ECAI 2014
T. Schaub et al. (Eds.)

doi:10.3233/978-1-61499-419-0-765

Optimal Planning in the Presence of Conditional Effects: Extending LM-Cut with Context Splitting

Gabriele Röger and **Florian Pommerening** and **Malte Helmert**[1]

Abstract. The LM-Cut heuristic is currently the most successful heuristic in optimal STRIPS planning but it cannot be applied in the presence of conditional effects. Keyder, Hoffmann and Haslum recently showed that the obvious extensions to such effects ruin the nice theoretical properties of LM-Cut. We propose a new method based on context splitting that preserves these properties.

1 INTRODUCTION

The aim of classical planning is to find a sequence of actions that leads from the current world state to some desired state. Conditional effects enable situation-dependent behavior of actions. For example, there can be an action *stop-*f in an elevator domain that boards waiting passengers at floor f and disembarks all passengers with destination f. To describe such a behavior without conditional effects, one would need specific actions for all different situations of waiting and boarded passengers related to this floor, or use some other formulation that applies several actions to cause the same world change.

Conditional effects can be compiled away [9] but only with severe disadvantages: any plan-preserving transformation leads to an exponential blow-up of the task description size. An alternative compact compilation does not preserve the delete relaxation, which many heuristics such as the LM-Cut heuristic [6] are based on. As a result, these heuristics do not give good guidance on such compiled tasks.

Haslum [4] uses an incremental compilation approach for solving delete-relaxed tasks optimally: starting from the compact compilation (which can cause further relaxation), it successively introduces the exponential transformation until an optimal solution for the compiled task can be transformed into a plan for the original task. In the worst case, this can lead to the full exponential compilation.

We take the different approach of supporting conditional effects natively in the heuristic computation. This is not unusual for inadmissible heuristics but among current *admissible* heuristics (which are required for cost-optimal planning) the support is rather weak and a suitable extension to conditional effects is not always obvious.

For the state-of-the-art LM-Cut heuristic [6], Keyder et al. [7] recently pointed out that obvious extensions either render the heuristic inadmissible or lose the dominance over the maximum heuristic [1].

We present an extension of the LM-Cut heuristic that preserves both admissibility and dominance over the maximum heuristic. For this purpose we introduce *context splitting* as a new general technique which allows us to split up actions in a task to distinguish different scenarios of their application. We show how context splitting can be made useful for the extension of the LM-Cut heuristic. After proving the desired theoretical properties of the heuristic, we also evaluate its performance empirically.

[1] University of Basel, Switzerland

2 BACKGROUND

We consider propositional STRIPS planning with action costs, extended with conditional effects. In this formalism, which we denote as STRIPS^c, a task is given as a tuple $\Pi = \langle F, A, I, G, \mathit{cost}\rangle$ where F is a set of propositional variables (or *facts*), A is a set of actions, $I \subseteq F$ is the initial state, $G \subseteq F$ describes the goal, and the cost function $\mathit{cost} : A \to \mathbb{N}_0$ defines the cost of each action. A *state* $s \subseteq F$ of a task is given by the variables that are true in this state. Every action $a \in A$ is given as a pair $a = \langle \mathit{pre}(a), \mathit{eff}(a)\rangle$. The precondition $\mathit{pre}(a) \subseteq F$ defines when the action is applicable. The set of effects $\mathit{eff}(a)$ consists of conditional effects e, each given by a triple $\langle \mathit{cond}(e), \mathit{add}(e), \mathit{del}(e)\rangle$ where all components are (possibly empty) subsets of F. If all facts in the *effect condition* $\mathit{cond}(e)$ are true in the current state, the successor state is determined by removing all facts in the *delete effect* $\mathit{del}(e)$ and adding the facts in the *add effect* $\mathit{add}(e)$. Given an effect $e \in \mathit{eff}(a)$, we use the notation $\mathit{act}(e)$ to refer to the action a.

