Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 398245, 6 pages
http://dx.doi.org/10.1155/2013/398245

Research Article

On the Completeness of Pruning Techniques for Planning with Conditional Effects

Dunbo Cai,[1,2] Sheng Xu,[1,2] Tongzhou Zhao,[1,2] and Yanduo Zhang[1,2]

[1] *School of Computer Science and Engineering, Wuhan Institute of Technology, Wuhan 430205, China*
[2] *Hubei Province Key Laboratory of Intelligent Robot, Wuhan Institute of Technology, Wuhan 430205, China*

Correspondence should be addressed to Yanduo Zhang; zhangyanduo@hotmail.com

Received 18 May 2013; Accepted 17 July 2013

Academic Editor: William Guo

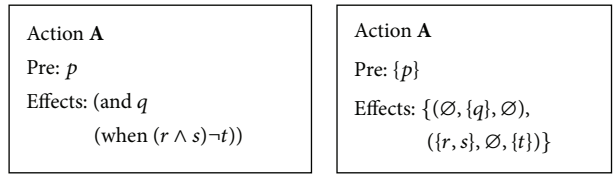
Copyright © 2013 Dunbo Cai et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Pruning techniques and heuristics are two keys to the heuristic search-based planning. The *helpful actions* pruning (HAP) strategy and *relaxed-plan-based heuristics* are two representatives among those methods and are still popular in the state-of-the-art planners. Here, we present new analyses on the properties of HAP. Specifically, we show new reasons for which HAP can cause incompleteness of a search procedure. We prove that, in general, HAP is incomplete for planning with conditional effects if factored expansions of actions are used. To preserve completeness, we propose a pruning strategy that is based on *relevance analysis* and *confrontation*. We will show that both *relevance analysis* and *confrontation* are necessary. We call it the *confrontation and goal relevant actions* pruning (CGRAP) strategy. However, CGRAP is computationally hard to be exactly computed. Therefore, we suggest practical approximations from the literature.

1. Introduction

The research of AI planning has advanced to a new level. Pioneers from the AI planning community have developed various practical methods that can solve much larger problems than those toy problems in early days. Two such well known methods are SAT-*based planning* [1], and *heuristic search-based planning* [2]. SAT-based planning translates planning problems into propositional satisfiability problems or more general constraint satisfaction problems (CSPs) [3]. An obvious advantage of the method is that it can exploit the power of fast algorithms from the CSP literature [4–6]. On the other hand, the "planning as search" community has been pursuing more informative heuristics to make the search fast [7, 8]. While recent studies show that *planning specific* heuristics can make SAT-based planning methods more competitive [9, 10], the *planning as search* method has shown its potential in many kinds of planning problems including classical planning [11, 12], conformant planning [13], contingent planning [14], and probabilistic planning [15]. Recent International Planning Competitions (IPCs) http://ipc.icaps-conference.org/ have witnessed the success of heuristic search-based planners, since the winners: Fast-Forward (FF) [11], LPG [16], SGPlan [17], and Fast Downward [12] and its successors—LAMA [18] all employ heuristic search. Two enabling techniques underlying heuristic search-based planning are *heuristic functions* and *pruning techniques*. A heuristic function measures the distances to goal of states, while pruning techniques eliminate branches that are safe to ignore. Here, we focus on the pruning techniques.

HAP is a pruning strategy developed in FF with the idea of making the search process goal directed [19].Though, initially, it was a byproduct of the *relaxed-plan heuristic*, its notion has been popular and important in the design of top performance planners, such as Fast Downward [12]. However, the HAP strategy does not guarantee completeness; that is, it may cut branches that can reach the goal. Some of these cases were explained by Hoffmann and Nebel [11]. Nevertheless, here, we uncover a new case in which HAP can cause incompleteness. We study the conditions under which the new case will occur, the way to remedy the HAP strategy, and the cost of doing that. Based on our work, one can gain more insights into why Fast Downward, which employs the *helpful transitions* strategy, is powerful.



(a) (b)

Figure 1: An action with conditional effects (a) and its compiling result by IPP (b).

The rest of the paper is organized as follows. In the next section, we introduce some background. Then, we show the incompleteness of HAP and extend it to a more general one called goal relevant actions pruning (GRAP), which is complete only for STRIPS planning. In Section 4, we propose our *confrontation and goal relevant actions* pruning (CGRAP), which is complete for both STRIPS planning and actions with conditional effects. In Section 5, we discuss some pruning techniques in the literature that can be seen as approximations of CGRAP. Finally, we conclude the paper and discuss some future work.

2. Background

We will first introduce notations from the state space search-based planning, then methods for handling conditional effects, and finally the heuristic function and the HAP strategy in FF.

