An Improved Multi-agent Epistemic Planner via Higher-Order Belief Change Based on Heuristic Search

Zhongbin Wu(✉)

Department of Computer Science, Sun Yat-sen University, Guangzhou 510006, China
wuzhb3@mail2.sysu.edu.cn

Abstract. Recently, multi-agent epistemic planning has drawn attention from both dynamic logic and planning communities. Existing implementations are based on compilation into classical planning, which suffers from limitations such as incapability to handle disjunctive beliefs, or higher-order belief change and forward state space search, as exploited by the planner MEPK. However, MEPK does not scale well. In this paper, we propose two improvements for MEPK. Firstly, we exploit another normal form for multi-agent KD45, which is more space efficient than the normal form used by MEPK, and propose efficient reasoning, revision, and update algorithms for it. Secondly, we propose a heuristic function for multi-agent epistemic planning, and apply heuristic search algorithm AO* with cycle checking and two heuristic pruning strategies. We implement a multi-agent epistemic planner called MEPL. Our experimental results show that MEPL outperforms MEPK in most planning instances, and solves a number of instances which MEPK cannot solve.

Keywords: Modal logic · Multi-agent epistemic planning
Heuristic search

1 Introduction

Many intelligent tasks involve the collaboration and communication among multiple agents, raising the need of reasoning about knowledge and beliefs, and their change. For example, reasoning is performed before taking an action to check whether its precondition has been satisfied. Belief change is incurred because the performance of actions may change not only the environment but also the beliefs of agents. Reasoning and progression about higher-order knowledge and beliefs are also needed in some cases. For example, an agent may wish that she knows some information and the other agents know that she knows the information.

In recent years, multi-agent epistemic planning has drawn great attention from both dynamic logic and planning communities. On the theory side, Bolander and Andersen [1] formalized multi-agent epistemic planning based on dynamic epistemic logic [2] and showed that it is undecidable in general. On the implementation side, both Kominis and Geffner [3], and Muise *et al.* [4]

W. Liu et al. (Eds.): KSEM 2018, LNAI 11062, pp. 102–116, 2018.
https://doi.org/10.1007/978-3-319-99247-1_10

have shown how to solve restricted versions of multi-agent epistemic planning problems by resorting to classical planning. However, this approach suffers from many limitations, like generating only linear plans, restriction to public actions, and incapability to handle disjunctive beliefs, etc.

Very recently, Huang *et al.* [5] have proposed a general representation framework for multi-agent epistemic planning. The framework can handle arbitrary higher-order multi-agent epistemic formulas and the solution is an action tree branching on sensing results. To support efficient reasoning and progression, they resorted to a normal form for multi-agent KD45 [6] logic called alternating cover disjunctive formulas (ACDFs) [7]. Besides, they also implemented a multi-agent epistemic planner, called MEPK, by using forward state space search. However, MEPK does not scale well. Firstly, transforming a random $\mathrm{KD45}_n$ formula into an ACDF may result in at most a single exponential blowup in size, which makes MEPK spend too much time on preprocessing. Secondly, the reasoning and progression algorithm proposed is much sensitive to the formula size of ACDFs. And finally, they implemented the planner based on a naive forward state space search algorithm, which also prevents MEPK from handling a larger domain.

In this paper, we propose two improvements for the multi-agent epistemic planner MEPK. Firstly, we exploit alternating disjunctive normal form for multi-agent KD45, which is more space efficient than ACDFs and also saves time in compilation, and propose efficient reasoning, revision, and update algorithms for it. Secondly, we propose a heuristic function for multi-agent epistemic planning, and apply heuristic search algorithm AO* with cycle checking and two heuristic pruning strategies for acceleration. Finally, based on our theoretic work, we have implemented a multi-agent epistemic planner called MEPL. Our experimental results show that MEPL outperforms MEPK in most planning instances, and solves a number of instances which MEPK cannot solve.

2 Preliminaries

2.1 Multi-agent Modal Logic $\mathrm{KD45}_n$

Throughout our paper, we let $\mathcal{A}$ denote a finite set of agents, and $\mathcal{P}$ a finite set of atoms. We use $\phi, \psi, \varphi, \delta$ to represent a formula, Φ, Ψ to represent the set of formulas, and $\top$ and $\bot$ to represent $True$ and $False$, respectively. Besides, we let $\bigvee \Phi$ (resp. $\bigwedge \Phi$) denote the disjunction (resp. conjunction) of members in Φ.

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

$$\varphi ::= p \mid \neg\phi \mid (\phi \wedge \psi) \mid K_a\phi \mid C\phi,$$

where $a \in \mathcal{A}, p \in \mathcal{P}, \phi, \psi \in \mathcal{L}_{\mathcal{KC}}$. We use $\mathcal{L}_{\mathcal{K}}$ for the language without C operator, and $\mathcal{L}_0$ for propositional language.

Intuitively, $K_a\phi$ means agent a knows ϕ and $C\phi$ means commonly knowing ϕ. Similar to Huang *et al.* [5], we only discuss the case of propositional common knowledge, i.e., $C\phi$ where $\phi \in \mathcal{L}_0$, and we also call it a constraint.

Definition 2. *A frame is a pair (W, R), where W is a non-empty set of possible worlds; for each agent $a \in \mathcal{A}$, R_a is a binary relation on W, called the accessibility relation for agent a and $R = \bigcup_{a\in\mathcal{A}} R_a$.*

We say that an accessibility relation R_a is serial if for any $w \in W$ we have $w' \in W$ s.t. $(w, w') \in R_a$; R_a is transitive if $\forall w, u, v \in W, (w, u) \in R_a$ and $(u, v) \in R_a$ implies $(w, v) \in R_a$; R_a is Euclidean if $\forall w, w_1, w_2 \in W, (w, w_1) \in R_a$ and $(w, w_2) \in R_a$ implies $(w_1, w_2) \in R_a$. 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 that maps each $w \in W$ to a subset of $\mathcal{P}$. A pointed Kripke model is a pair (M, w), where M is a Kripke model and w is a world of M, which is called the actual world.*

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

- *$M, w \models p$ 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 $M, w \models \psi$;*
- *$M, w \models K_a\phi$ iff for all v s.t. $(w, v) \in R_a$, $M, v \models \phi$;*
- *$M, w \models C\phi$ iff for all v s.t. $(w, v) \in R_{\mathcal{T}}$, $M, v \models \phi$, where $R_{\mathcal{T}}$ is the transitive closure of the union of R.*

We say ϕ is satisfiable if there is a pointed KD45$_n$ model s.t. $M, w \models \phi$. We say ϕ entails ψ, written $\phi \models \psi$, if for all model (M, w), $M, w \models \phi$ implies $M, w \models \psi$; ϕ and ψ are equivalent, written $\phi \Leftrightarrow \psi$, if $\phi \models \psi$ and $\psi \models \phi$. Similarly, we say ϕ is satisfiable under the constraint γ if there is a pointed KD45$_n$ model s.t. $M, w \models \phi \wedge \gamma \wedge C\gamma$. We say ϕ entails ψ under the constraint γ, written $\phi \models_\gamma \psi$, if for all model (M, w), $M, w \models \phi \wedge \gamma \wedge C\gamma$ implies $M, w \models \psi \wedge \gamma \wedge C\gamma$; ϕ and ψ are equivalent under the constraint γ, written $\phi \Leftrightarrow_\gamma \psi$, if $\phi \models_\gamma \psi$ and $\psi \models_\gamma \phi$.