Action a is applicable in state s if $\mathit{pre}(a) \subseteq s$. The resulting successor state is

$$s[a] = \Big(s \setminus \bigcup_{\substack{e \in \mathit{eff}(a) \text{ with} \\ \mathit{cond}(e) \subseteq s}} \mathit{del}(e)\Big) \cup \bigcup_{\substack{e \in \mathit{eff}(a) \text{ with} \\ \mathit{cond}(e) \subseteq s}} \mathit{add}(e)$$

A plan for a state s is a sequence of actions whose sequential application leads from s to a state s^* such that $G \subseteq s^*$. A plan for the task is a plan for I. The cost of a plan is the sum of the action costs as given by *cost*, and an *optimal* plan is one with minimal cost. We denote the cost of an optimal plan for s in task Π with $h^*_\Pi(s)$.

A task where all effect conditions are empty is a standard STRIPS task (with action costs). In this case, we can combine all add (and delete) effects of an action a to a single set $\mathit{add}(a)$ (and $\mathit{del}(a)$).

When introducing context splitting, we will briefly consider the more general ADL formalism, where action preconditions and effect conditions are arbitrary propositional formulas over the task variables F. For a formal semantics, we need to regard a state $s \subseteq F$ as a truth assignment $T(s)$ that assigns 1 to the variables in s and 0 to all other variables. An action a is then applicable in a state s if $T(s) \models \mathit{pre}(a)$ and an effect e triggers if $T(s) \models \mathit{cond}(e)$. If not explicitly mentioned otherwise, we are talking about STRIPS^c tasks.

The delete relaxation Π^+ of a planning task Π is equivalent to Π except that all delete effects are replaced with the empty set. We call such a task *delete-free*. The cost of an optimal plan for a state s in Π^+ is denoted with $h^+(s)$ and is an admissible estimate for $h^*_\Pi(s)$ in Π. To simplify the notation throughout this paper, we avoid making the state s explicit in all definitions. Instead, we compute heuristic estimates for a state s from a modified task Π_s where we replace the

initial state with s. The heuristic estimate $h(s)$ then only depends on the task Π_s and we can write $h(\Pi_s)$ instead.

Since computing h^+ is NP-complete [3], it is often approximated by polynomial-time computable heuristics. One such heuristic, which is dominated by h^+ and therefore also admissible, is the *maximum heuristic* $h^{\max}$ [1]. It assigns a value $V^{\max}$ to variables and sets of variables. The value $V^{\max}(P)$ of a non-empty set of variables $P \subseteq F$ is the maximal value of any of its elements: $V^{\max}(P) = \max_{p \in P} V^{\max}(p)$. For the empty set, $V^{\max}(\emptyset) = 0$. The value $V^{\max}(p)$ of a variable p is 0 if p is true in the initial state. Otherwise, it is the lowest estimated cost $C^{\max}(e)$ of any effect e that achieves (adds) it: $V^{\max}(p) = \min_{\{e \mid p \in add(e)\}} C^{\max}(e)$.

The cost $C^{\max}(e)$ of an effect e is the action cost plus the value $V^{\max}$ of all propositions that must be true for the effect to trigger: $C^{\max}(e) = cost(act(e)) + V^{\max}(cond(e) \cup pre(act(e)))$.[2]

The estimate of the maximum heuristic for the initial state is the value $V^{\max}$ of the goal: $h^{\max}(\Pi) = V^{\max}(G)$.

Another admissible heuristic which is also based on delete relaxation and dominates $h^{\max}$ is the *LM-Cut heuristic* $h^{\text{LM-Cut}}$ [6]. It relies on *disjunctive action landmarks* which are sets of actions of which at least one must occur in every plan. The LM-Cut heuristic is only defined for STRIPS tasks (without conditional effects).

To simplify the presentation, we assume in the following that the initial state consists of a single variable i and the goal of a single variable g. If the task does not have this form, we would introduce i and g as new variables and add a goal action (having the original goal as precondition and adding g) and an init action (requiring i, deleting i, and adding all variables from the original initial state), both with cost 0. We also require that every action has a precondition (if it is originally empty, we can add an artificial precondition).