A planning task is a quadtuple $\mathbf{T} = (\mathbf{P}, \mathbf{A}, \mathbf{I}, \mathbf{G})$, where $\mathbf{P}$ is the set of atoms, $\mathbf{A}$ is the set of actions, $\mathbf{I} \subseteq \mathbf{P}$ is a set of atoms called the initial state, and $\mathbf{G} \subseteq \mathbf{P}$ is the goal condition that each goal state must fulfill. States are denoted by sets of atoms. We adopted the "closed world" assumption; that is, an atom that is not in a state is false in the state. So, if $\mathbf{P} = \{p, q, r\}$ and a state $s = \{p, q\}$, then s is logically equal to $p \wedge q \wedge \neg r$. An action a is a pair $\langle \text{pre}(a), E(a)\rangle$, where $\text{pre}(a)$ is a set of atoms denoting the preconditions of a and $E(a)$ is the set of conditional effects of a. Each conditional effect $e \in E(a)$ has the form $\langle \text{con}(e), \text{add}(e), \text{del}(e)\rangle$, where $\text{con}(e)$, $\text{add}(e)$, and $\text{del}(e)$ are conditions, add effects, and delete effects of e, respectively. For an action a and a state s, if $\text{pre}(a) \subseteq s$, then we say a is applicable in s. We use $\text{App}(s) = \{a \mid \text{pre}(a) \subseteq s\}$ to denote all the actions that are applicable in s. The execution of a on s, denoted by $a(s)$, results in a state s', where $s' = s - \bigcup_{e:\text{con}(e)\subseteq s} \text{del}(e) + \bigcup_{e:\text{con}(e)\subseteq s} \text{add}(e)$ if a is applicable in s, and $s' = s$ otherwise. A state s is called a goal state if $G \subseteq s$. A plan for a planning task is an action sequence $\pi = \langle a_0, a_1, \ldots, a_{n-1}\rangle$ that transforms the initial state I into a goal state. We use $|\pi|$ to denote its length, which is the number of actions in it. Here, we assume that a plan does not have redundant actions; that is, when some action is removed from π, it will no longer be a plan.

Actions with conditional effects were introduced in the planning problem description language "Action description language" (ADL) [20]. And there are mainly three ways for handling conditional effects. Conditional effects of actions are expressed with the keyword "when". Figure 1 (left) shows an action with two conditional effects: the first effect q will happen with no condition and the second effect $\neg t$ will happen if $r \wedge s$ holds in the state where the action is executed. The three ways for handling conditional effects are, full expansion [21], IPP's method [22] and factored expansion [23]. Here, we focus on IPP's method, which is used in FF. The method will translate the action in Figure 1 (left) into the form shown in Figure 1 (right). From now on, we will call planning with actions with conditional effects ADL planning.

FF employs a forward state space search framework. Three key techniques of FF are the relaxed-plan-based heuristic (RP), HAP, and the enforced hill-climbing (EHC) algorithm. Here, we focus on RP and HAP. A relaxed plan is extracted from a relaxed version of a planning task where the delete effects of actions are ignored. Specifically, the relaxed version of a planning task $\mathbf{T} = (\mathbf{P}, \mathbf{A}, \mathbf{I}, \mathbf{G})$ is $\mathbf{T}' = (\mathbf{P}, \mathbf{A}', \mathbf{I}, \mathbf{G})$, where $\mathbf{A}' = \{\langle \text{pre}(a), E'(a)\rangle \mid a \in \mathbf{A}\}$ and $E'(a) = \{\langle \text{con}(e), \text{add}(e), \emptyset\rangle \mid \langle \text{con}(e), \text{add}(e), \text{del}(e)\rangle \in E(a)\}$. An action sequence $\pi = \langle a_0, a_1, \ldots, a_{n-1}\rangle$ is called a *relaxed plan* for $\mathbf{T}$ if it is a *plan* for $\mathbf{T}'$. For a state s, its heuristic value is the length of a relaxed plan for the planning task $\mathbf{T}_\mathbf{s} = (\mathbf{P}, \mathbf{A}, \mathbf{s}, \mathbf{G})$. Note that the relaxed plan for $\mathbf{T}_\mathbf{s}$ is not unique and FF finds one using the Graphplan [24] algorithm. During the process of computing a relaxed plan, FF keeps track of the subgoals generated in the second propositional level of a planning graph, which is saved in a set $G_1(s)$. Helpful actions are the set of actions $H(s) = \{a \mid \text{pre}(a) \subseteq s \wedge \exists e \in E(a) : (\text{con}(e) \subseteq s) \wedge (\text{add}(e) \cap G_1(s) \neq \emptyset)\}$. For a state s, the EHC algorithm only considers actions in $H(s)$ and ignores others. This search strategy is called *helpful actions* pruning. We say that a strategy is complete if using it does not make a complete algorithm eliminate search branches that are directions to goal states. As shown in [11], HAP is incomplete for STRIPS planinng.

3. Complete Pruning Strategy for STRIPS

In this section, we will extend HAP to a complete strategy for STRIPS that we call *goal relevant actions* pruning (GRAP). We will then prove the completeness of GRAP for STRIPS and show that GRAP is incomplete for ADL planning. As a result, we will extend GRAP to a complete strategy for ADL planning in the next section.