Next we introduce the normal form used in our paper. We let $L_a\phi$ denote $\neg K_a \neg\phi$, and $L_a\Phi$ denote the conjunction of $L_a\phi$, where $\phi \in \Phi$.

Definition 5. *We define the set of $\mathcal{L}_K$ normal terms, called KTerms, as follows:*

- *A propositional term, i.e., a conjunction of literals, is a normal term;*
- *A formula of the form $\phi_0 \wedge \bigwedge_{a\in\mathcal{A}}(K_a\phi_a \wedge L_a\Psi_a)$ is a normal term, where*

  - *ϕ_0 is a propositional term;*
  - *ϕ_a is a disjunction of $\mathcal{L}_K$ normal terms;*
  - *Ψ_a is a set of $\mathcal{L}_K$ normal terms.*

We say that a KTerm has the alternating property if modal operators of an agent does not occur directly inside modal operators of the same agent. For example, $K_a(K_b(K_a(p)))$ has the alternating property while $K_a(K_a(p))$ does not.

Definition 6. *A disjunction of alternating KTerms is called an alternating DNF (ADNF).*

It is easy to prove the following:

Proposition 1. *In $KD45_n$, every formula in $\mathcal{L}_\mathcal{K}$ can be converted to an equivalent ADNF.*

Proposition 2. *Let $\delta = \phi_0 \wedge \bigwedge_{a\in\mathcal{A}}(K_a\phi_a \wedge L_a\Psi_a)$, $\delta' = \phi'_0 \wedge \bigwedge_{a\in\mathcal{A}}(K_a\phi'_a \wedge L_a\Psi'_a)$ be two alternating KTerms and γ be a constraint. $\delta \models_\gamma \delta'$ iff the following hold:*

- *$\phi_0 \wedge \gamma \models \phi'_0$ propositionally;*
- *For each $a \in \mathcal{A}$, $\phi_a \models_\gamma \phi'_a$;*
- *For each $a \in \mathcal{A}$, for every $\psi'_a \in \Psi'_a$ there exists $\psi_a \in \Psi_a$ s.t. $\phi_a \wedge \psi_a \models \psi'_a$.*

2.2 Belief Revision and Update

Revision and update are two kinds of methods for belief change with new belief. Revision concerns belief change due to the partial or incorrect information while update concerns belief change caused by taking actions. Many guidelines about revision and update have been proposed, which include belief revision [8], belief update [9] and iterated belief revision [10], etc. We use $\circ$ to denote the revision operator and $\diamond$ to the denote the update operator. Katsuno and Mendelzon [9] have introduced the main difference between revision and update in a model-theoretic way: let ϕ and ψ be two formulas, $\phi \circ \psi$ selects from ψ the models that are closest to that in ϕ, and $\phi \diamond \psi$ selects, for each model M of ϕ, the set of models from ψ that are closest to M, intuitively. As easy properties, we can get:

- Satisfiability Property of revision: if $\phi \wedge \psi$ is satisfiable, then $\phi \circ \psi \Leftrightarrow \phi \wedge \psi$.
- Distribution Property of update: $(\phi_1 \vee \phi_2) \diamond \psi \Leftrightarrow (\phi_1 \diamond \psi) \vee (\phi_2 \diamond \psi)$.

2.3 Modeling Framework

We now introduce the modeling framework for multi-agent epistemic planning proposed by Huang *et al.* [5] with an simple example.

Example 1. There are three children a, b, c, and either one or two of them are muddy. Every child can sense whether another child is muddy. And every child can ask another child a_1 that whether a_1 can identify herself, and the question will be answered honestly. Now the child a doesn't know whether she is muddy, and the goal is that the child a gets to know she is muddy or she isn't.

Definition 7. *A multi-agent epistemic planning (MEP) 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 the set of agents, $\mathcal{P}$ is the set of atoms, $\mathcal{D}$ is the set of deterministic actions, $\mathcal{S}$ is the set of sensing actions, $\mathcal{I} \in \mathcal{L}_\mathcal{K}$ is the initial state, $\mathcal{G} \in \mathcal{L}_\mathcal{K}$ is the goal and $\gamma \in \mathcal{L}_0$ is the constraint.*

Now we can formalize Example 1 as:

- The atom is $muddy(i)$, which means the child i is muddy;
- $\mathcal{D} = \emptyset$, and the sensing actions are: $sense(a_1, a_2)$, the child a_1 sense the child a_2; $ask(a_1, a_2)$, the child a_1 ask the child a_2;
- $\mathcal{I} = L_a(muddy(a)) \wedge L_a(\neg muddy(a))$;
- $\mathcal{G} = K_a(muddy(a)) \vee K_a(\neg muddy(a))$;
- $\gamma = (muddy(a) \wedge muddy(b) \wedge \neg muddy(c)) \vee (muddy(a) \wedge \neg muddy(b) \wedge muddy(c)) \vee (\neg muddy(a) \wedge muddy(b) \wedge muddy(c)) \vee (muddy(a) \wedge \neg muddy(b) \wedge \neg muddy(c)) \vee (\neg muddy(a) \wedge muddy(b) \wedge \neg muddy(c)) \vee (\neg muddy(a) \wedge \neg muddy(b) \wedge muddy(c))$, means that either one or two of them are muddy.

Definition 8. *A deterministic action is a pair $\langle pre, effs \rangle$, where $pre \in \mathcal{L}_\mathcal{K}$ is the precondition, and $effs$ is a set of conditional effects, each of which is a pair $\langle con, cef \rangle$, where $con, cef \in \mathcal{L}_\mathcal{K}$ are the condition and the conditional effect.*

Definition 9. *A sensing action is a triple $\langle pre, pos, neg \rangle$, where $pre, pos, neg \in \mathcal{L}_\mathcal{K}$ are the precondition, positive result and negative result, respectively.*

For example, we can formalize the sensing action $sense(a_1, a_2)$ in Example 1 as: $pre(sense(a_1, a_2)) = True$, $pos(sense(a_1, a_2)) = K_{a_1}(muddy(a_2))$, and $neg(sense(a_1, a_2)) = K_{a_1}(\neg muddy(a_2))$.

An action a is executable on KB ϕ under constraint γ if $\phi \models_\gamma pre(a)$. Assume that $\phi \models_\gamma pre(a)$, the progression of ϕ wrt action a is defined by using update operator under constraint γ, written $\diamond_\gamma$, for deterministic action and revision operator under constraint γ, written $\circ_\gamma$, for sensing action.

