An Improved Multi-agent Epistemic Planner via Higher-order Belief Change based on Heuristic Search

No Author Given

No Institute Given

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 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.

Keywords: Modal logics, 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 about high-order knowledge and belief and their change is also needed in some cases. For example, an agent may wish that she knows an 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, Kominis and Geffner [3] and Muise *et al.* [4] showed 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.

Very recently, Huang *et al.* [5] proposed a general representation framework for multi-agent epistemic planning, where the initial KB and the goal, the preconditions and effects of actions can be arbitrary multi-agent epistemic formulas, progression of KBs wrt actions is achieved through the operation of belief revision or update, 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 logic called alternating cover disjunctive formulas (ACDF-s). Using forward state space search, they implemented a multi-agent epistemic planner MEPK. However, MEPK does not scale well.

In the field of planning, heuristic search has shown large potential and has been widely applied. Bonnet and Geffner [6] proposed the delete relaxation heuristic for classical planning, which was further developed in the FF planning system [7]. Besides, Mattmüller *et al.* [8] explored pattern database heuristics for fully observable non-deterministic (FOND) planning, and applied the heuristic search algorithm LAO* [9]. Winterer *et al.* [10] generalized structural symmetries and symmetry reduction to FOND planning with LAO*.

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 research AO* with cycle checking and two heuristic pruning strategies. Finally, based on our theoretic work, we 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 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 [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 $M,w \models \psi$; and ϕ and ψ are equivalent, written $\phi \Leftrightarrow \psi$, if $\phi \models \psi$ and $\psi \models \phi$. Similarly, we say ϕ is satisfiable under 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 constraint γ, written $\phi \models_\gamma \psi$, if for all model (M,w), $M,w \models \phi \wedge \gamma \wedge C\gamma$ implies $M,w \models \psi \wedge \gamma \wedge C\gamma$; and ϕ and ψ is equivalent under constraint γ, written $\phi \Leftrightarrow_\gamma \psi$, if $\phi \models_\gamma \psi$ and $\psi \models_\gamma \phi$.

Now 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, $K_a(K_b(p))$ 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 is a $\psi_a \in \Psi_a$ 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 in [11], belief update in [12] and iterated belief revision in [13]. We use $\circ$ to denote revision operator and $\diamond$ to denote update operator. [12] has 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 from [5].

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.*

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 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.*

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 $effs(a_d) = \{\langle con_1, cef_1 \rangle, ..., \langle con_n, cef_n \rangle\}$. Let $\{\langle c_1, e_1 \rangle, ..., \langle c_k, e_k \rangle\} \subseteq eff(a_d)$ s.t. $\phi \models_\gamma c_i$, for $1 \leq i \leq k$. The progression of ϕ wrt a_d under 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 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, writ-*

ten $prog(\phi, T)$ 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 $\forall \phi \in prog(\mathcal{I}, T)$, $\phi \models_\gamma \mathcal{G}$.*

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 knowledge 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 $pre(a)$ into negation form. Next, we will introduce the reasoning algorithm for 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.*

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.*

Now 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.

3.2 Strong Entailment and Strong Equivalence

Our planner searches for solution through the space of KBs, and performs cycle checking during searching. Thus, we need an efficient algorithm for checking equivalence relation. In the last section, we have defined the algorithm for knowledge reasoning. However, except for formulas that can be compiled into the negation ADNFs during preprocess, it's not tolerable to transform a new formula because the formula may become too long according to the distributive law. In this section, we will introduce a stronger notion of 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:

Proposition 5. *Let ϕ and ϕ' be two ADNFs, and γ be a constraint. $\phi \mapsto_\gamma \phi'$ implies $\phi \models_\gamma \phi'$.*

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 $\phi' \mapsto_\gamma \phi$.

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 are not entailed by some other elements of Φ. For example, we will store the formula $L_a\{p \wedge q\}$ rather than $L_a\{p, p \wedge q\}$.

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[14];*
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,*

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

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, and Rule 2 and 3 are for the purpose of conjunction property. In Rule 4, (3) means that we revise the knowledge with 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, otherwise revise each one with ϕ_K. Among the new possibilities, we remove the weaker one by maximizing. Next, before introducing the property of revision operator, we first define the notion of disjunctive-wise satisability.