3.1. Goal Relevant Actions. Helpful actions for a state s are actions in $\text{App}(s)$ that is relevant for adding the subgoals in $G_1(s)$. To obtain completeness, our goal relevant actions

for s are actions in App(s) that is relevant for adding every (sub)goal generated by the GraphPlan algorithm.

Definition 1 (Dependency among Facts). For two facts $l, g \in \mathbf{P}$ and a set of actions $\mathbf{A}$, l is *dependent on* g with respect to $\mathbf{A}$ (denoted as $l \lhd_{\mathbf{A}} g$) if

(1) $l = g$,

(2) $\exists a \in \mathbf{A} : \exists e \in E(a) : (l \in \text{add}(e) \wedge g \in (\text{pre}(a) \cup \text{con}(e)))$,

(3) $\exists h \in P : (l \lhd_{\mathbf{A}} h \wedge h \lhd_{\mathbf{A}} g)$.

Definition 2 (Dependency between Facts and Actions). For an atom $l \in \mathbf{P}$ and an action $a \in \mathbf{A}$, l is *dependent on* a with respect to $\mathbf{A}$ (denoted as $l \lhd_{\mathbf{A}} a$) if

(1) $\exists a \in \mathbf{A} : \exists e \in E(a) : l \in \text{add}(e)$,

(2) $\exists g \in P : l \lhd_{\mathbf{A}} g \wedge g \lhd_{\mathbf{A}} a$.

We note that Definitions 1 and 2 capture the relevant facts and actions to a goal. Specifically, if we are going to reach a goal g, then the actions on which g is dependent are relevant, and further, actions that adds facts on which g is dependent are also relevant. Note that in the previous definitions we use "dependent," instead of "relevant," to indicate a directional relation.

Now, we are ready to introduce the notion of goal relevant actions. The actions are those a search algorithm could explore for reaching some goal state. Actions that are not relevant are to be ignored.

Definition 3 (Goal Relevant Actions, GRA). Given a planning task $\mathbf{T} = (\mathbf{P}, \mathbf{A}, \mathbf{I}, \mathbf{G})$, actions that are relevant to $l \in \mathbf{P}$ are $\text{DEP}_{\lhd_{\mathbf{A}}}(l) = \{a \mid a \in \mathbf{A} \text{ and } l \lhd_{\mathbf{A}} a\}$, actions that are relevant to $\mathbf{G} \subseteq \mathbf{P}$ are $\text{DEP}_{\lhd_{\mathbf{A}}}(\mathbf{G}) = \bigcup_{l \in \mathbf{G}} \text{DEP}_{\lhd_{\mathbf{A}}}(l)$. Given a state s, the "goal relevant actions" for s is $\text{REL}_{\mathbf{G}, \lhd_{\mathbf{A}}}(s) = \text{DEP}_{\lhd_{\mathbf{A}}}(\mathbf{G}) \cap \text{App}(s)$.

We propose the following pruning strategy based on GRA. For any search algorithm and any state s, we only consider actions in $\text{REL}_{G, \lhd_{\mathbf{A}}}(s)$ and ignore those in App(s) – $\text{REL}_{G, \lhd_{\mathbf{A}}}(s)$. We call the strategy GRA pruning (GRAP). We will prove that GRAP is a generalization of HAP and is complete for STRIPS planning.

Proposition 4. *For a planning task* $\mathbf{T} = (\mathbf{P}, \mathbf{A}, \mathbf{I}, \mathbf{G})$ *and any state* s *of* T, $H(s) \subseteq REL_{G, \lhd_{\mathbf{A}}}(s)$.

As the directional relation $\lhd_{\mathbf{A}}$ is transitive, the correctness of Proposition 4 is straightforward in that $G_1(s) \subseteq \text{DEP}_{\lhd_{\mathbf{A}}}(G)$.

Next, we will prove that GRAP is complete for STRIPS planning.