Definition 10. *Let $\phi \in \mathcal{L}_\mathcal{K}$, and a_d a deterministic action with $effs(a_d) = \{\langle con_1, cef_1 \rangle, \ldots, \langle con_n, cef_n \rangle\}$. Let $\{\langle c_1, e_1 \rangle, \ldots, \langle c_k, e_k \rangle\} \subseteq eff(a_d)$ s.t. $\phi \models_\gamma c_i$, for $1 \leq i \leq k$. The progression of ϕ wrt a_d under the constraint γ, written $prog(\phi, a_d)$, is defined as $prog(\phi, a_d) = ((\phi \diamond_\gamma e_1)...) \diamond_\gamma e_k$.*

Definition 11. *Let $\phi \in \mathcal{L}_\mathcal{K}$, and $a_s = \{pre, pos, neg\}$ a sensing action. The progression of ϕ wrt a_s under the constraint γ, written $prog(\phi, a_s)$, is a pair $\langle \phi^+, \phi^- \rangle$, where $\phi^+ = \phi \circ_\gamma pos(a_s)$ and $\phi^- = \phi \circ_\gamma neg(a_s)$.*

A solution of a multi-agent epistemic planning problem is an action tree branching on sensing actions. We use ϵ to denote an empty action tree.

Definition 12. *Let $\mathcal{Q} = \langle \mathcal{A}, \mathcal{P}, \mathcal{D}, \mathcal{S}, \mathcal{I}, \mathcal{G}, \gamma \rangle$ be a MEP problem. The set $\mathcal{T}$ of action tree is defined recursively:*

- *ϵ is in $\mathcal{T}$;*
- *If $a_d \in \mathcal{D}$ and $T \in \mathcal{T}$, then $a_d; T$ is in $\mathcal{T}$;*
- *If $a_s \in \mathcal{S}$ and $T^+, T^- \in \mathcal{T}$, then $a_s; (T^+ | T^-)$ is in $\mathcal{T}$.*

Definition 13. *Let $\phi \in \mathcal{L}_\mathcal{K}$, T an action tree. The progression of ϕ wrt T, written $prog(\phi, T)$ is defined recursively:*

- $prog(\phi, \epsilon) = \{\phi\}$;
- *If* $\phi \models_\gamma pre(a_d)$, $prog(\phi, a_d; T') = prog(prog(\phi, a_d), T')$;
- *If* $\phi \models_\gamma pre(a_s)$, $prog(\phi, a_s; (T^+|T^-)) = prog(\phi^+, T^+) \bigcup prog(\phi^-, T^-)$, *where* $prog(\phi, a_s) = \langle \phi^+, \phi^- \rangle$;
- *Otherwise,* $prog(\phi, T)$ *is undefined.*

Definition 14. *Let* $\mathcal{Q} = \langle \mathcal{A}, \mathcal{P}, \mathcal{D}, \mathcal{S}, \mathcal{I}, \mathcal{G}, \gamma \rangle$ *be a MEP problem, an action tree* T *is a solution for* $\mathcal{Q}$ *if* $prog(\mathcal{I}, T)$ *is defined, and* $\forall \phi \in prog(\mathcal{I}, T)$, $\phi \models_\gamma \mathcal{G}$.

Figure 1 shows a solution action tree for Example 1. Firstly, the child a sense the child b and then the child c. If both b and c are muddy, then the child a knows that she is not muddy. Otherwise, the child a ask the child b, and if b can identify herself, then the child a knows that she is the same as the child c; otherwise, she is different from the child c.

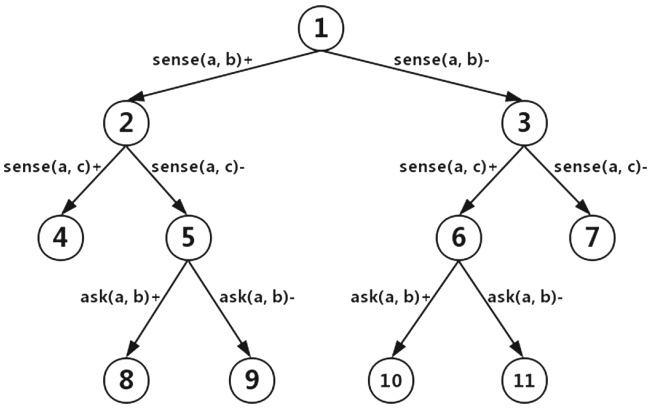

Fig. 1. Solution action tree for Example 1

3 Reasoning and Progression

In this section, we will introduce the basic algorithms for ADNFs: satisfiability, strong entailment and strong equivalence, revision and update. In our planner, in order to support efficient reasoning and progression, we use DNFs to represent the constraint and ADNFs for other formulas, including the initial state, the goal, the condition and the effect of actions.

3.1 Satisfiability

To support efficient reasoning, our planner firstly transforms $pre(a)$ and $\mathcal{G}$ into the negation form. Thus we can solve the reasoning problem by solving an equivalent satisfiability problem, i.e., $\phi \models_\gamma \phi'$ iff $\phi \wedge \neg\phi'$ is not satisfiable wrt γ. Next we introduce the satisfiability algorithm for alternating KTerms and ADNFs.

Proposition 3. *An alternating KTerm* $\phi = \phi_0 \wedge \bigwedge_{a\in\mathcal{A}}(K_a\phi_a \wedge L_a\Psi_a)$ *is satisfiable wrt constraint* γ *iff the following hold:*

1. $\phi_0 \wedge \gamma$ *is propositionally satisfiable;*
2. *For each* $a \in \mathcal{A}$, ϕ_a *is satisfiable wrt* γ*;*
3. *For each* $a \in \mathcal{A}$, *for each* $\psi_a \in \Psi_a$, $\phi_a \wedge \psi_a$ *is satisfiable wrt* γ.

Proof. The proof for the only-if direction is direct, so we only prove the if direction. We can construct a KD45$_n$ model (M, w) satisfying $\phi \wedge \gamma \wedge C\gamma$ by following the steps: (1) Create a new world w satisfying $\phi_0 \wedge \gamma$; (2) By induction, for each $a \in \mathcal{A}$, for each $\psi_a \in \Psi_a$, there exists a model (M_{ψ_a}, w_{ψ_a}) satisfying $\phi_a \wedge \psi_a \wedge \gamma \wedge C\gamma$. Thus, add (M_{ψ_a}, w_{ψ_a}) into M, and add the relation $(w, w_{\psi_a}) \in R_a$ [**serial**]; (3) Add bidirectional a-relations between w_{ψ_a} and $w_{\psi'_a}$, where $\psi_a, \psi'_a \in \Psi_a$ [**Euclidean**]; (4) Since ϕ is alternating, for each $a \in \mathcal{A}$ and for each $\psi_a \in \Psi_a$, the outmost modality in ϕ_a and ψ_a is neither K_a nor L_a, thus there is no more relation between (M, w) and (M_{ψ_a}, w_{ψ_a}) [**transitive**].

