unexplored$ if it is newly added to the search graph; $explored$ after it is chosen for expansion. During planning, nodes can be disconnected and reconnected to the search graph. We use $connected(n)$ to denote whether node n can be reached from the initial node via a path. Lines 1-5 return an empty tree if $\mathcal{I} \models \mathcal{G}$; otherwise, initialize the search graph, *i.e.*, set the initial node n_0 to the initial KB, and let $connected(n_0) = true$. We use $\mathcal{N}$ and $\mathcal{T}$ to denote the set of all nodes and the search graph, respectively. The algorithm then enters a loop that continues as long as there are nodes unexplored.

Within the loop, it selects an unexplored node n using a specific method such as breath-first search (BFS) or heuristic search (line 7). The selected node n is then explored. If, after this exploration, the initial node becomes goal-reachable, the algorithm *build_plan* simply extracts the plan from the expanded action tree ($\mathcal{T}$) and returns this plan as the solution. Thanks to the pruning technique, the remaining search graph is also the solution tree when a solution is found. If the initial state is determined to be $dead$, indicating that no action tree can route from the initial node while ensuring all its terminals entail the goal, the algorithm returns "null". Please refer to (To, Son, and Pontelli 2011) for more details.

For contingent planning, the function $explore(n)$ generates the children of node n by applying the executable deterministic and sensing action separately.

The efficiency of this algorithm is heavily influenced by the method used for node selection. For example, BFS chooses nodes in the order they are generated, and the heuristic search utilizes an auxiliary value to sort nodes.

3 Our Strategies

In this section, we first show our observations on the existing results of MEPK and then present the new strategies.

3.1 Enhancement Strategy

We rerun the experiments of MEPK[1] on all benchmarks and collected statistics on the first reachable node[2] and the total number of explored nodes.

Instance	$\|\mathcal{A}\|$	MEPK (BFS)	MEPK (Heu)
*CC(2,3)	2	26/29	6/10
*CC(2,3)	3	46/277	8/23
*CC(2,4)	2	8/626	10/279
*CC(3,3)	3	32/254	6/19
FT(1,2)	1	1/1	1/1
FT(2,3)	1	1/40	1/10
FT(2,3)	2	1/297	1/30

Table 1: Statistics of searching results on *CC and FT.

Table 1 presents the statistics of the search results in Collaboration-and-communication (CC) and Finding-the-truth (FT). The details of CC and FT will be introduced in Section 4.2. In Table 1, the first two columns indicate the name of instance and number of agents. In the MEPK columns (one for BFS and one for heuristic search), A/B stands for A nodes explored to find the first goal-reachable node, and B nodes explored in total.

In certain cases, such as *CC (2,4) and FT (2,3), it takes only 8 and 1 nodes to identify the first goal-reachable node by using BFS, which means a path from the initial node to a goal-reachable node is built. However, the solution tree has not yet been completed. There is a need to explore more than ten times the number of nodes (626 and 40 nodes) to find a final solution, significantly diminishing the efficiency of the search process.

To reduce the cost of exploring redundant nodes, we propose the enhancement strategy utilizing the information on the path from the initial node to the first goal-reachable node.

[1]The source code of MEPK can be accessed at https://github.com/sysulic/MEPK.

[2]In contingent planning, identifying the first goal-reachable node is not enough to find a solution; see the example of Figure 1.

Algorithm 2: *explore*(n)

```
   **Input:** A node $n$.
 1 **foreach** deterministic action $d \in \mathcal{D}$ **do**
 2     **if** $n \models pre(d)$ **then**
 3         Compute the successor $n'$ of $n$ after doing $d$
 4         **if** $n' \neq n$ **and** ($n' \notin \mathcal{N}$ **or** $n'$ *is not dead*) **then**
 5             $expand(n, d, n')$
 6             **if** $state(n') = goal$ **then**
 7                 $state(n) \leftarrow goal$
 8                 $MAX \leftarrow MAX + |\mathcal{N}|$
 9                 $enhancement\_propagation(n, a, MAX)$
10                 $goal\_propagation(n, d)$
11                 **return**
12 **foreach** sensing action $e \in \mathcal{S}$ **do**
13     **if** $n \models pre(e)$ **then**
14         Compute the successor $n_1$ of $n$ after doing $e^+$
15         Compute the successor $n_2$ of $n$ after doing $e^-$
16         **if** ($n_1 \notin \mathcal{N}$ **or** $n_1$ is not dead) **and** ($n_2 \notin \mathcal{N}$ **or**
             $n_2$ is not dead) **then**
17             $expand(n, e^+, n_1)$
18             $expand(n, e^-, n_2)$
19             **if** $state(n_1) = state(n_2) = goal$ **then**
20                 $state(n) \leftarrow goal$
21                 $MAX \leftarrow MAX + |\mathcal{N}|$
22                 $enhancement\_propagation(n, a, MAX)$
23                 $goal\_propagation(n, e)$
24                 **return**
```

In contingent planning, the solution branches on sensing actions (positive successor and negative successor). The intuition behind the enhancement strategy is that if the positive successor leads to the goal, the negative successor usually also leads to the goal. If it fails in some specific scenarios, the strategy will update the enhancement value accordingly to prevent the search from worsening. The basic idea is to use an enhancement value to sort the order of node exploration and update the enhancement value attached to each node during the search.

The main components of the enhancement strategy consist of a revised algorithm *explore*, and two novel algorithms *enhancement_propagation* and *update_children*.

The Algorithm 2 *explore* receives an unexplored node as input, then iterates over all deterministic actions ($\mathcal{D}$) available in the domain. For each action d, it checks if the precondition is entailed by the current KB (line 2). If the action is applicable and leads to a new, non-dead successor state n' (line 4), the algorithm calls the *expand* function to incorporate n' into the search space. If the newly generated state is a goal state, the algorithm updates the state of the current node to $goal$. The maximum enhancement value is then increased by the number of existing nodes ($\mathcal{N}$), ensuring that the subsequent boosted node is given priority in exploration (line 8). In addition, it performs *enhancement_propagation* to update the search heuristic based on the path to the goal, followed by a *goal_propagation* to back-propagate the goal information through the planning graph. The algorithm repeats a similar process for sensing actions ($\mathcal{S}$), generating two successors for each action based on its possible outcomes (lines 12-24).