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 shows some easy 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.*

We now 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[15];*
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 $\mathcal{B}, \mathcal{B}' \subseteq \mathcal{A}$,*

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

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 distribution property. 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.*

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

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

Note that we concern the distance from ϕ to ϕ' but not the other direction. Intuitively, in Rule 3, for each $a \in \mathcal{B}'$, we add up the distance between ϕ_a ($\top$ if $a \notin \mathcal{B}$) and ϕ'_a, and for each $\psi'_a \in \Psi'_a$, add 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*[9] 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 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 keeping AO* from halting. In this section, we will introduce the AO* with Cycle Checking algorithm for our planner. Because we represent KBs as 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. We begin with some notions used in our algorithm:

- We define a policy π : KBs $\rightarrow \mathcal{S} \cup \mathcal{D} \cup \{Ter., Undef.\}$ as a mapping from KBs to actions, where Ter. denotes terminate and Undef. denotes undefined;
- $negEntail(s, \mathcal{G})$ returns true iff $s \wedge \neg\mathcal{G}$ is unsatisfiable wrt γ;
- $HeuristicHelper(s)$ returns $h(s)$;
- $ExtractPolicy(\mathcal{I})$ extracts new policy π'. Note that our planner will perform Cycle Checking here to make sure that the policy contains no loops.

$$\pi'(s) := \begin{cases} Ter. & negEntail(s, \mathcal{G}) \text{ returns true}\\ Undef. & s \text{ is unexplored}\\ \arg\min\limits_{a \in A} Cost(a) & \text{otherwise} \end{cases}$$

  $A = \{a' | a' \in \mathcal{S} \cup \mathcal{D}, s \models_\gamma pre(a')\}$, and if $a \in \mathcal{D}, Cost(a) = 1 + f(prog(s, a))$; else $Cost(a) = 2 + 0.5 * f(\phi^+) + 0.5 * f(\phi^-)$, where $prog(s, a) = \langle \phi^+, \phi^- \rangle$.
- $ExtractSol(\pi)$ extracts the solution tree from policy π;
- $isSolved(s)$ returns true if one of the following holds:(1) $negEntail(s, \mathcal{G})$ returns true;(2) $\pi(s) \in \mathcal{D}$ and $isSolved(prog(s, \pi(s)))$ returns true;(3) $\pi(s) \in \mathcal{S}$, $isSolved(\phi^+)$ and $isSolved(\phi^-)$ return true where $prog(s, \pi(s)) = \langle \phi^+, \phi^- \rangle$.

Algorithm 1 shows the main algorithm for our planner.

4.3 Heuristic Pruning Strategy

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

Deadend Propagation: Deadend propagation aims to handle the dead end during searching. We say a state is a deadend if $isDeadend(s)$ returns true,

Algorithm 1 CAO*

```
Input: Initial state $\mathcal{I}$, goal $\mathcal{G}$, constraint $\gamma$, sets of sensing and deterministic actions $\mathcal{S}$ and $\mathcal{D}$
Output: Solution action tree $T$
  Initialize: Initialize the policy $\pi$ contains only $\pi(\mathcal{I}) = Undef$.
  if $negEntail(\mathcal{I}, \mathcal{G})$ returns true then
    return $T$
  end if
  repeat
    1. Choose a state $s$ in $\pi$ s.t $\pi(s) = Undef$.
    2. Explore $s$, and assign state cost for every new state $s'$:
       $f(s') = \begin{cases} 0 & negEntail(s', \mathcal{G}) \text{ returns true} \\ HeuristicHelper(s') & \text{otherwise} \end{cases}$
    3. Add all the new states to $\pi$, if $negEntail(s', \mathcal{G})$ returns true, then $\pi(s') = Ter.$;
    else $\pi(s') = Undef$. And assign a random action for $\pi(s)$.
    4. Apply backtracking algorithm to update state cost in $\pi$:
       $f(s) = \begin{cases} f(s) & \pi(s) \in \{Undef., Ter.\} \\ 1 + f(prog(s, \pi(s))) & \pi(s) \in \mathcal{D} \\ 2 + 0.5 * f(\phi^+) + 0.5 * f(\phi^-) & \pi(s) \in \mathcal{S}, prog(s, \pi(s)) = \langle \phi^+, \phi^- \rangle \end{cases}$
    5. Extract new policy, $\pi' \leftarrow ExtractPolicy(\mathcal{I})$
    6. If $\pi$ equals $\pi'$, do nothing; Else $\pi \leftarrow \pi'$, goto Step 4.
  until $isSolve(\mathcal{I})$ returns true
  $T \leftarrow ExtractSol(\pi)$
  return $T$
```

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 a big value as its state cost and apply $deadendPropagation$ function recursively:

- If there exist an action $a \in \mathcal{D}$ and a state s' s.t $prog(s', a) \leftrightarrow_\gamma s$, perform $deadendPropagation(s')$ recursively.
- If there exist 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.