Proposition 4. *Let* ϕ *and* ϕ' *be two ADNFs.*

- *When* ϕ *and* ϕ' *are propositional terms,* $\phi \wedge \phi'$ *is satisfiable wrt constraint* γ *iff* $\phi \wedge \phi' \wedge \gamma$ *doesn't have complementary literals;*
- *When* $\phi = \bigvee \Psi$ *and* $\phi' = \bigvee \Psi'$, $\phi \wedge \phi'$ *is satisfiable wrt constraint* γ *iff there exist* $\psi \in \Psi$ *and* $\psi' \in \Psi'$ *s.t.* $\psi \wedge \psi'$ *is satisfiable wrt constraint* γ*;*
- *When* $\phi = \phi_0 \wedge \bigwedge_{a\in\mathcal{B}}(K_a\phi_a \wedge L_a\Psi_a)$ *and* $\phi' = \phi'_0 \wedge \bigwedge_{a\in\mathcal{B}'}(K_a\phi'_a \wedge L_a\Psi'_a)$, *where* $\mathcal{B}, \mathcal{B}' \subseteq \mathcal{A}$, $\phi \wedge \phi'$ *is satisfiable wrt constraint* γ *iff the following hold:*

– ϕ, ϕ' *and* $\phi_0 \wedge \phi'_0$ *are satisfiable wrt constraint* γ*;*
– *For each* $a \in \mathcal{B} \cap \mathcal{B}'$, $\phi_a \wedge \phi'_a$ *is satisfiable wrt* γ*;*
– *For each* $a \in \mathcal{B} \cap \mathcal{B}'$, *for each* $\psi \in \Psi_a$, $\phi_a \wedge \phi'_a \wedge \psi$ *is satisfiable wrt* γ*;*
– *For each* $a \in \mathcal{B} \cap \mathcal{B}'$, *for each* $\psi' \in \Psi'_a$, $\phi_a \wedge \phi'_a \wedge \psi'$ *is satisfiable wrt* γ.

In Example 1, $\neg\mathcal{G} = L_a(muddy(a)) \wedge L_a(\neg muddy(a))$, thus $\mathcal{I} \wedge \neg\mathcal{G} = L_a(muddy(a)) \wedge L_a(\neg muddy(a))$, which is satisfiable wrt γ, i.e., $\mathcal{I} \not\models_\gamma \mathcal{G}$.

3.2 Strong Entailment and Strong Equivalence

Our planner searches for solution through the space of KBs, and performs cycle checking during search. Thus, we need an efficient algorithm for checking equivalence relation. In last section, we have defined the algorithm for reasoning. However, except for formulas that can be transformed into the negation ADNFs during preprocessing, it's not tolerable to transform a random formula because there may be a huge increase in the formula size according to the distributive law. In this section, we will introduce a stronger notion of entailment and equivalence.

Definition 15. *Let* ϕ *and* ϕ' *be two ADNFs and* γ *be a constraint.* ϕ *strongly entail* ϕ' *wrt* γ, *written* $\phi \mapsto_\gamma \phi'$, *is defined recursively:*

- *When* ϕ *and* ϕ' *are propositional formulas,* $\phi \mapsto_\gamma \phi'$ *if* $\phi \wedge \gamma \models \phi'$;
- *When* $\phi = \bigvee \Psi$ *and* $\phi' = \bigvee \Psi'$, $\phi \mapsto_\gamma \phi'$ *if* $\forall \psi \in \Psi, \exists \psi' \in \Psi'$ *s.t.* $\psi \mapsto_\gamma \psi'$;
- *When* $\phi = \phi_0 \wedge \bigwedge_{a\in\mathcal{B}}(K_a\phi_a \wedge L_a\Psi_a)$ *and* $\phi' = \phi'_0 \wedge \bigwedge_{a\in\mathcal{B}'}(K_a\phi'_a \wedge L_a\Psi'_a)$, *where* $\mathcal{B}, \mathcal{B}' \subseteq \mathcal{A}$. $\phi \mapsto_\gamma \phi'$ *if the following hold:*
  - $\mathcal{B}' \subseteq \mathcal{B}$ *and* $\phi_0 \mapsto_\gamma \phi'_0$;
  - *For each* $a \in \mathcal{B}'$, $\phi_a \mapsto_\gamma \phi'_a$;
  - *For each* $a \in \mathcal{B}'$, *for all* $\psi' \in \Psi'_a$ *there exists* $\psi \in \Psi_a$ *s.t.* $\phi_a \wedge \psi \mapsto_\gamma \psi'$.

By Proposition 2 we can easily get that $\phi \mapsto_\gamma \phi'$ implies $\phi \models_\gamma \phi'$. Moreover, we say that two ADNFs ϕ and ϕ' are strongly equivalent under γ, written $\phi \leftrightarrow_\gamma \phi'$, if both $\phi \mapsto_\gamma \phi'$ and $\phi' \mapsto_\gamma \phi$. For example, assume that $\phi = K_a(p) \wedge L_a(p \wedge q), \phi' = K_a(p \vee q) \wedge L_a(p \wedge q) \wedge L_a(p), \gamma = \top$, so $\phi \mapsto_\gamma \phi'$ since $p \wedge \top \models p \vee q, p \wedge q \wedge \top \models p$, but $\phi' \not\mapsto_\gamma \phi$ since $(p \vee q) \wedge \top \not\models p$. Thus, $\phi \not\leftrightarrow_\gamma \phi'$.

3.3 Revision and Update

In this section, we will introduce the revision and update algorithms. The basic idea is that we revise or update a higher-order epistemic formula by reducing it to that of lower-order, applying the revision or update recursively, and finally resorting it to the progression of propositional formulas as basis. We will also show that our algorithm satisfies the basic corollaries mentioned above. We begin with some notions. Let Φ and Φ' be two sets of formulas and γ be a constraint:

- $\Phi *_\gamma \Phi' =$
  - $\{(\phi, \phi') | \phi \in \Phi, \phi' \in \Phi', \phi \wedge \phi'$ is satisfiable wrt $\gamma\}$, if there exist $\phi \in \Phi$ and $\phi' \in \Phi'$ s.t. $\phi \wedge \phi'$ is satisfiable wrt γ;
  - $\{(\phi, \phi') | \phi \in \Phi, \phi' \in \Phi'\}$, otherwise.
- $\max(\Phi) = \Phi - \{\phi \mid \phi \in \Phi$, and there exists $\phi' \in \Phi$ s.t. $\phi' \models_\gamma \phi\}$.

Intuitively, $\Phi * \Phi'$ means that we will always restrict our attention to those pairs that are consistent whenever possible. $\max(\Phi)$ is the set of the maximal elements of Φ, i.e., elements which cannot be entailed by other elements of Φ. For example, it is reasonable to store the formula $\phi = L_a\{p \wedge q\}$ rather than $\phi' = L_a\{p, p \wedge q\}$, since $\phi \mapsto_\gamma \phi'$ (assume that $\gamma = \top$), i.e., ϕ is strictly stronger than ϕ' wrt γ.