The $h^{\text{LM-Cut}}$ computation works in rounds: based on the values $V^{\max}$, each round computes a disjunctive action landmark, accounts for its cost and adapts the task so that the result will be admissible:

Definition 1 (Round of LM-Cut for STRIPS) *Each round of the LM-Cut algorithm for STRIPS works as follows:*

1. *Compute $V^{\max}$ for all variables. If $V^{\max}(g) = 0$ then terminate.*
2. *Define a* precondition choice function *pcf that maps each action to one of its precondition variables with a maximal $V^{\max}$ value.*
3. *Create the weighted, directed graph $G = (V, E)$, where $V = F$ and E contains labeled edges for all actions a from the selected precondition to each add effect: $E = \{(pcf(a), a, v) \mid a \in A, v \in add(a)\}$. Each edge has weight $cost(a)$.*
   The goal zone $V_g \subseteq V$ *consists of all nodes from which one can reach the goal variable g via edges with weight 0. The* cut C *contains all edges (v, a, v') such that $v \notin V_g$, $v' \in V_g$ and v can be reached from i without traversing a node in V_g.*
   The landmark L consists of all actions that occur as a label in C.
4. *Add the cost c_{min} of the cheapest action in L to the heuristic value (which starts as 0).*
5. *Reduce the action costs of all actions in L by c_{min}.*

Helmert and Domshlak [6] call the graph G a *justification graph* of the current task because by the definition of the precondition choice function and its construction, the $h^{\max}$ value of a fact p is the cost of a cheapest (with respect to the edge weights) path from i to p. This is relevant for the proof that $h^{\text{LM-Cut}}$ dominates $h^{\max}$, so we will retain this property in our adaption to conditional effects.

[2] Strictly speaking, $V^{\max}$ is not well-defined in the presence of 0-cost actions. In this case, $V^{\max}$ is the pointwise maximal function that satisfies the given properties. A unique maximum always exists.

3 RUNNING EXAMPLE

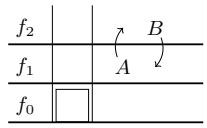


Figure 1: Running example.

Throughout the paper we use a running example (Figure 1), borrowed from Haslum [4, Example 1]. It is based on a delete-free variant of the Miconic domain,[3] where passengers are transported between floors by an elevator. In this small example there are three floors (f_0, f_1, f_2) and two passengers (A and B). Passenger A wants to go from f_1 to f_2 and passenger B from f_2 to f_1. The elevator starts at f_0. The possible actions are to stop at any floor f which causes all passengers who start at f to board and all boarded passengers with target f to disembark. This is implemented by conditional effects: Each action *stop-f* has a conditional effect $board(p) = \langle \emptyset, \{boarded(p)\}, \emptyset \rangle$ for each person p originated at f. The effect condition can stay empty because in the delete-relaxed variant it is irrelevant whether we "re-board" a passenger who has already been served. For each person who has f as destination floor, the *stop-f* action has a conditional effect $disembark(p) = \langle \{boarded(p)\}, \{served(p)\}, \emptyset \rangle$ that marks p as served if she was in the cabin. Both actions, *stop-f_1* and *stop-f_2*, have no preconditions and a cost of 1.

An optimal plan for the example is $\langle stop\text{-}f_1, stop\text{-}f_2, stop\text{-}f_1 \rangle$. At least one *stop* action must be used twice because the first application of such an action can only trigger the effect causing the passenger to board and not the one causing the other passenger to disembark.

4 LM-CUT FOR CONDITIONAL EFFECTS

We will now introduce a generic adaption of the LM-Cut algorithm to STRIPS^c tasks. As above, we assume that the input task has a single initial variable i and a single goal atom g. Moreover, we require without loss of generality that every conditional effect in the task only adds a single variable. If this is not the case, we can simply break up the conditional effect accordingly.

Since we still want to compute a justification graph in every round of the computation, we need to consider the effect conditions in the (pre-) condition choice function. It is also necessary that the cut in the graph distinguishes different conditional effects of an action.