Proposition 5. *GRAP is a complete pruning strategy for STRIPS planning.*

Proof. Let $\mathbf{T} = (\mathbf{P}, \mathbf{A}, \mathbf{s}, \mathbf{G})$ be a STRIPS planning task, and let $\pi = \langle a_0, a_1, \ldots, a_{n-1} \rangle$ be one of the plans for s. Note that π is not redundant. As we restrict to STRIPS planning, each action $a \in \mathbf{A}$ has only one conditional effect, which is denoted by $e(a)$. We will prove that $a_0 \in \text{REL}_{G, \lhd_{\mathbf{A}}}(s)$. We will compute $\text{DEP}_{\lhd_{\mathbf{A}}}(G)$ with $k = n-1, \ldots, 0$. Initially, $\text{DEP}_{\lhd_{\mathbf{A}}}(G) = G$. For the action a_{n-1}, if its effect $e(a_{n-1})$ does not add a fact in G then π with a_{n-1} removed will still be a plan. This contradicts the assumption that π is not redundant. Following Definition 1, $\text{DEP}_{\lhd_{\mathbf{A}}}(G) = \text{DEP}_{\lhd_{\mathbf{A}}}(G) \cup \text{pre}(a_{n-1})$. Similarly, for $a_k (0 < k < n-1)$, a_k must add a fact in $\text{DEP}_{\lhd_{\mathbf{A}}}(G) \cup \text{pre}(a_{n-1}) \cup \cdots \cup \text{pre}(a_{k+1})$; otherwise, π with a_k removed is still a plan, which contradicts the assumption that π is not redundant. Therefore, $\text{DEP}_{\lhd_{\mathbf{A}}}(G) = \text{DEP}_{\lhd_{\mathbf{A}}}(G) \cup \text{pre}(a_{n-1}) \cup \cdots \cup \text{pre}(a_{k+1})$. For a_0, as π is not redundant, it must hold that $e(a_0) \cap (G \cup \text{pre}(a_{n-1}) \cup \cdots \cup \text{pre}(a_1)) \neq \emptyset$. Following Definition 2, a_0 is in $\text{DEP}_{\lhd_{\mathbf{A}}}(G)$. And according to Definition 3, $a_0 \in \text{REL}_{G, \lhd_{\mathbf{A}}}(s)$ holds. As π is an arbitrary plan, we finish the proof. □

Note that the above proof cannot be adapted to prove the completeness of HAP, as helpful actions are defined with respect to a specific relaxed plan. In other words, the arbitrariness of plans is not guaranteed.

3.2. GRAP is Incomplete for ADL Planning. Hoffmann and Nebel [11] pointed out that HAP is incomplete as the GraphPlan algorithm is greedy in computing shorter relaxed plans. Therefore, the source of this incompleteness could be eliminated if we use other algorithms to compute relaxed plans, other than GraphPlan. The method proposed by Hoffmann and Nebel [11] works in the following way: for a state s and a relaxed planning task $\mathbf{T}' = (\mathbf{P}, \mathbf{A}', \mathbf{s}, \mathbf{G})$, they expand the planning graph to the level $|\mathbf{A}'|$ and collect subgoals backward from the level $|\mathbf{A}'|$ to level 1. Specifically, let G'_i be the subgoals at level i, then

$$G'_{|\mathbf{A}'|} = \mathbf{G},$$
$$G'_i = G'_{i+1} \cup \bigcup_{a \in A_i \wedge (\exists e \in E(a) : \text{add}(e) \cap G_{i+1} \neq \emptyset)} \text{pre}(e) \cup \text{con}(e). \quad (1)$$

Following the method, G'_1 is the union of subgoals of every relaxed plan for T' at level 1. We call actions in $\{a \mid a \in \text{App}(s), \exists e \in E(a) : \text{add}(e) \cap G'_1 \neq \emptyset\}$ "full helpful actions" (FHAs). Intuitively, an FHA is equivalent to our definition $\text{REL}_{G, \lhd_{\mathbf{A}}}(s)$. However, the former is more procedural, and ours is more formal. We will use our definition to develop a new pruning strategy for ADL planning. Before that, we will show, by example, that both the FHA pruning (FHAP) strategy and GRAP are generally *incomplete* for ADL planning.

Example 6. Given a planning task $\mathbf{T}''$, where $\mathbf{P}'' = \{p, q, r, s, m\}$, $\mathbf{I}'' = \{p, q, r\}$, $\mathbf{G}'' = \{r, m\}$, and $\mathbf{A}'' = \{a_1, a_2\}$ where a_1 is $(\{p\}, \{(\emptyset, \{m\}, \emptyset), (\{q\}, \{m\}, \{r\})\})$ and a_2 is $(q, (\emptyset, \emptyset, \{q\}))$. The meaning of a_1 is as follows: its preconditions are $\{q\}$ and it has two conditional effects—the first is $(\emptyset, \{m\}, \emptyset)$ (denoted as e_0) and the second is $(\{q\}, \{m\}, \{r\})$ (denoted as e_1). The action a_2 has one condition q, and has a conditional effect $(\emptyset, \emptyset, \{q\})$ that falsifies q.

Next, let us consider the plan for Example 6. The difference between $\mathbf{G}''$ and $\mathbf{I}''$ is that the atom m is not *true* in $\mathbf{I}''$.

To make m true, we would use action a_1. However, executing a_1 on $\mathbf{I}''$ will result in a state $s'' = \{p, q, m\}$ where atom r does not hold. After that, there is no action that can transform s'' into a goal state. One could notice that this dead end is due to the fact that e_0 and e_1 both happened and e_1 destroyed r. If we could prevent e_1 from happening, then we would succeed in finding a plan. This is the idea proposed by Weld [25], which is called "confrontation." It is easy to see that with "confrontation" as a choice, we can find a plan $< a_1, a_0 >$.

In Example 6, a_1 is relevant for reaching a goal state. However, pruning strategies HAP, GRAP, and FHAP all ignore it. With a generalization, we have the following results.