Definition 16. *Let* ϕ *and* ϕ' *be two ADNFs and* γ *be a constraint. The revision of* ϕ *with* ϕ' *under* γ, *written* $\phi \circ_\gamma \phi'$, *is defined recursively:*

1. *When* ϕ *and* ϕ' *are propositional formulas,* $\phi \circ_\gamma \phi' = \phi \circ_s (\phi' \wedge \gamma)$, *where* $\circ_s$ *is the Satoh's revision operator [11];*
2. *When* $\phi = \bigvee \Psi$ *and* $\phi' = \bigvee \Psi'$, $\phi \circ_\gamma \phi' = \bigvee\{\psi \circ_\gamma \psi' | (\psi, \psi') \in \Psi *_\gamma \Psi'\}$;
3. *When* $\phi = \phi_0 \wedge \bigwedge_{a\in\mathcal{B}}(K_a\phi_a \wedge L_a\Psi_a)$ *and* $\phi' = \phi'_0 \wedge \bigwedge_{a\in\mathcal{B}'}(K_a\phi'_a \wedge L_a\Psi'_a)$ *where* $\mathcal{B}, \mathcal{B}' \subseteq \mathcal{A}$, *and* $\phi \wedge \phi'$ *is satisfiable wrt* γ, $\phi \circ_\gamma \phi' = \phi \wedge \phi'$;

4. *When* $\phi = \phi_0 \wedge \bigwedge_{a\in\mathcal{B}}(K_a\phi_a \wedge L_a\Psi_a)$ *and* $\phi' = \phi'_0 \wedge \bigwedge_{a\in\mathcal{B}'}(K_a\phi'_a \wedge L_a\Psi'_a)$ *where* $\mathcal{B}, \mathcal{B}' \subseteq \mathcal{A}$, *and* $\phi \wedge \phi'$ *is unsatisfiable wrt* γ,

$$
\begin{aligned}
\phi \circ_\gamma \phi' \quad &= \quad \phi_0 \circ_\gamma \phi'_0 && (1)\\
&\wedge \quad \bigwedge\nolimits_{a\in\mathcal{B}-\mathcal{B}'}(K_a\phi_a \wedge L_a\Psi_a) \wedge \bigwedge\nolimits_{a\in\mathcal{B}'-\mathcal{B}}(K_a\phi'_a \wedge L_a\Psi'_a) && (2)\\
&\wedge \quad \bigwedge\nolimits_{a\in\mathcal{B}\cap\mathcal{B}'} K_a[(\phi_a \circ_\gamma \phi'_a) \vee \bigvee\nolimits_{\psi'_a\in\Psi'_a}(\phi_a \circ_\gamma (\phi'_a \wedge \psi'_a))] && (3)\\
&\wedge \quad \bigwedge\nolimits_{a\in\mathcal{B}\cap\mathcal{B}'} L_a[\max(\Psi'_a \cup \{\psi \circ_\gamma \psi' | (\psi, \psi') \in \Psi_a *_\gamma \{\phi_K\}\})] && (4)
\end{aligned}
$$

We use $\phi_K = (\phi_a \circ_\gamma \phi'_a) \vee \bigvee_{\psi'_a\in\Psi'_a}(\phi_a \circ_\gamma (\phi'_a \wedge \psi'_a))$ to denote the result knowledge. Intuitively, Rule 1 is for propositional formulas, Rule 2 and 3 are for the purpose of the Satisfiability Property of revision. In Rule 4, (3) means that we revise the knowledge with the new knowledge, and then to avoid contradiction, we also revise it with each $\phi'_a \wedge \psi'_a$. (4) denotes that we keep the old possibilities consistent with ϕ_K if possible, otherwise revise each one with ϕ_K. Among the new possibilities, we remove the weaker one by maximizing.

Next, we will introduce a stronger satisfiability notion, called disjunctive-wise satisfiability, of ADNFs before introducing the property of our revision operator. The motivation is that, firstly, remove the unsatisfiable terms in the disjunctive formula, i.e., $\phi = p$ rather than $\phi = p \vee \bot$; secondly, remove the unsatisfiable possibilities, i.e., $\phi = L_a\{p\}$ rather than $\phi = L_a\{p, \bot\}$.

Definition 17. *Let* ϕ *be a ADNF, and* γ *be a constraint. We say* ϕ *is disjunctive-wise satisfiable, in short d-satisfiable, wrt* γ *if one of the following holds:*

- ϕ *is a propositional term and* $\phi \wedge \gamma$ *is propositional satisfiable;*
- $\phi = \phi_0 \wedge \bigwedge_{a\in\mathcal{A}}(K_a\phi_a \wedge L_a\Psi_a)$, ϕ *is satisfiable wrt* γ, *and for all* $a \in \mathcal{A}$, ϕ_a *is d-satisfiable wrt* γ *and for all* $\psi_a \in \Psi_a$, ψ_a *is d-satisfiable wrt* γ*;*
- $\phi = \bigvee \Psi$, *and for all* $\psi \in \Psi, \psi$ *is d-satisfiable wrt* γ.

The following proposition shows some properties of our revision operator.

Proposition 6. *Let* ϕ *and* ϕ' *be two ADNFs and* γ *be a constraint. If* $\phi \wedge \phi'$ *is satisfiable,* $\phi \circ_\gamma \phi' \Leftrightarrow_\gamma \phi \wedge \phi'$. *Moreover, if both* ϕ *and* ϕ' *are d-satisfiable wrt* γ, *then* $\phi \circ_\gamma \phi' \models_\gamma \phi'$ *and* $\phi \circ_\gamma \phi'$ *is d-satisfiable wrt* γ.

In Example 1, assume that $a_1 = sense(a, b), a_2 = ask(a, b)$, then the positive result after applying a_1 is $\phi^+ = (L_a(muddy(a)) \wedge L_a(\neg muddy(a))) \circ_\gamma K_a(muddy(b)) = K_a(muddy(b)) \wedge L_a(muddy(a)) \wedge L_a(\neg muddy(a))$ since $\mathcal{I} \wedge pos(a_1)$ is satisfiable wrt γ. And the next positive result after applying a_2 is $\phi_1^+ = \phi^+ \circ_\gamma (K_a((muddy(a) \wedge \neg muddy(b) \wedge muddy(c)) \vee (\neg muddy(a) \wedge muddy(b) \wedge \neg muddy(c)))) = K_a(\neg muddy(a) \wedge muddy(b) \wedge \neg muddy(c)) \wedge L_a(\neg muddy(a) \wedge muddy(b) \wedge \neg muddy(c))$, satisfying $\phi_1^+ \models_\gamma \mathcal{G}$. Next we move to update.

Definition 18. *Let ϕ and ϕ' be two ADNFs and γ be a constraint. The update of ϕ with ϕ' under γ, written $\phi \diamond_\gamma \phi'$, is defined recursively:*