Definition 2 (Generic Round of LM-Cut for STRIPS^c) *Each round of the LM-Cut algorithm for* STRIPS^c *works as follows:*

1. *Compute the $V^{\max}$ values for all variables. If $V^{\max}(g) = 0$ then terminate.*
2. *Define a* condition choice function *ccf that maps each effect to a fact from the effect condition or its action's precondition that has a maximal $V^{\max}$ value.*
3. *Create the justification graph $G = (V, E)$, where $V = F$ and E contains edges for all conditional effects e from the selected condition to the single add effect of e (labeled with e). Each edge has weight $cost(act(e))$. The goal zone V_g and the cut C are defined as in the standard STRIPS case. The landmark L consist of all actions of which an effect occurs as a label in C.*

[3] Compared to the domain reported in our experiments, there are no *move* actions and the stop action is delete-free, to get a simpler example.

4. *Add the cost c_{min} of the cheapest action in L to the heuristic value (which starts as 0).*
5. *Adapt the task.*

In our example, the generic LM-Cut algorithm would calculate a $V^{\max}$ value of 1 for each *boarded*(p) fact and a $V^{\max}$ value of 2 for each *served*(p) fact and the artificial goal fact g. The condition choice function would select one *served*(p) fact arbitrarily. Let us assume it selects *served*(A). The resulting justification graph is shown in Figure 2a (the continuation in Figures 2b and 2c belongs to a later example). The only effect achieving *served*(A) is *disembark*(A), which will be the only effect in the cut. It belongs to the action *stop*-f_2, so we have $L = \{$*stop*-$f_2\}$ and $c_{min} = 1$.

The open question here is how to adapt the task. The most obvious way would be to apply the same strategy as in the STRIPS case and to reduce the costs of all actions in L. We denote this instantiation of the algorithm by $h^{\text{LM-Cut}}_{\text{basic}}$.

With this strategy *stop*-f_2 is free of cost after the first round in our example. In the second round the $V^{\max}$ value of both *served*(p) facts is 1 and one is selected arbitrarily by the condition choice function. The discovered landmark is either $\{$*board*$(A)\}$ or $\{$*disembark*$(B)\}$ depending on this choice, but in both cases the cost of *stop*-f_1 is reduced next. After this round both *stop* actions are free of cost, the $V^{\max}$ value of the goal becomes 0, and the LM-Cut algorithm terminates with a heuristic value of 2. In this example, the $h^{\text{LM-Cut}}_{\text{basic}}$ estimate is still as high as $V^{\max}(g)$ but this is not guaranteed in general. Keyder et al. [7] showed that $h^{\text{LM-Cut}}_{\text{basic}}$ does not dominate $h^{\max}$ with an example task Π for which $h^{\text{LM-Cut}}_{\text{basic}}(\Pi) < h^{\max}(\Pi)$.

They also considered a strategy where each conditional effect is treated separately and showed that this leads to an inadmissible heuristic. With this strategy LM-Cut would run for 4 rounds in our example. It discovers the landmarks $\{$*disembark*$(A)\}$, $\{$*disembark*$(B)\}$, $\{$*board*$(A)\}$, and $\{$*board*$(B)\}$ in an order that depends on the condition choice function. The heuristic value of 4 is inadmissible because increasing the heuristic value by 1 for each of these landmarks ignores the fact that two effects can be achieved with one action application. For example, *board*(B) and *disembark*(A) can be achieved by *stop*-f_2 if *stop*-f_1 was executed before.

In the following sections, we will show how one can adapt the task without sacrificing either admissibility or dominance over $h^{\max}$.

5 CONTEXT SPLITTING

Before we present the adaption specifically for the LM-Cut heuristic, we would like to introduce *context splitting* as a new general concept. For this, we briefly consider the more general ADL formalism.

Actions behave differently if they are applied in different scenarios (e. g., a conditional effect triggers only if the effect condition is true). The core idea of context splitting is that we can include such scenarios in the action preconditions, splitting up an action into several ones with disjoint scenarios. An extreme case of this general idea is the compilation from STRIPS^c to STRIPS by Nebel [9]. For each action, it introduces new actions for each possible subset of effects and adds a corresponding condition to the action precondition.