Proposition 7. *Let* $\mathbf{T} = (\mathbf{P}, \mathbf{A}, \mathbf{s}, \mathbf{G})$ *be an ADL planning task and the set of plans for* $\mathbf{T}$ *be* $PLANs(\mathbf{T})$. *If every plan* $\pi \in PLANs(\mathbf{T})$ *contains an action* a *of the form* $(\mathrm{pre(a)}, \{(\mathrm{con}(e_0), \emptyset, \mathrm{del}(e_0)), \ldots, (\mathrm{con}(e_m), \emptyset, \mathrm{del}(e_m))\})$, *then HAP, FHAP, and GRAP are incomplete for* $\mathbf{T}$.

The correctness of Proposition 7 is in that if an action does not have any add effects, then it will not be considered as "helpful" or "relevant" anyway. As a result, this kind of actions will be mistakenly ignored.

From Example 6, we can see that actions that make "confrontations" are also "helpful" and "relevant." Following this direction, we extend GRAP to a new pruning strategy that is complete for both STRIPS and ADL planning.

4. Complete Pruning Strategy for ADL Planning

We first introduce the notion of "confrontation and goal relevant actions" and then prove that its corresponding pruning strategy CGRAP is complete for ADL planning.

Definition 8 (Confrontational Dependency among Facts). For two atoms l, $g \in \mathbf{P}$, a set of actions $\mathbf{A}$, l is *confrontationally dependent on* g with respect to $\mathbf{A}$ (denoted as $l \trianglelefteq_{\mathbf{A}} g$) if

(1) $l = g$,

(2) $\exists a \in \mathbf{A} : \exists e \in E(a) : l \in ((\mathrm{add}(e) \cup \mathrm{del}(e)) \wedge g \in (\mathrm{pre}(a) \cup \bigcup_{e' \in E(a)} \mathrm{con}(e'))$,

(3) $\exists h \in \mathbf{P} : l \trianglelefteq_{\mathbf{A}} h \wedge h \trianglelefteq_{\mathbf{A}} g$.

Definition 9 (Confrontational Dependency between Facts and Actions). For two atoms $l, g \in \mathbf{P}$, an action $a \in \mathbf{A}$, l is *confrontationally dependent on* a with respect to $\mathbf{A}$ (denoted as $l \trianglelefteq_{\mathbf{A}} a$) if

(1) $\exists a \in \mathbf{A} : \exists e \in E(a) : l \in (\mathrm{add}(e) \cup \mathrm{del}(e))$,

(2) $\exists g \in \mathbf{P} : l \trianglelefteq_{\mathbf{A}} g \wedge g \trianglelefteq_{\mathbf{A}} a$.

According to the previous two definitions, one could notice that actions that add or *delete* an atom l are considered as relevant to l.

Definition 10 (Confrontation and Goal Relevant Actions). Given a planning task $\mathbf{T} = (\mathbf{P}, \mathbf{A}, \mathbf{I}, \mathbf{G})$, actions that are confrontationally relevant to $l \in \mathbf{P}$ are $\mathrm{DEP}_{\trianglelefteq_{\mathbf{A}}}(l) = \{a \mid a \in \mathbf{A} \text{ and } l \trianglelefteq_{\mathbf{A}} a\}$, and actions that are confrontationally relevant to $\mathbf{G} \subseteq \mathbf{P}$ are $\mathrm{DEP}_{\trianglelefteq_{\mathbf{A}}}(\mathbf{G}) = \bigcup_{l \in \mathbf{G}} \{\mathrm{DEP}_{\trianglelefteq_{\mathbf{A}}}(l)\}$. Given a state s, the "confrontation and goal relevant actions" for s is $\mathrm{REL}_{\mathbf{G}, \trianglelefteq_{\mathbf{A}}}(s) = \mathrm{DEP}_{\trianglelefteq_{\mathbf{A}}}(\mathbf{G}) \cap \mathrm{App}(s)$.

The pruning strategy that considers actions in $\mathrm{REL}_{\mathbf{G}, \trianglelefteq_{\mathbf{A}}}(s)$ only, that is, ignores $\mathrm{App}(s) - \mathrm{REL}_{\mathbf{G}, \trianglelefteq_{\mathbf{A}}}(s)$ is called *confrontation and goal relevant actions* pruning (CGRAP). In the following proposition, we will prove that CGRAP is complete for ADL planning.