The key revision of *explore* is employing a new *enhancement_propagation* mechanism, Algorithm 2 dynamically updates the search heuristic, potentially reducing the number of nodes that must be explored to find a solution. This method aims to improve the efficiency, especially in complex instances where the search space is vast.

Example 1. ***(continued)*** *Recall the nodes in Figure 1, and suppose the planner is about to explore node 5: $explore(5)$. The domain contains three deterministic actions: $\mathcal{D} = \{tell, left, right\}$, and one sensing action: $\mathcal{S} = \{sense\}$.*

The Algorithm 2 first examines the deterministic actions in $\mathcal{D}$ (line 1). It finds that KB at node 5 entails the precondition of the action $tell$ (line 2). Consequently, it progresses the KB with respect to $tell$, computing a new KB at node 6, which is found to entail the goal (line 6). Node 6 is then marked as goal-reachable, and this information will be back-propagated later (line 10). The maximum enhancement value is updated based on the number of nodes (line 8), and finally the priority of the dual node of node 3, i.e., node 7, for the action $sense$, is increased via enhancement propagation (line 9).

The process is similar for sensing actions (lines 12-24).

Intuitively, Algorithm 3 *enhancement_propagation* is responsible for propagating a new enhancement value through the planning graph to dynamically refine the search strategy. This algorithm is invoked when a goal state is reached, and it backtracks from the goal state to the initial state, updating the enhancement values of the nodes along this path.

The algorithm receives three inputs: the current node n, the action t that led to n, and a new enhancement value v. If n is the initial node, the propagation is complete and the algorithm returns immediately as there is no predecessor. If n is not the initial node, the algorithm identifies the predecessor of n, denoted as n', and the action t' that expanded to n'. Then it recursively calls itself the predecessor n', the action t', and a decremented enhancement value $v - 1$, effectively backtracking and updating the enhancement values along the path from the goal state to the initial state.

The Algorithm 4 *update_children* recursively updates the enhancement value of a node and all its descendants in the planning graph.

Together, these algorithms ensure that the search heuristic is informed by the latest path information, guiding the search process toward the goal states more efficiently. The enhancement strategy is effective because, typically, if one branch of an action tree achieves the goal, other branches corresponding to the same sensing action are also likely to lead to the goal.

The idea of the enhancement strategy is similar to the AO* algorithm. A key difference is that the enhancement strategy updates the enhancement value only after the first goal-reachable node is identified bottom-up. This is because doing high-order belief change is time-consuming. In contrast, the AO* algorithm adjusts cost estimates continuously throughout the search process.

Algorithm 3: *enhancement_propagation*(n, t, v)

```
Input: A node $n$, the action $t$ expanding to $n$, and the
       new enhancement value $v$.
1 if $n$ is the initial node then
2     return
3 Let $n'$ be the predecessor node of $n$
4 if $n'$ is not the initial node then
5     Let $a'$ be the action that expands to $n'$
6     enhancement_propagation$(n', t', v - 1)$
7 if $a \in \mathcal{S}$ then
8     update_children$(n, v)$
```

Algorithm 4: *update_children*(n, v)

```
Input: A node $n$, and enhancement value $v$.
1 $n.enhancement\_value \leftarrow v$
2 Let $N$ be the set of child nodes of $n$
3 foreach node $n' \in N$ do
4     update_children$(n', v)$
```

3.2 Belief Lock Strategy

Inspired by the idea of "*the curse of knowledge*" (Heath and Heath 2006), saying that once people acquire a certain knowledge, they cannot return to the state of not knowing it, we propose the belief lock strategy.

We explain the intuition using an example in the domain of Selective-communication (SC). Recall that the goal of the instances from the domain SC is that some agents get to know some information, while some other agents do not.

Example 2. *Given an instance of domain SC. Let $B_a p \wedge B_b p \wedge B_c p$ be the KB of state i, $\neg B_a p \wedge \neg B_b p \wedge \neg B_c p$ the KB of state j, and $B_a p \wedge \neg B_b p \wedge B_c p$ the goal of the instance. The goal says that agent a and c get to know the information p while agent b does not.*

In planning, according to the heuristic strategy based on the count of shared terms between the KB and the goal, state i will be explored prior to the state j because there are two agents' beliefs that are consistent with the goal in state i, i.e., $B_a p$ and $B_c p$, while there is only one term $\neg B_b p$ in state j.

However, within the SC domain, no actions can induce agent b to forget the information p or believe $\neg p$. Consequently, this implies that state i cannot progress to a goal-reachable state in this domain.

We use the term "belief lock" to describe a scenario in which a particular state is unable to progress to the goal state due to an agent's possession of certain knowledge.

Next, we express the idea of "*the curse of knowledge*" in the language of MEP as the following proposition.

Proposition 1 (Belief Lock)**.** *Consider an MEP problem $\mathcal{Q} = \langle \mathcal{A}, \mathcal{P}, \mathcal{D}, \mathcal{S}, \mathcal{I}, \mathcal{G}, \gamma \rangle$. Let $a \in \mathcal{A}$, $p \in \mathcal{P}$, and σ a sequence of actions. Then $prog(prog(\mathcal{I}, \sigma), \tau) = \bot$ or*

$prog(prog(\mathcal{I},\sigma),\tau) \not\models_\gamma \mathcal{G}$ for any action sequence τ if the following hold:

1. *$prog(\mathcal{I},\sigma) \models_\gamma B_a p$*
2. *$\mathcal{G} \models_\gamma \neg B_a p$*
3. *$e_i \not\models_\gamma \neg B_a p$, where $\langle c_i, e_i\rangle \in eff(d)$, for any $d \in \mathcal{D}$*
4. *$pos(s) \not\models_\gamma \neg B_a p$ and $neg(s) \not\models_\gamma \neg B_a p$, for any $s \in \mathcal{S}$*

Intuitively, conditions 2-4 in Proposition 1 suggest that dedicated domains do not involve actions that alter the belief lock situation. If the search reaches a state where a specific agent a is aware of the secret, and conditions 2-4 hold for the domain, then progression for any subsequent action sequence leads to either $\perp$ or fails to entail the goal. A detailed proof is provided in the appendix. See Section A below.