However, such scenario information can also be useful for heuristic computations: if we account for an action application in a heuristic computation, we often know that some desired effects only trigger in a certain scenario. If the action has other required effects that do not trigger at the same time, we could account for its cost again for a later application of the action.

In general, a context split is defined by the description of a scenario. Such a description is given as a propositional formula over the task variables, which we call the *context*. If we split an action with a context, we introduce two new actions, one requiring the context to be true, the other one requiring it to be false.

Definition 3 (Context splitting) *A* context *is a propositional formula.* Context-splitting *an action a with context φ means replacing a with two new actions of the same cost: $a_\varphi = \langle pre(a) \wedge \varphi, eff(a)\rangle$ and $a_{\neg\varphi} = \langle pre(a) \wedge \neg\varphi, eff(a)\rangle$.*

Context splitting is a task transformation that does not affect the optimal goal distance of any state:

Theorem 1 *Let Π be an* ADL *planning task with action set A. For action $a \in A$ and context φ, let a_φ and $a_{\neg\varphi}$ be the two new actions resulting from context-splitting a with φ. Let Π' denote the task that only differs from Π in its action set $A' = (A \setminus \{a\}) \cup \{a_\varphi, a_{\neg\varphi}\}$.*

For all states s of Π (and Π') it holds that $h^*_\Pi(s) = h^*_{\Pi'}(s)$.

Proof: We can associate every plan π for s in Π with a plan π' for s in Π' of the same cost and vice versa.

From π' to π, we simply replace every occurrence of an action a_φ or $a_{\neg\varphi}$ with the original action a. This is possible because these actions only differ in the precondition and $pre(a_\varphi) \models pre(a)$ and $pre(a_{\neg\varphi}) \models pre(a)$.

From π to π' we check for every occurrence of a if φ is true in the state s' in which action a is applied. If yes, we replace a with a_φ, otherwise we replace it with $a_{\neg\varphi}$. These actions will be applicable and have the same effect and cost as the original action a. ∎

The theorem ensures that an admissible heuristic estimate for the transformed task is also an admissible estimate for the original task.

6 RELAXED CONTEXT SPLITTING

The key idea of our adaption of the LM-Cut heuristic is to reduce action costs only where necessary. After discovering the landmark $\{$*disembark*$(A)\}$ in our example we would like to reduce the cost of *stop*-f_2 whenever it is used in a way that this effect triggers. If we stick to the original actions, however, we can only reduce the cost of the whole action, i. e., also in situations where the effect does not trigger because A has not boarded yet. Another way of looking at this is that we can reduce the cost of the original actions at most twice before all actions are free of cost, so the heuristic value can be at most 2 when no actions are modified. This is where context splitting comes into play.

The context for each action should capture all situations in which the LM-Cut heuristic accounts for its cost. This is the case whenever one of its effects occurs as a label in the cut C. So we need to formulate a context that covers all situations in which one of the effects in the cut triggers. This leads to the natural definition of the context as

$$\varphi_a = \bigvee_{(v,e,v') \in C \text{ with } act(e)=a} cond(e).$$

If we split all actions in the LM-Cut landmark L with their respective context, the set of actions $\{a_{\varphi_a} \mid a \in L\}$ will be a landmark of the modified task. So we can admissibly count the landmark cost, reduce the cost of all a_{φ_a}, leave the cost of all $a_{\neg\varphi_a}$ unchanged, and proceed.

However, this idea cannot be implemented directly because we leave the STRIPS^c formalism with the context splitting. To see this, consider a context split of action a.