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 follows:

- When ϕ and ϕ' are propositional terms, $csf(\phi, \phi')$ is the conjunction of the common literals of ϕ and ϕ';
- When $\phi = \bigvee\{\psi_1, \psi_2, ..., \psi_n\}$, $csf(\phi, \phi') = csf(\psi_1, csf(..., csf(\psi_n, \phi')))$;
- When $\phi' = \bigvee\{\psi'_1, \psi'_2, ..., \psi'_n\}$, $csf(\phi, \phi') = csf(csf(...csf(\phi, \psi'_1)..., \psi'_{n-1}), \psi'_n)$;
- 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 $\mathcal{B}, \mathcal{B}' \subseteq \mathcal{A}$, $csf(\phi, \phi') = csf(\phi_0, \phi'_0) \wedge \bigwedge_{a \in \mathcal{B} \cap \mathcal{B}'}[K_a(csf(\phi_a, \phi'_a)) \wedge L_a(\{csf(\psi_a, \psi'_a) | \psi_a \in \Psi_a, \psi'_a \in \Psi'_a\})]$.

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 [16] for describing MEP problems, and outputs a solution tree if it exists and the search statistics. We evaluate MEPL with all the domains from [5], which include Hexa Game, Collaboration-and-Communication(CC), Gossip, Grapevine, Selective-Communication(SC) and Assembly-Line(AL). Besides, inspired by [3], we make up a new domain Simple Muddy Children(SMC) by removing the public announcement action. The domain is as follows:

There are n children and at least one and at most $n-1$ children are muddy in their heads. A child A can sense whether another child B is muddy and can also ask whether B knows if B is muddy. Initially there is a child who doesn't know whether he is muddy and the goal is for him to know if he is muddy.

Our experiments were run on a Linux machine with 2.50GHz CPU and 4GB RAM, and Table 1 shows the experimental statistics. Note that the first column in the table is the name of the domain, and the 2nd-3th columns indicate the number of agents and the modal depth, respectively. In the last two columns, $A - B(X/Y/Z)$ denotes A seconds of total time, B seconds of search time, X is the solution tree depth, Y nodes of solution tree, and Z nodes explored during search. N/A denotes unsolvable within the allotted time.

Problem	$\lvert\mathcal{A}\rvert$	d	MEPK	MEPL
Hexa Game	3	1	0.00-0.00(1/3/1)	**0.00-0.00(1/3/1)**
	4	1	0.02-0.02(3/11/6)	**0.02-0.02(4/11/5)**
	5	1	28.08-24.64(6/47/185)	**1.14-1.13(7/47/23)**
	6	1	N/A	**173.31-173.20(11/239/119)**
CC(2,4)	2	1	0.14-0.13(3/4/4)	**0.09-0.08(3/4/3)**
CC(3,4)	2	1	1.87-1.85(3/4/4)	**1.01-1.00(3/4/3)**
CC(4,4)	2	1	81.31-81.24(3/4/4)	**34.57-34.55(3/4/3)**
*CC(2,3)^1	2	1	**3.49-3.48(4/12/35)**	3.76-3.75(4/9/16)
*CC(2,3)^2	2	1	11.70-11.69(5/16/312)	**1.31-1.30(5/11/37)**
*CC(2,4)	2	1	8.68-8.67(6/18/300)	**0.78-0.78(7/21/28)**
*CC(3,3)	3	1	10.12-10.09(3/15/50)	**2.86-2.85(5/13/23)**
*CC(3,4)	3	1	N/A	**32.76-32.75(9/21/29)**
Gossip	3	2	0.04-0.03(3/4/5)	**0.02-0.01(3/4/3)**
	4	2	4.98-4.35(4/5/47)	**1.20-1.18(4/5/13)**
	5	2	17.46-11.60(4/5/47)	**4.43-4.33(4/5/13)**
Grapevine(2)	3	2	0.02-0.01(2/3/7)	**0.01-0.01(2/3/4)**
Grapevine(2)	4	1	0.09-0.05(2/3/8)	**0.06-0.05(2/3/5)**
Grapevine(2)	4	2	0.11-0.07(2/3/8)	**0.06-0.04(2/3/4)**
Grapevine(2)	4	3	0.17-0.11(2/3/8)	**0.08-0.05(2/3/4)**
Grapevine(2)	4	4	0.30-0.20(2/3/8)	**0.10-0.07(2/3/4)**