The Proposition 1 limits the scope of where the strategy works. If the goal only involves the belief of some agents that change over time during planning, such as the beliefs of boxes' location $B_a(in(box_1, room_1))$ from domain Collaboration-and-communication, then the strategy does not work and may even mislead the search.

Therefore, the belief lock strategy is suitable for the domains where the goal involves maintaining certain information as a secret, undisclosed to some specific agents, and the domain itself cannot alter the belief lock situation, such as the domains Selective-communication, Grapevine, and Gossip.

To apply Proposition 1 to planning, we revise the algorithm *expand* as in Algorithm 5.

Before planning, we initialize a set Ψ of formulas derived from the goal $\mathcal{G}$. Each formula in the set represents the knowledge that the goal is to keep unknown to a particular agent. The algorithm *expand* receives a node n, the action t that led to a new node n', and an enhancement value v. Firstly, a new edge is added (line 1) and the enhancement value of the new node is initialized (line 2). Then the state of n' is updated (lines 3-6). When the planner expands to a new node, we check each formula for the node (line 7). If the node entails any of the formulas (line 8), indicating the occurrence of a "belief lock" scenario, we assign a minimum value as the enhancement value for the new node (line 9).

Example 2. ***(continued)*** *For the previous instance of domain SC, $\Psi = \{\neg B_b p\}$, and $n_j.enhancement\_value$ = MIN after the node j is created. That is, the search priority of the newly created node j is minimized (the node j can even be tagged as dead if the problem are checked to satisfy the conditions of Proposition 1). Thus, the planner will not waste time exploring valueless nodes.*

Due to the space limit, we only describe the new proposed strategies and the revised parts of the algorithms. Please refer to (To, Son, and Pontelli 2011) and the source code of MEPK for more details of sub-algorithms.

4 Experimentation

In this section, we conduct two experiments. Firstly, we evaluate the feasibility of two new strategies on existing benchmarks used in the previous literature. Secondly, We build hard instances based on the existing benchmarks by increasing the number of agents, objects such as rooms, blocks, and secrets, and setting complex goals. Then, we evaluate the effectiveness of the two new strategies in the hard instances.

All the experiments are run on a Linux machine with a 2.90GHz CPU and 16GB RAM. The time limit for each run is set to 3,600 seconds.

Algorithm 5: *$expand(n,t,n')$*

```
Input: A node $n$, the action $t$, the successor node $n'$, and
       the enhancement value $v$.
1  $\mathcal{T} \leftarrow \mathcal{T} \cup \{(n,t,n')\}$
2  $n'.enhancement\_value \leftarrow v$
3  if $n' \not\models \mathcal{G}$ then
4      $state(n') = unexplored$
5  else
6      $state(n') = goal$
7  foreach secret knowledge $\psi \in \Psi$ do
8      if $n' \models \psi$ then
9          $n'.enhancement\_value \leftarrow MIN$
10         break
11 $\mathcal{N} \leftarrow \mathcal{N} \cup \{n'\}$
```

4.1 Experiments on Easy Benchmarks

Table 2 presents the results on easy benchmarks.

Evaluation of enhancement strategy To assess the effectiveness of the enhancement strategy, we compare the column BFS (resp. Heu) and BFS$^+$ (resp. Heu$^+$). The improved results are underlined.

Recall that the enhancement strategy is trying to speed up the search process after identifying the first goal-reachable node. For the BFS method, there are 24 out of 41 instances in which the first goal-reachable node is found before exploring the last node to get the solution ($B - A > 0$). The performance of 18 out of 24 instances is improved with the enhancement strategy. For the heuristic search method, there are 23 instances in which the first goal-reachable node is found before exploring the last node to get the solution. Performance has improved in 20 of the 24 instances, and, notably, an instance previously unsolvable has now been successfully resolved.

Evaluation of belief lock strategy To evaluate the belief lock strategy, we consider the domains that involve the goal of keeping certain information secret, undisclosed to some specific agents. There are 15 instances from the three domains: SC, Grapevine, and Gossip, which are the last three domains listed in the table.

Table 3 presents a comparison within two sets of strategies: BFS, BFS$^+$, and BFS$^+$K on the one hand, and Heu, Heu$^+$, and Heu$^+$K on the other.

For each method (BFS and heuristic search), Table 3 counts the number of instances in which the corresponding method explores the smallest number of nodes to obtain the solution among the three variants.