Proposition 11. *CGRAP strategy is complete for ADL planning.*

Proof (Proof by Contradiction). Given a planning task $\mathbf{T} = (\mathbf{P}, \mathbf{A}, \mathbf{I}, \mathbf{G})$ and any plan $\pi = \langle a_0, a_1, \ldots, a_{n-1} \rangle$ for it, we use $s_{j+1} = a_j(s_j)$ to denote the result of executing a_j on s_j. Without loss of generality, suppose that $a_i \, (i = 0, \ldots, n-1)$ is eliminated by CGRAP; that is, $a_i \notin \mathrm{REL}_{\mathbf{G}, \trianglelefteq_{\mathbf{A}}}(s_i)$. When $i = n-1$, as π is not a redundant plan, a_{n-1} must add one atom of $\mathbf{G}$; that is, $\exists e \in E(a_{n-1}) : \mathrm{add}(e) \cap \mathbf{G} \neq \emptyset$. Therefore, $a_i \in \mathrm{DEP}_{\trianglelefteq_{\mathbf{A}}}(\mathbf{G})$. For $i \leq n-2$, a_i either (1) adds an atom $g \in \mathbf{G}$, (2) adds an atom q which is the (pre)condition of $a_{i+1}, \ldots$, or a_{n-1}, or (3) deletes one condition r of $e' \in E(a_m)$ $(m = i+1, \ldots, n-1)$ in order to prevent e' from happening. In case (1), $\exists g \in \mathbf{G} : g \trianglelefteq_{\mathbf{A}} a_i$, in case (2), $\exists g' \in \mathbf{G} : g' \trianglelefteq_{\mathbf{A}} q$ *and* $q \trianglelefteq_{\mathbf{A}} a_i$, and in case (3), $\exists g'' \in \mathbf{G} : g'' \trianglelefteq_{\mathbf{A}} r$ *and* $r \trianglelefteq_{\mathbf{A}} a_i$. In either cases, and according to our Definitions 8 and 9, $a_i \in \mathrm{DEP}_{\trianglelefteq_{\mathbf{A}}}(G)$ holds. Note that $\mathrm{pre}(a_i) \subseteq s_i$. Therefore, we conclude that $a_i \in \mathrm{REL}_{\mathbf{G}, \trianglelefteq_{\mathbf{A}}}(s_i)$. □

The completeness of CGRAP for ADL planning costs. One reason is that the pruning power of CGRAP is weak. In other words, CGRAP may cut a rather limited amount of branches of a search space. In addition, computing CGRAP is PSPACE-hard, as deciding irrelevant actions for a planning task is PSPACE-hard [26]. Therefore, it is practical to collect *confrontation and goal relevant actions* in an approximate way. In the next section, we will discuss some methods from the literature that fall into this scope.

5. Discussion

We will first review the *helpful transition* notion developed in Fast Downward [12] and then the *delayed partly reasoning procedure* [27].

Fast Downward is a representative planner that uses the SAS^+ planning formalism [28], which supports multivalued variables. It translates a propositional planning problem into an SAS^+ planning problem by utilizing an invariants analysis procedure [12]. After that, Fast Downward builds a *causal graph* that involves all the variables and a *domain transition graph* for each variable. Dependencies among variables are reasoned through the causal graph, and dependencies among values of a variable are reasoned through the corresponding domain transition graph. For details of the two kinds of graphs, please refer to [12]. The goal distance of a state s is the sum of goal distances of variables. For each variable, its goal distance is computed by solving a shortest path problem

formulated on the corresponding domain transition graph. In the problem, the source node is the value the variable currently takes, and the target node is the value that goal conditions require. When such a path is obtained, the transition associated with the first edge is labeled as "helpful transition." Note that transitions are conditional effects of actions. So, we can collect "helpful actions" based on "helpful transitions." Here we note that "helpful transitions" consider both goal relevant actions and actions that are for confrontations. The ability of collecting actions that are helpful for confrontations originates from the multivalued variable representation. In the representation, the change from one value to another one models both the *add and delete effects* of an action on a variable. As a result, actions have only one kind of effects, which are considered by Fast Downward to collect "helpful transitions." In contrast, propositional planners, such as FF, consider only the *add effects* of actions for collecting "helpful actions." Therefore, the "helpful transitions" strategy can be considered as an approximation of CGRAP.

The "delayed partly reasoning procedure" is proposed by Cai et al. [7]. This procedure is implemented on top of the propositional planning formalism. In the first action level of a planning graph, the procedure tracks harmful inducements with respect to an order of conditional effects and collects actions that confront the inducements. An inducement is that one conditional effect e induces another conditional effect e'. It is harmful if e' deletes some previously added atoms of other conditional effects. As the procedure only operates on the first actions level and works with a predefined order, it is an approximation of CGRAP. Therefore, the computational cost of the procedure is not high.

6. Conclusions and Future Work

In this work, we analyzed some well-known pruning techniques, which are currently utilized by state-of-the-art planners. In particular, we showed that the *helpful actions* pruning strategy is incomplete for ADL planning and extended it to a complete strategy called *confrontation and goal relevant actions* pruning. Though our proposed strategy is computationally hard, we discussed methods from the AI planning literature that can be seen as approximations of it. In addition, we believe that this work will help us gain more insights into why the planner Fast Downward is powerful.

This work was done on pruning techniques in search-based planning. Future directions may consider pruning techniques in SAT-based ADL planning and conformant planning. As IPP's method for handling conditional effects does not lead to a high increase in problem size, it is suitable for SAT-based planning. Therefore, developing adaptations or approximations of our proposed strategy CGRAP in that settings could be interesting.