1. *When ϕ and ϕ' are propositional formulas, $\phi \diamond_\gamma \phi' = \phi \diamond_w (\phi' \wedge \gamma)$, where $\diamond_w$ is the Winslett's update operator [12];*
2. *When $\phi = \bigvee \Psi$, $\phi \diamond_\gamma \phi' = \bigvee_{\psi \in \Psi} (\psi \diamond_\gamma \phi')$;*
3. *When $\phi' = \bigvee \Psi'$, $\phi \diamond_\gamma \phi' = \bigvee \{\psi \diamond_\gamma \psi' | (\psi, \psi') \in \{\phi\} * \Psi'\}$;*
4. *When $\phi = \phi_0 \wedge \bigwedge_{a \in \mathcal{B}} (K_a \phi_a \wedge L_a \Psi_a)$, $\phi' = \phi'_0 \wedge \bigwedge_{a \in \mathcal{B}'} (K_a \phi'_a \wedge L_a \Psi'_a)$ where $\mathcal{B}, \mathcal{B}' \subseteq \mathcal{A}$,*

$$
\begin{aligned}
\phi \diamond_\gamma \phi' \quad &= \quad \phi_0 \diamond_\gamma \phi'_0 && (1)\\
&\wedge \quad \bigwedge\nolimits_{a \in \mathcal{B} - \mathcal{B}'} (K_a \phi_a \wedge L_a \Psi_a) \wedge \bigwedge\nolimits_{a \in \mathcal{B}' - \mathcal{B}} (K_a \phi'_a \wedge L_a \Psi'_a) && (2)\\
&\wedge \quad \bigwedge\nolimits_{a \in \mathcal{B} \cap \mathcal{B}'} K_a [(\phi_a \diamond_\gamma \phi'_a) \vee \bigvee\nolimits_{\psi'_a \in \Psi'_a} (\phi_a \diamond_\gamma (\phi'_a \wedge \psi'_a))] && (3)\\
&\wedge \quad \bigwedge\nolimits_{a \in \mathcal{B} \cap \mathcal{B}'} L_a [\max (\Psi'_a \cup \{\psi_a \diamond_\gamma \phi_K | \psi_a \in \Psi_a\})] && (4)
\end{aligned}
$$

Similar, $\phi_K = (\phi_a \diamond_\gamma \phi'_a) \vee \bigvee_{\psi'_a \in \Psi'_a} (\phi_a \diamond_\gamma (\phi'_a \wedge \psi'_a))$ denotes the result knowledge. Intuitively, Rule 1 is for propositional formulas and Rule 2 for the distribution property of the update operator. Rule 3 denotes that we concern the consistent formulas whenever possible and Rule 4 is similar to that in revision operator.

Proposition 7. *Let ϕ_1, ϕ_2, ϕ and ϕ' be the ADNFs and γ be a constraint. Then $(\phi_1 \vee \phi_2) \diamond_\gamma \phi' \Leftrightarrow_\gamma (\phi_1 \diamond_\gamma \phi' \vee \phi_2 \diamond_\gamma \phi')$. Moreover, if both ϕ and ϕ' are d-satisfiable wrt γ, then $\phi \diamond_\gamma \phi' \models_\gamma \phi'$ and $\phi \diamond_\gamma \phi'$ is d-satisfiable wrt γ.*

Since there is no deterministic action in Example 1, we use another example to illustrate the algorithm. Assume that $\phi = K_a(p), \phi' = L_a(\neg p), \gamma = \top$, $\phi \diamond_\gamma \phi' = K_a((p \diamond_\gamma \top) \vee (p \diamond_\gamma \neg p)) \wedge L_a(\neg p) = K_a(p \vee \neg p) \wedge L_a(\neg p) = L_a(\neg p)$.

4 Planning

Currently, heuristic search has been widely used for planning. Recall that our planner searches for solution through the space of KBs, thus we also apply heuristic search for acceleration. In this section, we will introduce the heuristic function, the main search algorithm, and finally two heuristic pruning strategies.

4.1 Heuristic Function

Definition 19. *Let ϕ and ϕ' be two ADNFs. The distance from ϕ to ϕ', written $dist(\phi, \phi')$, is defined recursively:*

1. *When ϕ and ϕ' are propositional terms, $dist(\phi, \phi')$ is the number of literals of ϕ' which are not entailed by ϕ;*
2. *When $\phi = \bigvee \Psi, \phi' = \bigvee \Psi'$, $dist(\phi, \phi') = \min(\{dist(\psi, \psi') \mid \psi \in \Psi, \psi' \in \Psi'\})$;*

3. *When* $\phi = \phi_0 \wedge \bigwedge_{a\in\mathcal{B}}(K_a\phi_a \wedge L_a\Psi_a)$, $\phi' = \phi'_0 \wedge \bigwedge_{a\in\mathcal{B}'}(K_a\phi'_a \wedge L_a\Psi'_a)$,

$$
\begin{aligned}
dist(\phi,\phi') \quad &= \quad dist(\phi_0,\phi'_0) && (1)\\
&+ \quad \sum\nolimits_{a\in\mathcal{B}'-\mathcal{B}} dist(\top,\phi'_a) + \sum\nolimits_{a\in\mathcal{B}'-\mathcal{B}} \sum\nolimits_{\psi'_a\in\Psi'_a} dist(\top,\psi'_a) && (2)\\
&+ \quad \sum\nolimits_{a\in\mathcal{B}'\cap\mathcal{B}} dist(\phi_a,\phi'_a) && (3)\\
&+ \quad \sum\nolimits_{a\in\mathcal{B}'\cap\mathcal{B}} \sum\nolimits_{\psi'_a\in\Psi'_a} \min(\{dist(\psi_a,\psi'_a)|\psi_a\in\Psi_a\}) && (4)
\end{aligned}
$$

Note that we concern the distance from ϕ to ϕ' but not the other direction. The basic idea is by reductively computing distance in an abstract version, which contains at most one agent and is of lower-order, of the original problem, which involves multi-agents and is of higher-order. Intuitively, in Rule 3, for each $a \in \mathcal{B}'$, we sum up the distance between ϕ_a ($\top$ if $a \notin \mathcal{B}$) and ϕ'_a, and for each $\psi'_a \in \Psi'_a$, sum up the minimal distance between each $\psi_a \in \Psi_a$ ($\top$ if $a \notin \mathcal{B}$) and ψ'_a.

Definition 20. *Let ϕ be an ADNF and $\mathcal{G}$ be the goal. The heuristic value of ϕ, written $h(\phi)$, is defined as: $h(\phi) = dist(\phi, \mathcal{G})$.*

4.2 AO* with Cycle Checking

AO* and LAO* [13] are two heuristic search algorithms for AND/OR graph with the difference that LAO* can handle graphs with loops and find a solution with loops while AO* cannot. Recall that the solution for a MEP problem is an action tree without loops, so we adapt AO* as our main search algorithm which is much more efficient. However, there may be loops throughout the search space preventing AO* from halting. Therefore, we combine AO* algorithm with Cycle Checking for contingent planning. Because we represent KBs with ADNFs, we apply the satisfiability algorithm for reasoning, strong equivalence algorithm for cycle checking, revision and update algorithms for progression wrt sensing and deterministic actions, respectively. Besides, note that our planner will perform cycle checking while extracting policies from AND/OR graph to make sure that they contain no loops. And finally, the algorithm returns the action tree extracting from the solution policy.

4.3 Heuristic Pruning Strategy

In this section, we will introduce two heuristic pruning strategies.