Instance	$\lvert\mathcal{A}\rvert$	md	BFS	BFS$^+$	BFS$^+$K	Heu	Heu$^+$	Heu$^+$K
CC(2,4)	2	1	19.77 (130/130/263)	19.74 (130/130/263)	19.76 (130/130/263)	**1.63 (11/11/45)**	**1.64 (11/11/45)**	**1.65 (11/11/45)**
CC(3,4)	2	1	232.21 (130/130/263)	231.14 (130/130/263)	231.59 (130/130/263)	**21.80 (11/11/45)**	**22.00 (11/11/45)**	**22.03 (11/11/45)**
CC(4,4)	2	1	–	–	–	**465.52 (11/11/45)**	**465.34 (11/11/45)**	**464.95 (11/11/45)**
*CC(2,3)	2	1	0.38 (26/29/119)	0.39 (26/29/116)	0.39 (26/29/116)	0.10 (6/10/51)	**0.09 (6/8/46)**	**0.10 (6/8/46)**
*CC(2,3)	3	1	2.19 (46/277/1223)	0.31 (46/77/451)	0.31 (46/77/451)	0.04 (8/23/156)	**0.03 (8/13/106)**	**0.03 (8/13/106)**
*CC(2,4)	2	1	5.73 (8/626/1160)	0.27 (8/57/178)	0.27 (8/57/178)	1.94 (10/279/751)	**0.22 (10/45/172)**	**0.22 (10/45/172)**
*CC(3,3)	3	1	9.07 (32/254/1429)	1.57 (32/67/570)	1.56 (32/67/570)	**0.16 (6/19/185)**	0.33 (6/27/265)	0.33 (6/27/265)
†CC(2,3)	2	1	0.26 (60/63/342)	0.28 (60/65/371)	0.28 (60/65/371)	0.08 (8/31/188)	**0.02 (8/10/92)**	**0.02 (8/10/92)**
†CC(2,3)	3	1	0.19 (46/51/343)	0.32 (46/60/430)	0.32 (46/60/430)	0.05 (10/18/130)	**0.04 (10/14/127)**	**0.04 (10/14/127)**
†CC(2,4)	2	1	**0.04 (8/11/49)**	0.07 (8/15/78)	0.07 (8/15/78)	2.10 (11/330/850)	0.17 (11/29/147)	0.17 (11/29/147)
†CC(3,3)	3	1	0.54 (32/39/312)	2.42 (32/62/666)	2.42 (32/62/666)	**0.31 (7/28/238)**	1.03 (7/37/406)	1.03 (7/37/406)
FT(1,2)	1	1	**0.01 (1/1/4)**	**0.01 (1/1/4)**	**0.01 (1/1/4)**	**0.01 (1/1/4)**	**0.01 (1/1/4)**	**0.01 (1/1/4)**
FT(2,3)	1	1	0.06 (1/40/106)	**0.01 (1/7/24)**	**0.01 (1/7/24)**	0.01 (1/10/39)	0.01 (1/9/32)	0.01 (1/9/32)
FT(2,3)	2	1	6.51 (1/297/1423)	**0.02 (1/8/62)**	**0.02 (1/8/62)**	0.27 (1/30/251)	0.04 (1/11/87)	0.04 (1/11/87)
Coin(1)	3	1	**0.01 (2/2/9)**	**0.01 (2/2/9)**	**0.01 (2/2/9)**	**0.01 (2/2/9)**	**0.01 (2/2/9)**	**0.01 (2/2/9)**
Coin(2)	3	1	0.01 (9/9/31)	0.01 (9/9/31)	0.01 (9/9/31)	**0.01 (3/3/18)**	**0.01 (3/3/18)**	**0.01 (3/3/18)**
Coin(3)	3	1	**0.02 (39/39/79)**	**0.02 (39/39/79)**	**0.02 (39/39/79)**	0.03 (63/63/94)	0.03 (63/63/94)	0.03 (63/63/94)
HG	3	1	**0.01 (1/1/3)**	**0.01 (1/1/3)**	**0.01 (1/1/3)**	**0.01 (1/1/3)**	**0.01 (1/1/3)**	**0.01 (1/1/3)**
	4	1	0.01 (2/6/42)	**0.01 (2/5/37)**	**0.01 (2/5/37)**	0.01 (2/11/50)	**0.01 (2/5/37)**	**0.01 (2/5/37)**
	5	1	8.86 (38/185/1670)	16.74 (38/270/2532)	16.69 (38/270/2532)	–	**3.39 (8/113/1161)**	**3.38 (8/113/1161)**
AL	2	2	0.01 (9/22/32)	**0.01 (9/16/26)**	**0.01 (9/16/26)**	0.01 (7/24/32)	0.01 (7/23/33)	0.01 (7/23/33)
	2	3	0.01 (9/22/32)	**0.01 (9/16/26)**	**0.01 (9/16/26)**	0.01 (7/24/32)	0.01 (7/23/33)	0.01 (7/23/33)
	2	4	0.02 (9/22/32)	**0.02 (9/16/26)**	**0.02 (9/16/26)**	0.03 (7/24/32)	0.03 (7/23/33)	0.03 (7/23/33)
	2	5	0.05 (9/22/32)	**0.03 (9/16/26)**	**0.03 (9/16/26)**	0.06 (7/24/32)	0.06 (7/23/33)	0.06 (7/23/33)
	2	7	0.32 (9/22/32)	**0.21 (9/16/26)**	**0.22 (9/16/26)**	0.39 (7/24/32)	0.35 (7/23/33)	0.36 (7/23/33)
	2	10	5.06 (9/22/32)	**3.28 (9/16/26)**	**3.32 (9/16/26)**	6.02 (7/24/32)	5.51 (7/23/33)	5.52 (7/23/33)
SC(4)	3	1	0.01 (12/18/26)	0.01 (12/17/27)	0.01 (11/14/23)	0.01 (9/14/22)	**0.01 (9/11/20)**	**0.01 (9/11/19)**
SC(4)	7	1	0.01 (12/18/26)	0.01 (12/17/27)	0.01 (11/14/23)	0.01 (9/14/22)	**0.01 (9/11/20)**	**0.01 (9/11/19)**
SC(4)	8	1	0.02 (12/19/30)	0.02 (12/17/29)	0.02 (11/14/24)	0.03 (13/24/32)	0.02 (13/20/31)	**0.01 (9/12/21)**
SC(4)	3	3	0.01 (12/18/26)	0.01 (12/17/27)	0.01 (11/14/23)	0.02 (14/28/32)	0.02 (14/27/32)	**0.01 (9/12/20)**
SC(4)	3	4	0.02 (12/18/26)	0.02 (12/17/27)	0.01 (11/14/23)	0.02 (6/17/23)	0.02 (6/14/23)	**0.01 (8/12/20)**
SC(8)	3	1	0.03 (27/31/40)	0.02 (27/28/36)	0.02 (23/25/33)	0.03 (26/35/43)	0.03 (26/35/43)	**0.02 (23/24/31)**
Grap(2)	3	2	0.01 (4/4/15)	0.01 (4/4/15)	0.01 (4/4/15)	**0.01 (3/3/15)**	**0.01 (3/3/15)**	**0.01 (3/3/13)**
Grap(2)	4	1	1.58 (697/697/2607)	1.59 (697/697/2607)	1.10 (539/539/2068)	**0.01 (13/13/61)**	**0.01 (13/13/61)**	**0.01 (13/13/67)**
Grap(2)	4	2	0.02 (4/4/22)	0.02 (4/4/22)	0.02 (4/4/22)	**0.02 (3/3/20)**	**0.02 (3/3/20)**	**0.02 (3/3/18)**
Grap(2)	4	3	0.02 (4/4/22)	0.02 (4/4/22)	0.02 (4/4/22)	**0.03 (3/3/20)**	**0.03 (3/3/20)**	**0.02 (3/3/18)**
Grap(2)	4	4	0.04 (4/4/22)	0.04 (4/4/22)	0.04 (4/4/22)	**0.06 (3/3/20)**	**0.06 (3/3/20)**	**0.04 (3/3/18)**
Grap(3)	4	1	0.04 (4/4/27)	0.04 (4/4/27)	0.04 (4/4/27)	0.59 (115/115/472)	0.58 (115/115/472)	**0.04 (3/3/21)**
Gossip	3	2	0.01 (5/5/11)	0.01 (5/5/11)	0.01 (5/5/11)	**0.01 (3/3/7)**	**0.01 (3/3/7)**	**0.01 (3/3/7)**
	4	2	1.65 (47/47/133)	1.66 (47/47/133)	1.66 (47/47/133)	**0.16 (7/7/30)**	**0.16 (7/7/30)**	**0.16 (7/7/30)**
	5	2	2798.24 (2378/2378/6604)	2803.24 (2378/2378/6604)	2794.51 (2378/2378/6604)	**1.04 (14/14/99)**	**1.05 (14/14/99)**	**1.05 (14/14/99)**