Acknowledgments

This work is supported by Natural Science Foundation of China (Grant no. 61103136), Educational Commission of Hubei Province of China (Grant no. D20111507), Hubei Province Key Laboratory of Intelligent Robot Open Foundation (Grant no. HBIR200909), and Youths Science Foundation of Wuhan Institute of Technology (Grant no. 12106022).

References

[1] H. Kautz and B. Selman, "Pushing the envelope: planning, propositional logic, and stochastic search," in *Proceedings of the 13th National Conference on Artificial Intelligence*, pp. 1194–1201, August 1996.

[2] B. Bonet and H. Geffner, "Planning as heuristic search," *Artificial Intelligence*, vol. 129, no. 1-2, pp. 5–33, 2001, Heuristic search in artificial intelligence.

[3] M. B. Do and S. Kambhampati, "Planning as constraint satisfaction: solving the planning graph by compiling it into CSP," *Artificial Intelligence*, vol. 132, no. 2, pp. 151–182, 2001.

[4] N. Eén and A. Biere, "Effective preprocessing in SAT through variable and clause elimination," in *Theory and Applications of Satisfiability Testing*, vol. 3569, pp. 61–75, Springer, Berlin, Germany, 2005.

[5] X. Li and M. Yin, "An opposition-based differential evolution algorithm for permutation flow shop scheduling based on diversity measure," *Advances in Engineering Software*, vol. 55, pp. 10–31, 2013.

[6] X. Li and M. Yin, "A discrete artificial bee colony algorithm with composite mutation strategies for permutation flow shop scheduling problem," *Scientia Iranica*, vol. 19, no. 6, pp. 1921–1935, 2012.

[7] D. Cai, J. Sun, and M. Yin, "Making FF faster in ADL domains," in *Proceedings of the 3rd International Conference on Natural Computation (ICNC '07)*, vol. 2, pp. 160–164, August 2007.

[8] M. Katz, J. Hoffmann, and C. Domshlak, "Who said we need to relax all variables?" in *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS '13)*, AAAI Press, 2013.

[9] P. H. Tu, T. C. Son, M. Gelfond, and A. R. Morales, "Approximation of action theories and its application to conformant planning," *Artificial Intelligence*, vol. 175, no. 1, pp. 79–119, 2011.

[10] J. Rintanen, "Planning as satisfiability: heuristics," *Artificial Intelligence*, vol. 193, pp. 45–86, 2012.

[11] J. Hoffmann and B. Nebel, "The FF planning system: fast plan generation through heuristic search," *Journal of Artificial Intelligence Research*, vol. 14, pp. 253–302, 2001.

[12] M. Helmert, "The fast downward planning system," *Journal of Artificial Intelligence Research*, vol. 26, pp. 191–246, 2006.

[13] J. Hoffmann and R. I. Brafman, "Conformant planning via heuristic forward search: a new approach," *Artificial Intelligence*, vol. 170, no. 6-7, pp. 507–541, 2006.

[14] J. Hoffmann and R. Brafman, "Contingent planning via heuristic forward search with implicit belief states," in *Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS' 05)*, vol. 2005, 2005.

[15] B. Bonet and E. Hansen, "Heuristic Search for Planning under Uncertainty," in *Heuristics, Probability and Causality: A Tribute To Judea Pearl*, pp. 3–22, College Publications, 2010.

[16] A. Gerevini, I. Serina, A. Saetti, and S. Spinoni, "Local search techniques for temporal planning in lpg," in *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS '03)*, pp. 62–72, 2003.

[17] Y. Chen, B. W. Wah, and C.-W. Hsu, "Temporal planning using subgoal partitioning and resolution in SGPlan," *Journal of Artificial Intelligence Research*, vol. 26, pp. 323–369, 2006.

[18] S. Richter, M. Helmert, and M. Westphal, "Landmarks revisited," in *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pp. 975–982, July 2008.

[19] D. McDermott, "Using regression-match graphs to control search in planning," *Artificial Intelligence*, vol. 109, no. 1, pp. 111–159, 1999.

[20] E. P. D. Pednault, "ADL: exploring the middle ground between STRIPS and the situation calculus," in *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning*, pp. 324–332, 1989.

[21] B. C. Gazen and C. A. Knoblock, "Combining the expressivity of ucpop with the efficiency of graphplan," in *Proceedings of the 4th European Conference on Planning (ECP '97)*, pp. 221–233, 1997.

[22] J. Koehler, B. Nebel, J. Hoffmann, and Y. Dimopoulos, "Extending planning graphs to an adl subset," in *Proceedings of the 4th European Conference on Planning: Recent Advances in AI Planning (ECP '97)*, pp. 273–285, 1997.

[23] C. R. Anderson, D. E. Smith, and D. S. Weld, "Conditional effects in graphplan," in *Proceedings of the AIPS*, pp. 44–53, 1998.