1. **Deadend Propagation:** Deadend propagation aims to handle the dead end during searching. We say a state is a deadend if $isDeadend(s)$ returns true, where $isDeadend(s)$ returns true if one of the following holds: (1) $s \not\models_\gamma \mathcal{G}$ and $\forall a \in \mathcal{S} \cup \mathcal{D}, s \not\models_\gamma pre(a)$; (2) $s \not\models_\gamma \mathcal{G}$, $\forall a \in \mathcal{D}, isDeadend(prog(s,a))$ returns true and $\forall a \in \mathcal{S}$, $isDeadend(s^+)$ and $isDeadend(s^-)$ return true, where $prog(s,a) = \langle s^+, s^-\rangle$. When we encounter a deadend s, we will label it by assigning infinite to its state cost and apply *deadendPropagation* function recursively:

- If there exists an action $a \in \mathcal{D}$ and a state s' s.t $prog(s', a) \leftrightarrow_\gamma s$, perform $deadendPropagation(s')$ recursively;
- If there exists an action $a \in \mathcal{S}$ and a state s' s.t $s^+ \leftrightarrow_\gamma s$ or $s^- \leftrightarrow_\gamma s$, where $prog(s', a) = \langle s^+, s^- \rangle$, perform $deadendPropagation(s')$ recursively.

2. **Common Sub-Formula:** This strategy aims to enhance our heuristic function by distinguishing the formulas inconsistent with the common part of the initial state and the goal. Let ϕ and ϕ' be two ADNFs. The Common Sub-Formula of ϕ and ϕ', written $csf(\phi, \phi')$, is an ADNF defined as followings:
   - When ϕ and ϕ' are propositional terms, $csf(\phi, \phi')$ is the conjunction of the common literals of ϕ and ϕ';
   - When $\phi = \bigvee\{\psi_1, \psi_2, \ldots, \psi_n\}$, $csf(\phi, \phi') = csf(\psi_1, csf(\ldots, csf(\psi_n, \phi')))$;
   - When $\phi' = \bigvee\{\psi'_1, \psi'_2, \ldots, \psi'_n\}$, $csf(\phi, \phi') = csf(\phi', \phi)$;
   - When $\phi = \phi_0 \wedge \bigwedge_{a \in \mathcal{B}}(K_a\phi_a \wedge L_a\Psi_a)$ and $\phi' = \phi'_0 \wedge \bigwedge_{a \in \mathcal{B}'}(K_a\phi'_a \wedge L_a\Psi'_a)$, where $\mathcal{B}, \mathcal{B}' \subseteq \mathcal{A}$, $csf(\phi, \phi') = csf(\phi_0, \phi'_0) \wedge \bigwedge_{a \in \mathcal{B} \cap \mathcal{B}'}[K_a(csf(\phi_a, \phi'_a)) \wedge L_a(\{csf(\psi_a, \psi'_a) \mid \psi_a \in \Psi_a, \psi'_a \in \Psi'_a\})]$.

Our planner will first calculate the common sub-formula of the initial state and the goal, that is $csf(\mathcal{I}, \mathcal{G})$, and for every new formula ϕ generated during searching, we will increase its heuristic value if $\phi \wedge csf(\mathcal{I}, \mathcal{G})$ is not satisfiable wrt γ. The intuition for this pruning strategy is that $csf(\mathcal{I}, \mathcal{G})$ denotes the part in $\mathcal{I}$ consistent with $\mathcal{G}$, therefore we think that a new formula inconsistent with $csf(\mathcal{I}, \mathcal{G})$ may have higher probability of misleading the search.

5 Implementation and Experiments

Based on the theoretic work, we have implemented a Multi-Agent Epistemic Planner called MEPL. MEPL takes as input a file in EPDDL, an extension of PDDL [14] for describing MEP problems, and outputs a solution tree if it exists and the search statistics. We evaluate MEPL with all the domains from MEPK [5], including Hexa Game, Collaboration-and-Communication (CC), Gossip, Grapevine, Selective-Communication (SC) and Assembly-Line (AL). In domain Collaboration-and-Communication, CC(n, m) and *CC(n, m) denote n the number of boxes and m the number of rooms. Specially, CC(n, m) and *CC(n, m) vary from each other slightly in a way that the encoding of the sensing action is different. In domain Grapevine, Grapevine (n) denotes n the number of rooms. In domain Selective-Communication, SC (n) denotes n the number of rooms. Besides, inspired by Kominis and Geffner [3], we make up a new domain Simple Muddy Children (SMC) by removing the public announcement action. Note that Example 1 is an instance of the SMC domain. The domain is described as:

There are n children, and at least 1 and at most $n - 1$ children are muddy in their heads. A child a_1 can sense whether another child a_2 is muddy and can also ask the child a_2 whether a_2 can identify herself. Assume that there is a child

a_1, initially the child a_1 doesn't know she is muddy or she isn't, and the goal is that the child a_1 gets to know she is muddy or she isn't.

Our experiments were run on a Linux machine with 2.50 GHz CPU and 4 GB RAM, and Table 1 shows the experimental statistics, each of which is the optimal one among ten experiments. Note that the first column in the table is the name of

Table 1. Experimental results

Problem	$\|\mathcal{A}\|$	$\|\mathcal{S}\| + \|\mathcal{D}\|$	d	MEPK	MEPL
Hexa Game	3	$18+0$	1	$0.00-0.00(1/1)$	$\mathbf{0.00-0.00(1/1)}$
	4	$48+0$	1	$0.02-0.02(3/6)$	$\mathbf{0.02-0.02(4/5)}$
	5	$100+0$	1	$28.08-24.64(6/185)$	$\mathbf{1.14-1.13(7/23)}$
	6	$180+0$	1	N/A	$\mathbf{173.31-173.20(11/119)}$
CC(2,4)	2	$0+18$	1	$0.14-0.13(3/4)$	$\mathbf{0.09-0.08(3/3)}$
CC(3,4)	2	$0+18$	1	$1.87-1.85(3/4)$	$\mathbf{1.01-1.00(3/3)}$
CC(4,4)	2	$0+18$	1	$81.31-81.24(3/4)$	$\mathbf{34.57-34.55(3/3)}$
*CC(2,3)	2	$12+42$	1	$11.70-11.69(5/312)$	$\mathbf{1.31-1.30(5/37)}$
*CC(2,4)	2	$12+20$	1	$8.68-8.67(6/300)$	$\mathbf{0.78-0.78(7/28)}$
*CC(3,3)	3	$18+60$	1	$10.12-10.09(3/50)$	$\mathbf{2.86-2.85(5/23)}$
*CC(3,4)	3	$27+78$	1	N/A	$\mathbf{32.76-32.75(9/29)}$
Gossip	3	$0+6$	2	$0.04-0.04(3/5)$	$\mathbf{0.01-0.01(3/3)}$
	4	$0+24$	2	$6.20-5.58(5/47)$	$\mathbf{0.34-0.33(5/5)}$
	5	$0+120$	2	N/A	$\mathbf{8.38-8.35(7/7)}$
	6	$0+720$	2	N/A	$\mathbf{213.88-213.45(9/9)}$
Grapevine(2)	3	$0+18$	2	$0.02-0.01(2/7)$	$\mathbf{0.01-0.01(2/4)}$
Grapevine(2)	4	$0+56$	1	$0.09-0.05(2/8)$	$\mathbf{0.06-0.05(2/5)}$
Grapevine(2)	4	$0+56$	2	$0.11-0.07(2/8)$	$\mathbf{0.06-0.04(2/4)}$
Grapevine(2)	4	$0+56$	3	$0.17-0.11(2/8)$	$\mathbf{0.08-0.05(2/4)}$
Grapevine(2)	4	$0+56$	4	$0.30-0.20(2/8)$	$\mathbf{0.10-0.07(2/4)}$
Grapevine(3)	4	$0+152$	1	$1.10-0.71(2/9)$	$\mathbf{0.20-0.13(2/5)}$
Grapevine(3)	5	$0+730$	1	N/A	$\mathbf{7.45-7.31(2/20)}$
SC(4)	3	$1+3$	1	$\mathbf{0.04-0.03(5/23)}$	$0.06-0.06(5/20)$
SC(4)	7	$1+3$	1	$13.72-0.05(5/23)$	$\mathbf{0.28-0.28(5/20)}$
SC(4)	8	$1+3$	1	$110.48-0.10(5/39)$	$\mathbf{0.60-0.59(5/26)}$
SC(4)	3	$1+3$	3	$0.04-0.03(5/23)$	$\mathbf{0.04-0.04(5/14)}$
SC(4)	3	$1+3$	4	$0.07-0.05(5/23)$	$\mathbf{0.05-0.05(5/14)}$
SC(8)	3	$1+3$	1	$\mathbf{0.21-0.14(10/41)}$	$0.36-0.35(10/30)$
AL	2	$4+2$	2	$0.02-0.01(5/25)$	$\mathbf{0.02-0.02(5/16)}$
	2	$4+2$	3	$0.03-0.03(5/25)$	$\mathbf{0.03-0.03(5/16)}$
	2	$4+2$	4	$0.03-0.03(5/25)$	$\mathbf{0.03-0.03(5/16)}$
	2	$4+2$	5	$0.05-0.04(5/25)$	$\mathbf{0.04-0.04(5/16)}$
	2	$4+2$	7	$0.13-0.12(5/25)$	$\mathbf{0.07-0.07(5/16)}$
	2	$4+2$	10	$1.02-0.96(5/25)$	$\mathbf{0.15-0.14(5/16)}$
SMC	3	$12+0$	1	$0.10-0.10(3/24)$	$\mathbf{0.01-0.01(3/5)}$
	4	$36+0$	1	$406.95-406.94(5/706)$	$\mathbf{0.42-0.40(5/13)}$
	5	$140+0$	1	N/A	$\mathbf{1.59-1.55(5/13)}$
	6	$750+0$	1	N/A	$\mathbf{6.57-6.37(5/13)}$