The precondition of the first new action a_{φ_a} is of the form $\mathit{pre}(a) \wedge (\mathit{cond}(e_1) \vee \cdots \vee \mathit{cond}(e_n))$ for some conditional effects $e_1, \ldots, e_n \in \mathit{eff}(a)$. Since the precondition $\mathit{pre}(a)$ and the effect conditions are all conjunctions of atoms, we can break up the action into n new STRIPS^c actions $a^e_{\varphi_a} = \langle \mathit{pre}(a) \wedge \mathit{cond}(e), \mathit{eff}(a)\rangle$ for $e \in \{e_1, \ldots, e_n\}$. Whenever a plan contains an action a_{φ_a}, there would also be a plan using an action $a^e_{\varphi_a}$ instead and vice versa.

The problem arises from the second new action $a_{\neg\varphi_a}$ whose precondition in general cannot be expressed as a negation-free formula. So we cannot easily reformulate these actions in STRIPS^c as we did with the actions a_{φ_a}.

As a solution, we propose *relaxed context splitting* which ignores the condition $\neg\varphi_a$ and simply preserves the original action:

Definition 4 (Relaxed Context Splitting) *The* relaxed context splitting *of an action a with context φ adds a new action $a_\varphi = \langle \mathit{pre}(a) \wedge \varphi, \mathit{eff}(a)\rangle$ with cost $c(a)$ to the task.*

Like unrelaxed context splitting, relaxed context splitting preserves the goal distance of states. It also preserves the value $V^{\max}$ of all variables: in general, adding actions to a task can only lead to a decrease of $V^{\max}$. However, in this case a decrease cannot happen: the new actions have the same effects and costs as the original ones but their precondition is a superset of the original precondition. Therefore the cost $C^{\max}$ of the effects of the new action cannot be lower than the one of the original effects, so no variable can be achieved more cheaply.

Unfortunately, with *relaxed* context splitting the set of actions $\{a_{\varphi_a} \mid a \in L\}$ is not a landmark of the modified task because a plan could contain action $a \in L$ instead of a_{φ_a}. So we cannot obviously apply the cost adaption as proposed at the beginning of this section. In the next section we will show that we still can define an extension to LM-Cut based on relaxed context splitting that preserves the desired properties of the heuristic.

7 LM-CUT WITH RELAXED CONTEXT SPLITTING

The key insight of our proposed heuristic is that we can safely leave the cost of all actions unchanged in each round of the LM-Cut computation as long as we add new reduced-cost actions that "fit" the context of the cut.

Definition 5 (LM-Cut heuristic with relaxed context splitting) *The* LM-Cut heuristic with relaxed context splitting *($h^{\text{LM-Cut}}_{\text{context}}$) instantiates the generic heuristic from Definition 2. In the task adaption step, for every edge $(v, e, v') \in C$ it extends the task with an action $a_e = \langle \mathit{pre}(a) \cup \mathit{cond}(e), \mathit{eff}(a)\rangle$ with $\mathit{cost}(a_e) = \mathit{cost}(a) - c_{\mathit{min}}$, where $a = \mathit{act}(e)$.*

In our example, we discover the landmark $\{\mathit{disembark}(A)\}$ in the first round (Figure 2a). Since there is only one effect in the cut, the disjunction in the context collapses to a single condition $\varphi_{\mathit{stop}\text{-}f_2} = \mathit{cond}(\mathit{disembark}(A)) = \mathit{boarded}(A)$. With relaxed context splitting we create the new action $\mathit{stop}\text{-}f_2{}' = \mathit{stop}\text{-}f2_{\mathit{disembark}(A)}$ with the additional precondition $\mathit{boarded}(A)$ and the reduced cost 0.

In the next round (Figure 2b) we discover the landmark $\{\mathit{disembark}(B)\}$, which is handled just like in the first round and we add the action $\mathit{stop}\text{-}f_1{}' = \mathit{stop}\text{-}f1_{\mathit{disembark}(B)}$ with the additional precondition $\mathit{boarded}(B)$ and the reduced cost 0.