Table 2: Experimental results on easy benchmarks. The first three columns indicate the name of instance, the number of agents, and the modal depth of the KBs, followed by subsequent columns presenting the results from various methods. Each result is represented as $T(A/B/C)$, indicating T seconds of run time, A nodes explored to locate the first goal-reachable node, B nodes explored in total, and C nodes expanded across the entire search graph. The symbol "–" indicates timeout. Results of the **smallest number of nodes explored** are in boldface.

	BFS	BFS$^+$	BFS$^+$K	Heu	Heu$^+$	Heu$^+$K
$\lvert$best$\rvert$	8	8	**15**	8	10	**15**

Table 3: Results on domains SC, Grapevine, and Gossip.

The results show that the version with both the enhancement strategy and the belief lock strategy (BFS$^+$K and Heu$^+$K) performs best in all 15 instances, showing the significance of the belief lock strategy in the domains where the goal involves maintaining certain information as a secret, undisclosed to some specific agents.

Because the existing benchmarks are easily solved, the improvements brought about by the enhancement strategies are limited. Search efficiency may already have reached optimal levels in certain domains, such as Grapevine, limiting the scope for significant improvement through these strategies.

4.2 Experiments on Hard Benchmarks

We conduct experiments on hard instances to further evaluate the effectiveness of the two new strategies.

Benchmarks We build hard instances by increasing the number of agents, objects such as rooms, blocks, and secrets, and setting complex goals. Then, we only keep the instances on which the BFS method finds the plan after expanding more than 1,000 nodes or finds no plans within 3,600 seconds. Eventually, it results in 33 hard instances.

The benchmarks are described below.

Collaboration-and-communication (Kominis and Geffner 2015). There is a corridor of some rooms. Several boxes are located in some of the rooms. The agents can move back and forth along this corridor. When an agent enters a room, she can see if a box is in the room. An agent can communicate information to another agent. Furthermore, there is a variant with a $cheat$ action. The $cheat$ action means agent i can mislead agent j about whether box b is in room r.

Finding-the-truth (Le et al. 2018): There are several boxes located in some rooms. The agents start with wrong

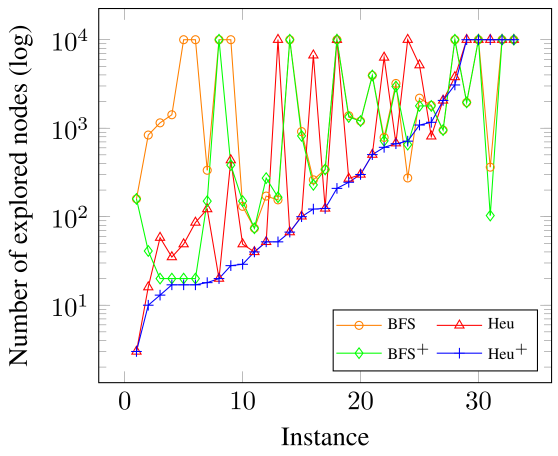

Figure 2: Explored nodes of four methods on hard instances

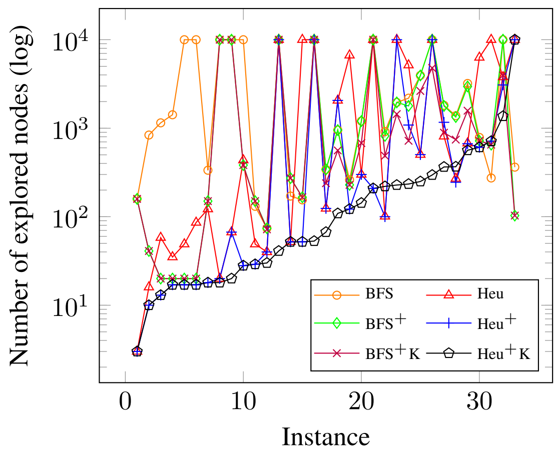

Figure 4: Explored nodes of six methods on hard instances

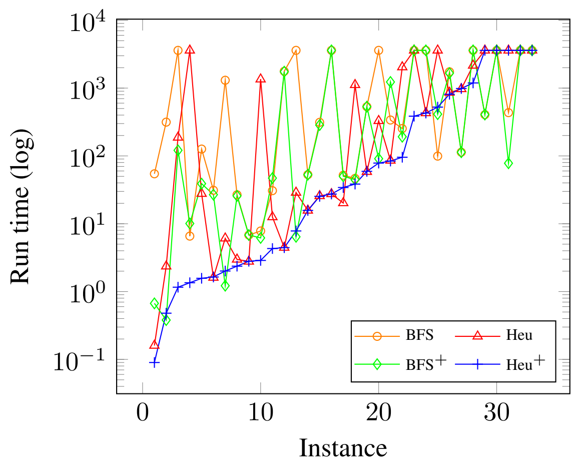

Figure 3: Run time (s) of four methods on hard instances

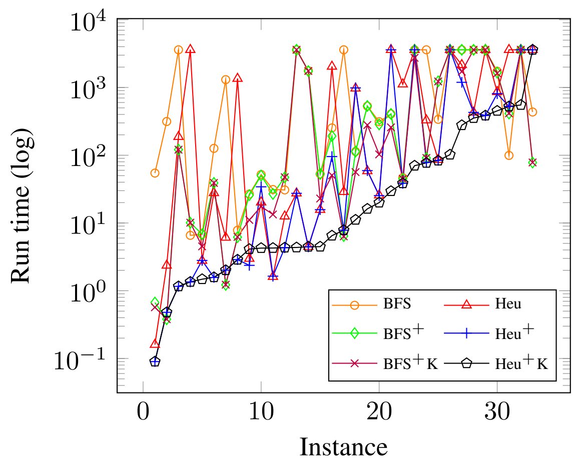

Figure 5: Run time (s) of six methods on hard instances

beliefs about the positions of the boxes. The agents can move between the rooms and check if a box is in a room. The goal is for the agents to find out the true locations of the boxes.