Grapevine(3)	4	1	1.10-0.71(2/3/9)	**0.20-0.13(2/3/5)**
Grapevine(3)^1	5	1	N/A	**7.45-7.31(2/3/20)**
Grapevine(3)^2	5	1	N/A	**32.31-32.17(3/4/46)**
SC(4)	3	1	**0.04-0.03(5/10/23)**	0.06-0.06(5/10/20)
SC(4)	7	1	13.72-0.05(5/10/23)	**0.28-0.28(5/10/20)**
SC(4)^1	8	1	110.48-0.10(5/10/39)	**0.60-0.59(5/10/26)**
SC(4)^2	8	1	118.17-0.03(3/6/8)	**0.06-0.04(3/6/4)**
SC(4)	3	3	0.04-0.03(5/10/23)	**0.04-0.04(5/10/14)**
SC(4)	3	4	0.07-0.05(5/10/23)	**0.05-0.05(5/10/14)**
SC(8)	3	1	**0.21-0.14(10/19/41)**	0.36-0.35(10/19/30)
	2	2	0.02-0.01(5/12/25)	**0.02-0.02(5/12/16)**
	2	3	0.03-0.03(5/12/25)	**0.03-0.03(5/12/16)**
AL	2	4	0.03-0.03(5/12/25)	**0.03-0.03(5/12/16)**
	2	5	0.05-0.04(5/12/25)	**0.04-0.04(5/12/16)**
	2	7	0.13-0.12(5/12/25)	**0.07-0.07(5/12/16)**
SMC(3)	3	1	0.10-0.10(3/11/24)	**0.01-0.01(3/11/5)**
SMC(4)	4	1	406.95-406.94(5/27/706)	**0.42-0.40(5/27/13)**
SMC(5)	5	1	N/A	**1.59-1.55(5/27/13)**
SMC(6)	6	1	N/A	**6.57-6.37(5/27/13)**

Table 1: Experimental Results

The experimental results show that our planner outperforms MEPK in three aspects. Firstly, our planner performs much better than MEPK for most instances, especially when the size of the instances grows. The reason is that ADNFs are more space efficient, and our reasoning and progression algorithms are more efficient. Secondly, for all instances, the number of nodes explored by our planner is less than that by MEPK, which shows the viability of our heuristic function and pruning strategies. Finally, the preprocessing of our planner is much faster than that of MEPK: our planner can complete the preprocessing within 1s while MEPK may take more than 100s in some domains like SC. Thus it's possible for our planner to handle larger instances, i.e., our planner scales better.

6 Conclusions

In this paper, we have proposed two improvements for the multi-agent epistemic planner MEPK, which is based on higher-order belief change and forward state space search. 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, as the planning algorithm, we applied heuristic research AO* with cycle checking and two heuristic pruning strategies. We 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.

References

1. Bolander, T., Andersen, M.B.: Epistemic planning for single and multi-agent systems. Journal of Applied Non-Classical Logics **21**(1) (2011) 9–34
2. Van Ditmarsch, H., van Der Hoek, W., Kooi, B.: Dynamic epistemic logic. Springer (2007)
3. Kominis, F., Geffner, H.: Beliefs in multiagent planning: From one agent to many. In: Proceedings of the 25th ICAPS, Jerusalem, Israel, June 7-11, 2015. (2015) 147–155
4. Muise, C.J., Belle, V., Felli, P., McIlraith, S.A., Miller, T., Pearce, A.R., Sonenberg, L.: Planning over multi-agent epistemic states: A classical planning approach. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA. (2015) 3327–3334
5. Xiao Huang, Biqing Fang, H.W.Y.L.: 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. (2017) 1093–1101
6. Bonnet, B., Geffner, H.: Hsp: Heuristic search planner. In AIPS-98 Planning Competition Pittsburgh, PA. (1998)
7. Hoffmann, J., Nebel, B.: The FF planning system: Fast plan generation through heuristic search. CoRR (2011)
8. Mattmüller, R., Ortlieb, M., Helmert, M., Bercher, P.: Pattern database heuristics for fully observable nondeterministic planning. In: Proceedings of the 20th ICAPS, Toronto, Ontario, Canada, May 12-16, 2010. (2010) 105–112
9. Hansen, E.A., Zilberstein, S.: Lao*: A heuristic search algorithm that finds solutions with loops. Artif. Intell. **129**(1-2) (2001) 35–62
10. Winterer, D., Wehrle, M., Katz, M.: Structural symmetries for fully observable nondeterministic planning. In: Proceedings of the 25th IJCAI, New York, NY, USA, 9-15 July 2016. (2016) 3293–3299
11. Alchourrón, C.E., Gärdenfors, P., Makinson, D.: On the logic of theory change: Partial meet contraction and revision functions. J. Symb. Log. **50**(2) (1985) 510–530
12. 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. (1991) 387–394
13. Darwiche, A., Pearl, J.: On the logic of iterated belief revision. Artif. Intell. **89**(1-2) (1997) 1–29
14. Satoh, K.: Nonmonotonic reasoning by minimal belief revision. In: FGCS. (1988) 455–462
15. Winslett, M.: Reasoning about action using a possible models approach. In: Proceedings of the 7th National Conference on Artificial Intelligence. St. Paul, MN, August 21-26, 1988. (1988) 89–93
16. McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., Wilkins, D.: Pddl-the planning domain definition language. Technical Report CVC TR98003/DCS TR1165, Yale Center for Computational Vision and Control, New Haven, CT. (1998)