In the final round (Figure 2c) the values $V^{\max}$ of all $\mathit{boarded}(p)$ and $\mathit{served}(p)$ facts and g are 1. The discovered landmark consists

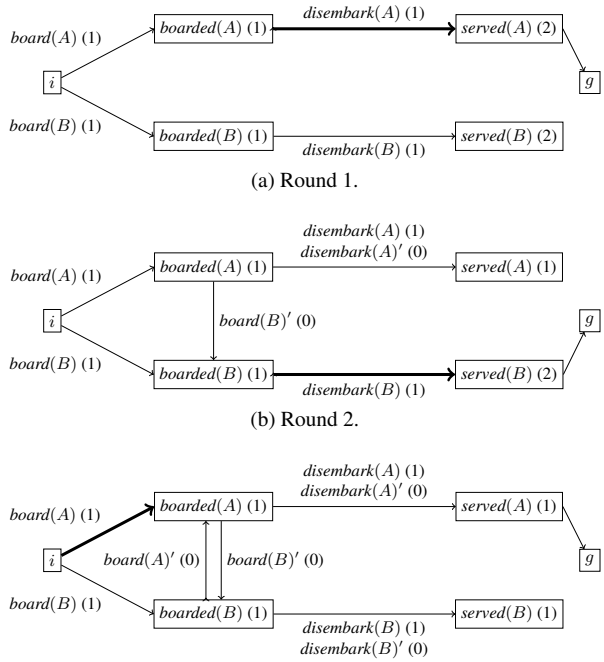


Figure 2: Justification graphs in the LM-Cut rounds for $h^{\text{LM-Cut}}_{\text{context}}$ on the example task. Action costs for effects and $V^{\max}$ values for facts are given in parentheses, edges in the cut are bold.

of a single *board*-effect. Which of the two is chosen depends on the condition choice function, but we assume that $\mathit{board}(A)$ is selected. Since this effect has no condition, the context is $\varphi_{\mathit{stop}\text{-}f_1} = \mathit{cond}(\mathit{board}(A)) = \top$ and the newly added action $\mathit{stop}\text{-}f_1{}''$ is identical to $\mathit{stop}\text{-}f_1$, except that it is free of cost.

With this new action, the $V^{\max}$ value of all facts now is 0. In particular, $\mathit{boarded}(B)$ can be reached from $\mathit{boarded}(A)$ with action $\mathit{stop}\text{-}f_2{}'$ without additional cost. The LM-Cut algorithm stops with a perfect heuristic value of 3.

In the following, we will show that $h^{\text{LM-Cut}}_{\text{context}}$ is admissible and dominates $h^{\max}$.

Theorem 2 *The LM-Cut heuristic with relaxed context splitting ($h^{\text{LM-Cut}}_{\text{context}}$) is* admissible.

Proof: We will show that the optimal delete-relaxation heuristic h^+ dominates $h^{\text{LM-Cut}}_{\text{context}}$. Since h^+ is admissible, we can conclude that $h^{\text{LM-Cut}}_{\text{context}}$ is also admissible.

If $h^{\max}(\Pi) = 0$, the LM-Cut algorithm directly terminates with $h^{\text{LM-Cut}}(\Pi) = 0$, so there is nothing to show in this case. Otherwise, let Π and Π' be the (relaxed) tasks before and after a round of $h^{\text{LM-Cut}}_{\text{context}}$, respectively. We will show that $h^+(\Pi) \geq c_{\mathit{min}} + h^+(\Pi')$. The dominance of h^+ then follows from an inductive application of this argument.

Every atom (except i) of the task Π can only be made true by an effect of an incoming edge in the justification graph and this effect only triggers if the source of the edge has been true. So any plan of Π must use all action effects of some path from i to g in the justification graph and therefore also at least one effect from the cut.