Selective-communication (Kominis and Geffner 2015) is introduced in Section 2.

Grapevine (Muise et al. 2015): A few guests attend a meeting in a villa with n rooms. Each guest has her own secret to share with others. Each guest can move between the rooms, and broadcast her secret to the guests in the same room. The goal is that only some of the guests obtain the designated secrets.

Hexa Game (van Ditmarsch 2001): There are some agents and cards, each with a unique color. Initially, everyone is holding a card, and can only see the color of her own card. A player can ask a question to another player whether her card is of a certain color. The question should always be honestly answered. The goal is for some agents to know the cards of some players.

Gossip (Attamah et al. 2014): Each of several friends has her own secret to share. Instead of sharing in public, they are only allowed to make a call to each other. In each call, they exchange all the secrets they know. The goal is that some agents get to know the secrets of some friends while some other agents do not.

Results Figure 2 and Figure 3 compare four methods: the BFS, BFS equipped with the enhancement strategy (BFS$^+$), the heuristic search and the heuristic search equipped with enhancement strategy (Heu$^+$). The x-axis represents various instances, arranged according to the increasing number of nodes explored by the Heu$^+$ method. Due to the extensive scale range, the y-axis employs a logarithmic scale (base 10) of the raw data.

The results in Figure 2 show that Heu$^+$ explored the smallest number of nodes to get the solution in most of the cases among the four methods. Specifically, for all 33 instances, Heu$^+$ performs best in 28 instances, and BFS$^+$ performs best in 3 instances. Figure 3 reports the corresponding time cost in the search, which is consistent with the results of the nodes explored. In particular, in our experiments, the enhanced version of BFS (BFS$^+$) achieved a substantial reduction in time cost, exceeding 80%, compared to the standard BFS in 7 distinct instances. Similarly, the enhanced heuristic search with our strategy (Heu$^+$) demonstrated a significant time saving of over 80% compared to the heuristic search in 8 instances. In summary, BFS$^+$ and Heu$^+$ achieve the best result except in a single instance.

Figure 4 and Figure 5 compare six methods: BFS, BFS$^+$, BFS$^+$ equipped with the belief lock strategy (BFS$^+$K), heuristic search, Heu$^+$, and Heu$^+$ equipped with the belief lock strategy (Heu$^+$K). The x-axis represents various instances, arranged according to the increasing number of nodes explored by the Heu$^+$K method.

The results in Figure 4 show that the belief lock strategy further improves the performance of BFS and heuristic search. In the figure, BFS$^+$K and Heu$^+$K explored the smallest number of nodes to obtain the solution in 30 out of 33 instances. The best-resulting method, Heu$^+$K, performs best in 29 instances. Figure 5 reports the corresponding time cost in search. When integrated with the belief lock strategy, BFS$^+$K demonstrated a time saving of more than 50% compared to its predecessor BFS$^+$ in 9 distinct cases. Likewise, the Heu$^+$K, which incorporates this strategy into the heuristic search, yielded a time reduction of over 50% when measured against Heu$^+$ in 11 cases, underscoring the significant efficiency gains achieved through this advanced strategy. In summary, BFS$^+$K and Heu$^+$K achieve the best except in 3 instances.

Because our new strategies are based on heuristic observations, there are instances where they may not be effective and in some cases they might even lead to suboptimal results. Consider, for instance, the domain of the Hexa Game, where actions are only sensing actions, meaning that the resulting solution (policy) to this domain is a full binary tree under contingent planning. In such a scenario, the simple BFS method may outperform all other methods.

5 Conclusions

Multi-agent epistemic planning is an emerging field at the intersection of *automated planning* and *epistemic logic*. In this paper, we have introduced two new strategies designed to refine the MEPK search algorithms. The enhancement strategy leverages information from the search path to dynamically adjust heuristics, aiming to speed up the search by decreasing unnecessary node exploration. The belief lock strategy prevents the planner from continuing to search a particular state that is unable to progress to the goal state due to an agent's possession of certain knowledge. Then, we demonstrate that the performance of MEPK, equipped with the new strategies, has improved. In addition, we have constructed hard instances derived from existing benchmarks. The evaluation results affirm the effectiveness of the new strategies, marking a significant leap in computational efficiency for MEP.

In the future, we plan to extend Proposition 1 to encompass more general scenarios, such as cases where the information kept unknown to certain agents includes not just literals, but also the beliefs of other agents.

Acknowledgements

We are grateful to anonymous reviewers whose comments and suggestions have helped us improve the paper.

A Proof of Proposition 1

The higher-order revision operator in MEPK is defined on ACDF, a normal form for KD45$_n$ to support efficient reasoning and progression. First, we introduce the cover modality and ACDF.

We let $L_a\phi$ stand for $\neg B_a \neg\phi$, and we use $L_a\Phi$ to represent the conjunction of $L_a\phi$ where $\phi \in \Phi$.

Definition 12. Let $a \in \mathcal{A}$, and Φ a finite set of formulas. The cover modality is defined as follows:

$$\nabla_a\Phi \doteq B_a(\bigvee \Phi) \wedge L_a\Phi.$$

Intuitively, $\nabla_a\Phi$ means that each world considered possible by agent a satisfies an element of Φ, and each element of Φ is satisfied by some world considered possible by agent a.

Definition 13. The set of *cover disjunctive formulas* (CDFs) is inductively defined as follows:

1. A propositional term, *i.e.*, a conjunction of propositional literals, is a CDF;
2. If ϕ_0 is a propositional CDF, and for each $a \in \mathcal{B} \subseteq \mathcal{A}$, Φ_a is a finite set of CDFs, then $\phi_0 \wedge \bigwedge_{a\in\mathcal{B}} \nabla_a\Phi_a$ is a CDF, called a *CDF term*;
3. If Φ is a non-empty finite set of CDF terms, then $\bigvee \Phi$ is a CDF, called a *disjunctive CDF*.