the domain, and the 2nd-4th columns indicate the number of agents, the number of sensing actions and deterministic actions, and the modal depth, respectively. In the last two columns, $A-B(X/Y)$ denotes A seconds of total time, B seconds of search time, X depth of the solution tree, and Y nodes explored during search. N/A denotes that the instance is unsolvable within the allotted time (1000 s).

The experimental results show that our planner outperforms MEPK in three aspects. Firstly, from the aspect of the total time, MEPL performs much better than MEPK for most instances, especially when the size of the instances grows. Also, MEPL can solve some instances that MEPK cannot within the allocated time. It's reasonable due to the more space efficient normal forms and the more efficient reasoning and progression algorithm. Secondly, for all instances, the number of nodes explored by MEPL is less than that explored by MEPK, which shows the viability of our heuristic function and pruning strategies. Finally, the preprocessing of MEPL is much faster than that of MEPK in a way that MEPL can complete the preprocessing within 1 s while MEPK may take more than 100 s in some domains like SC. It's reasonable too since MEPL doesn't need to transform a random formula into an ACDF. In conclusion, MEPL outperforms MEPK in most instances, and solves some instances that MEPK cannot solve, which means that MEPL can handle larger instances, i.e., scales better.

6 Conclusions

In this paper, we have proposed two improvements, which are based on higher-order belief change and heuristic search, for the multi-agent epistemic planner MEPK. Firstly, we exploited alternating disjunctive normal form (ADNF) for multi-agent KD45, which is more space efficient than the normal form used by MEPK and also saves time in compilation, and proposed efficient reasoning, revision, and update algorithms for it. Secondly, we proposed a heuristic function for multi-agent epistemic planning based on the distance between ADNFs, and as the planning algorithm, we applied heuristic research AO* with cycle checking and two heuristic pruning strategies. We have implemented a multi-agent epistemic planner MEPL, which outperformed MEPK in most planning instances, and solved a number of instances that MEPK cannot solve. Like MEPK, MEPL can handle propositional common knowledge, called constraints. Nonetheless, MEPL does not handle complex constraints well. In the future, we are interested in overcoming this limitation and further dealing with general common knowledge.

Acknowledgement. We acknowledge support from the Natural Science Foundation of China under Grant Nos. 61572535.

References

1. Bolander, T., Andersen, M.B.: Epistemic planning for single and multi-agent systems. J. Appl. Non-Class. Log. **21**(1), 9–34 (2011)
2. Van Ditmarsch, H., van Der Hoek, W., Kooi, B.: Dynamic Epistemic Logic. Springer, Heidelberg (2007). https://doi.org/10.1007/978-1-4020-5839-4
3. Kominis, F., Geffner, H.: Beliefs in multiagent planning: from one agent to many. In: Proceedings of the 25th ICAPS, 7–11 June 2015, Jerusalem, Israel, pp. 147–155 (2015)
4. Muise, C.J., et al.: Planning over multi-agent epistemic states: a classical planning approach. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, 25–30 January 2015, Austin, Texas, USA, pp. 3327–3334 (2015)
5. Huang, X., Fang, B., Wan, H., Liu, Y.: 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-17, pp. 1093–1101 (2017)
6. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.: Reasoning About Knowledge. MIT Press, Cambridge (1995)
7. Hales, J., French, T., Davies, R.: Refinement quantified logics of knowledge and belief for multiple agentsc. In: Advances in Modal Logic 9, Papers From the Ninth Conference on "Advances in Modal Logic," 22–25 August 2012, Copenhagen, Denmark, pp. 317–338 (2012)
8. Alchourrón, C.E., Gärdenfors, P., Makinson, D.: On the logic of theory change: partial meet contraction and revision functions. J. Symb. Log. **50**(2), 510–530 (1985)
9. Katsuno, H., Mendelzon, A.O.: On the difference between updating a knowledge base and revising it. In: Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning, Cambridge, MA, USA, pp. 387–394 (1991)
10. Darwiche, A., Pearl, J.: On the logic of iterated belief revision. Artif. Intell. **89**(1–2), 1–29 (1997)
11. Satoh, K.: Nonmonotonic reasoning by minimal belief revision. In: FGCS, pp. 455–462 (1988)
12. Winslett, M.: Reasoning about action using a possible models approach. In: Proceedings of the 7th National Conference on Artificial Intelligence, 21–26 August 1988, St. Paul, MN, pp. 89–93 (1988)
13. Hansen, E.A., Zilberstein, S.: Lao*: a heuristic search algorithm that finds solutions with loops. Artif. Intell. **129**(1–2), 35–62 (2001)
14. McDermott, D., et al.: PDDL-the planning domain definition language. Technical report CVC TR98003/DCS TR1165, Yale Center for Computational Vision and Control, New Haven, CT (1998)