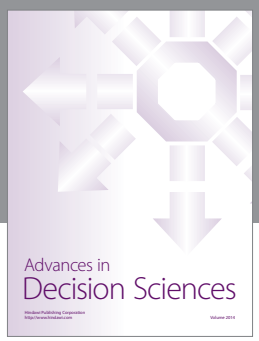
[24] A. L. Blum and M. L. Furst, "Fast planning through planning graph analysis," *Artificial Intelligence*, vol. 90, no. 1-2, pp. 281–300, 1997.

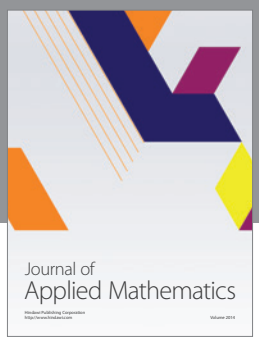
[25] D. S. Weld, "Introduction to least commitment planning," *AI Magazine*, vol. 15, no. 4, pp. 27–61, 1994.

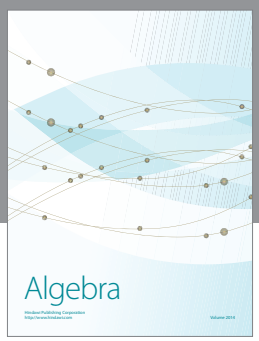
[26] B. Nebel, Y. Dimopoulos, and J. Koehler, "Ignoring irrelevant facts and operators in plan generation," in *Proceedings of the European Conference on Planning (ECP '97)*, pp. 338–350, 1997.

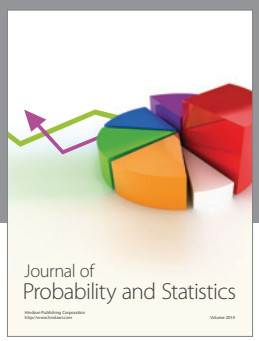
[27] D. Cai, J. Hoffmann, and M. Helmert, "Enhancing the context-enhanced additive heuristic with precedence constraints," in *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS '09)*, pp. 50–57, grc, September 2009.

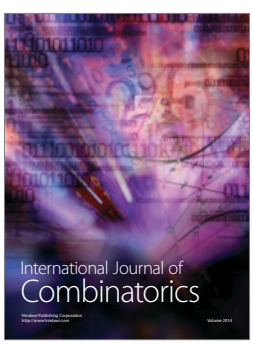
[28] C. Bäckström and B. Nebel, "Complexity results for SAS+ planning," *Computational Intelligence*, vol. 11, no. 4, pp. 625–655, 1995.

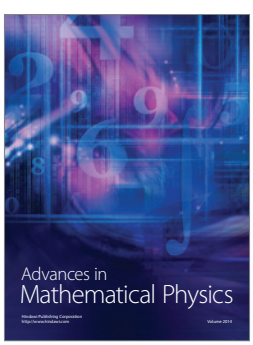


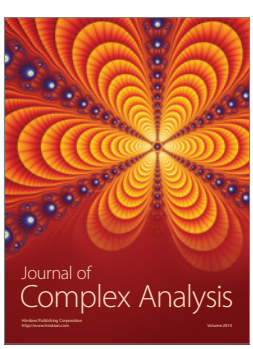


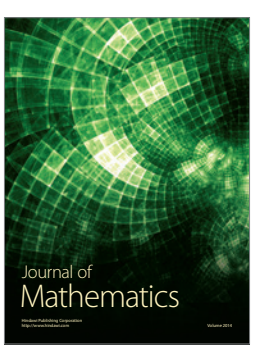


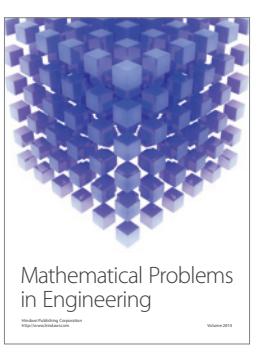


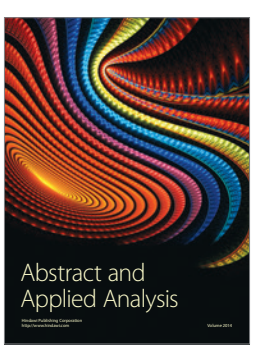


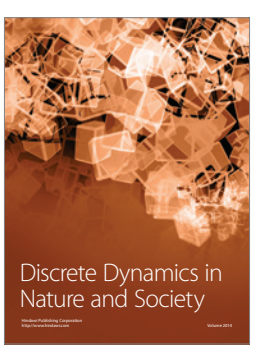


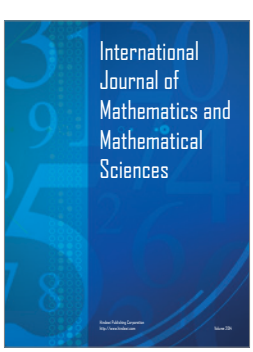












Hindawi

Submit your manuscripts at
http://www.hindawi.com