Definition 14. The non-alternating factor of a formula ϕ, denoted by $na(\phi)$, is the number of modal operators of an agent which directly occur inside those of the same agent. We say that a formula is *alternating* if its non-alternating factor is 0.

Definition 15. We call an alternating CDF an ACDF (alternating cover disjunctive formula).

For example, $\nabla_a\{\top, q\} \wedge \nabla_b\{\top, \nabla_a\{\top, \neg q\}\}$ is an ACDF; but the CDF $\nabla_a\{\nabla_b\{p\}, \nabla_a\{\nabla_a\{\neg q\}\}\}$ is not, and its non-alternating factor is 2 since it happens twice that ∇_a directly appears after ∇_a. Hales, French, and Davies (2012) introduced the notion of ACDFs, and showed that in KD45$_n$, every formula in $\mathcal{L}_K$ is equivalent to such a formula.

Next, we introduce the higher-order revision operator in two cases used in MEPK.

Definition 16. Let ϕ and ϕ' be ACDFs, γ a DNF formula. The revision of ϕ with ϕ' under γ, denoted $\phi \circ_\gamma \phi'$, is recursively defined as follows:

1. When ϕ and ϕ' are propositional, the result is $\phi \circ_s (\phi' \wedge \gamma)$, where $\circ_s$ is Satoh's revision operator (Satoh 1988).
2. When $\phi = \phi_0 \wedge \bigwedge_{a\in\mathcal{B}} \nabla_a\Phi_a$, $\phi' = \phi'_0 \wedge \bigwedge_{a\in\mathcal{B}'} \nabla_a\Phi'_a$, and $\phi \wedge \phi'$ is satisfiable w.r.t. γ, $\phi \circ_\gamma \phi'$ is defined as:

$$(\phi_0 \circ_\gamma \phi'_0) \wedge \bigwedge_{a\in\mathcal{B}-\mathcal{B}'} \nabla_a\Phi_a \wedge \bigwedge_{a\in\mathcal{B}'-\mathcal{B}} \nabla_a\Phi'_a \wedge$$
$$\bigwedge_{a\in\mathcal{B}\cap\mathcal{B}'} \nabla_a[(\Phi_a \circ_\gamma \bigvee \Phi'_a) \cup (\Phi'_a \circ_\gamma \bigvee \Phi_a)].$$

The intuition of rule 2 is the conjunction property of cover modality: $\nabla_a \Phi \wedge \nabla_a \Phi' \Leftrightarrow \nabla_a[\Phi \wedge (\bigvee \Phi') \cup \Phi' \wedge (\bigvee \Phi)]$.

Next, we present the proof.

Proof of Proposition 1. We prove by induction on the sequence of actions τ.

Base case (τ consists of only one action t): For simplicity, let us focus on the beliefs of agent a, and modal depth is 1.

Let $prog(\mathcal{I}, \sigma) = \phi_0 \wedge B_a\phi_a$ and $\mathcal{G} = \psi_0 \wedge B_a\psi_a$, where ϕ_0, ϕ_a, ψ_0 and $\psi_a \in \mathcal{L}_0$.

For computation, MEPK compiles $prog(\mathcal{I}, \sigma)$ and $\mathcal{G}$ to ACDFs: $\phi_0 \wedge \nabla_a\{\phi_a\}$ and $\psi_0 \wedge \nabla_a\{\psi_a\}$, respectively. Note that in KD45, $B_a\phi \models L_a\phi$, so $B_a\phi \Leftrightarrow \nabla_a\{\phi\}$.

1. If t is a sensing action which is a tuple $\langle pre, pos, neg\rangle$, let the positive result $pos(t) = \phi_0' \wedge B_a\phi_a'$, then the progression of $prog(\mathcal{I}, \sigma)$ w.r.t. t with positive result is as follows:
   - If $prog(\mathcal{I}, \sigma) \wedge pos(t)$ is propositionally unsatisfiable w.r.t. γ, then (by Definition 8)
   $$prog(prog(\mathcal{I}, \sigma), \tau) \Leftrightarrow \bot$$
   - Otherwise, we have
   $$\begin{aligned}&prog(prog(\mathcal{I}, \sigma), \tau)\\ \Leftrightarrow\ & prog(\mathcal{I}, \sigma) \circ_\gamma pos(t)\\ \Leftrightarrow\ & (\phi_0 \wedge B_a\phi_a) \circ_\gamma (\phi_0' \wedge B_a\phi_a')\\ \Leftrightarrow\ & (\phi_0 \wedge \nabla_a\{\phi_a\}) \circ_\gamma (\phi_0' \wedge \nabla_a\{\phi_a'\})\\ \Leftrightarrow\ & (\phi_0 \circ_\gamma \phi_0') \wedge \nabla_a\{\phi_a \circ_\gamma \phi_a'\} \text{ (rule 2, Definition 16)}\\ \Leftrightarrow\ & [\phi_0 \circ_s (\phi_0' \wedge \gamma)] \wedge B_a[\phi_a \circ_s (\phi_a' \wedge \gamma)]\end{aligned}$$

   Given $prog(\mathcal{I}, \sigma) \models_\gamma B_ap$, and $pos(t) \not\models_\gamma \neg B_ap$, then $\phi_a \models_\gamma p$, and $\phi_a' \wedge p$ is satisfiable w.r.t. γ.
   Thus $[\phi_a \circ_s (\phi_a' \wedge \gamma)] \models_\gamma p$, and $prog(prog(\mathcal{I}, \sigma), \tau) \models_\gamma B_ap$.
   Also as $\mathcal{G} \models_\gamma \neg B_ap$, then $\psi_a \models_\gamma \neg p$, and $\neg\psi_a \models_\gamma p$. Thus $[\phi_a \circ_s (\phi_a' \wedge \gamma)] \wedge \neg\psi_a$ is satisfiable.
   However, $prog(prog(\mathcal{I}, \sigma), \tau) \models_\gamma \mathcal{G}$ requires $[\phi_a \circ_s (\phi_a' \wedge \gamma)] \wedge \neg\psi_a$ to be unsatisfiable, which is a contradiction. Thus $prog(prog(\mathcal{I}, \sigma), \tau) \not\models_\gamma \mathcal{G}$.
   The progression of $prog(\mathcal{I}, \sigma)$ w.r.t. t with the negative result neg is similar.
2. If t is a deterministic action, the progression of $prog(\mathcal{I}, \sigma)$ w.r.t. t is similar to sensing action except that the belief change operator is higher-order update operator.

Induction step (τ consists of two or more actions): Let $\tau = (t; \tau')$, then

$$\begin{aligned}&prog(prog(\mathcal{I}, \sigma), \tau)\\ \Leftrightarrow\ & prog(prog(\mathcal{I}, \sigma), (t; \tau'))\\ \Leftrightarrow\ & prog(prog(prog(\mathcal{I}, \sigma), t), \tau')\end{aligned}$$

By the base case, we have $prog(prog(\mathcal{I}, \sigma), \tau) \models_\gamma B_ap$. Then, we repeatedly retrieve actions from τ' until it consists of only one action. □

References

Attamah, M.; van Ditmarsch, H.; Grossi, D.; and van der Hoek, W. 2014. Knowledge and gossip. In *ECAI-14*, 21–26.

Aucher, G., and Bolander, T. 2013. Undecidability in epistemic planning. In *IJCAI-13*, 27–33.

Baral, C.; Bolander, T.; van Ditmarsch, H.; and McIlrath, S. 2017. Epistemic planning (dagstuhl seminar 17231). *Dagstuhl Reports* 7(6).

Belle, V.; Bolander, T.; Herzig, A.; and Nebel, B. 2023. Epistemic planning: Perspectives on the special issue. *Artificial Intelligence* 316:103842.

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

Bolander, T.; Dissing, L.; and Herrmann, N. 2021. Del-based epistemic planning for human-robot collaboration: Theory and implementation. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, 120–129.

Charrier, T.; Maubert, B.; and Schwarzentruber, F. 2016. On the impact of modal depth in epistemic planning. In *IJCAI-16*, 1030–1036.

Cong, S. L.; Pinchinat, S.; and Schwarzentruber, F. 2018. Small undecidable problems in epistemic planning. In *IJCAI-18*, 4780–4786.

Cooper, M. C.; Herzig, A.; Maffre, F.; Maris, F.; and Régnier, P. 2016a. A simple account of multi-agent epistemic planning. In *ECAI-16*, 193–201.

Cooper, M. C.; Herzig, A.; Maffre, F.; Maris, F.; and Régnier, P. 2016b. Simple epistemic planning: Generalised gossiping. In *ECAI-16*, 1563–1564.

Cooper, M. C.; Herzig, A.; Maffre, F.; Maris, F.; Perrotin, E.; and Régnier, P. 2021. A lightweight epistemic logic and its application to planning. *Artificial Intelligence* 298:103437.

Dissing, L., and Bolander, T. 2020. Implementing theory of mind on a robot using dynamic epistemic logic. In *IJCAI-20*, 1615–1621.

Engesser, T.; Bolander, T.; Mattmüller, R.; and Nebel, B. 2017. Cooperative epistemic multi-agent planning for implicit coordination. In *Proceedings of the Ninth Workshop on Methods for Modalities*, 75–90.

Fabiano, F.; Burigana, A.; Dovier, A.; and Pontelli, E. 2020. Efp 2.0: A multi-agent epistemic solver with multiple e-state representations. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, 101–109.

Hales, J.; French, T.; and Davies, R. 2012. Refinement quantified logics of knowledge and belief for multiple agents. In *Advances in Modal Logic*, volume 9, 317–338.

Heath, C., and Heath, D. 2006. The curse of knowledge. *Harvard Business Review* 84(12):20–23.

Hu, G.; Miller, T.; and Lipovetzky, N. 2022. Planning with perspectives–decomposing epistemic planning using functional strips. *Journal of Artificial Intelligence Research* 75:489–539.

Kominis, F., and Geffner, H. 2015. Beliefs in multiagent planning: From one agent to many. *Proceedings of the 25th International Conference on Automated Planning and Scheduling* 147–155.

Le, T.; Fabiano, F.; Son, T. C.; and Pontelli, E. 2018. EFP and PG-EFP: epistemic forward search planners in multi-agent domains. In *Proceedings of the International Conference on Automated Planning and Scheduling*, 161–170.

Lin, F. 2004. Discovering state invariants. *KR* 4:536–544.

Löwe, B.; Pacuit, E.; and Witzel, A. 2011. DEL planning and some tractable cases. In *Proceedings of the Third International Workshop on Logic, Rationality, and Interaction*, 179–192.

Muise, C. J.; Belle, V.; Felli, P.; McIlraith, S. A.; Miller, T.; Pearce, A. R.; and Sonenberg, L. 2015. Planning over multi-agent epistemic states: A classical planning approach. In *AAAI-15*, 3327–3334.

Muise, C.; Belle, V.; Felli, P.; McIlraith, S.; Miller, T.; Pearce, A. R.; and Sonenberg, L. 2022. Efficient multi-agent epistemic planning: Teaching planners about nested belief. *Artificial Intelligence* 302:103605.

Satoh, K. 1988. Nonmonotonic reasoning by minimal belief revision. In *Proc. the First International Conference on Fifth Generation Computer Systems*.

Thielscher, M. 2017. Gdl-iii: A description language for epistemic general game playing. In *IJCAI-17*, 1276–1282.

To, S. T.; Son, T. C.; and Pontelli, E. 2011. Contingent planning as and/or forward search with disjunctive representation. *Proceedings of the 21st International Conference on Automated Planning and Scheduling* 258–265.

van Ditmarsch, H.; van der Hoek, W.; and Kooi, B. P. 2007. *Dynamic epistemic logic*. Springer.

van Ditmarsch, H. 2001. Knowledge games. *Bulletin of Economic Research* 249–273.

Wan, H.; Fang, B.; and Liu, Y. 2021. A general multi-agent epistemic planner based on higher-order belief change. *Artificial Intelligence* 301:103562.

Yu, Q.; Wen, X.; and Liu, Y. 2013. Multi-agent epistemic explanatory diagnosis via reasoning about actions. In *IJCAI-13*, 1183–1190.