Let $\pi = \langle a_1, \ldots, a_n\rangle$ be an optimal plan for Π and let a_i be the first action in this plan whose application triggers an effect e from the cut. Π' has an action $a'_i = \langle \mathit{pre}(a_i) \cup \mathit{cond}(e), \mathit{eff}(a_i)\rangle$ with cost $c(a_i) - c_{\mathit{min}}$. Since e triggers in π, $\mathit{pre}(a_i) \cup \mathit{cond}(e)$ must be true

after the application of $\langle a_1, \ldots, a_{i-1} \rangle$. As a_i and a'_i have the same effect, $\pi' = \langle a_1, \ldots, a_{i-1}, a'_i, a_{i+1}, \ldots, a_n \rangle$ is a plan for Π' that costs c_{min} less than π and therefore $h^+(\Pi') \leq h^+(\Pi) - c_{min}$. ∎

The new heuristic is more informed than the maximum heuristic:

Theorem 3 *The LM-Cut heuristic with relaxed context splitting ($h^{\text{LM-Cut}}_{\text{context}}$) dominates $h^{\max}$.*

Proof: To increase clarity, in the following we denote the $V^{\max}$ value of a variable v in a task Π by $V^{\max}_{\Pi}(v)$.

If $h^{\max}(\Pi) = 0$ there is nothing to show. If $h^{\max}(\Pi) > 0$, we again denote the original (relaxed) task by Π and the transformed one after one LM-Cut round by Π'. We show that $h^{\max}(\Pi) \leq c_{min} + h^{\max}(\Pi')$. An inductive application of this argument proves the theorem.

Let A and A' denote the action sets of Π and Π', respectively. Consider the standard algorithm for computing $V^{\max}$: it uses a priority queue, initially containing the initial facts with a priority 0. The algorithm successively pops a fact with minimal priority from the queue and assigns it the priority as value $V^{\max}$ if the fact has not already been popped before. Whenever all relevant conditions of an effect e have been popped, the algorithm enqueues its added fact f with priority $C^{\max}(e)$.

Let $f' \in F$ be the first fact which is popped during the $V^{\max}_{\Pi'}$ computation that gets assigned a value $V^{\max}_{\Pi'}(f') < V^{\max}_{\Pi}(f')$, if such a fact exists. If g is popped before f' or no such fact f' exists, then $h^{\max}(\Pi) = V^{\max}_{\Pi}(g) = V^{\max}_{\Pi'}(g) = h^{\max}(\Pi')$ and there is nothing to show. In the following, we assume that g is popped after f' and hence $h^{\max}(\Pi') = V^{\max}_{\Pi'}(g) \geq V^{\max}_{\Pi'}(f')$.

Let e' be the effect due to which f' had been enqueued. Then e' must be an effect of some newly added action $a' \in A' \setminus A$: since f' is the first value with a differing $V^{\max}$, the change cannot be due to "cheaper" condition costs.

The action a' must have been added because an effect e (of an action a) occurred in the cut. Therefore, $a' = \langle pre(a) \cup cond(e), eff(a) \rangle$ with cost $cost(a') = cost(a) - c_{min}$ for some action a of Π and effect e of a. Let f be the fact added by e.

Since e was in the cut, f must have been in the goal zone and therefore it holds that $V^{\max}_{\Pi}(pre(a) \cup cond(e)) + cost(a) \geq V^{\max}_{\Pi}(f) \geq V^{\max}_{\Pi}(g) = h^{\max}(\Pi)$ (*).

We can bound $h^{\max}(\Pi)$ as follows:

$$
\begin{aligned}
h^{\max}(\Pi) &\leq V^{\max}_{\Pi}(pre(a) \cup cond(e)) + cost(a) && (1)\\
&= V^{\max}_{\Pi'}(pre(a) \cup cond(e)) + cost(a) && (2)\\
&\leq V^{\max}_{\Pi'}(pre(a) \cup cond(e) \cup cond(e')) + cost(a) && (3)\\
&= V^{\max}_{\Pi'}(f') + c_{min} && (4)
\end{aligned}
$$

Statement (1) uses the previously derived bound (*). Equation (2) holds as $pre(a) \cup cond(e)$ is the precondition of a' and hence all facts in this set must have been popped before f' was enqueued by effect e'. Since f' is the first popped fact for which $V^{\max}_{\Pi} \neq V^{\max}_{\Pi'}$ it follows for all $p \in pre(a) \cup cond(e)$ that $V^{\max}_{\Pi}(p) = V^{\max}_{\Pi'}(p)$. Inequality (3) is due to $V^{\max}(P) \leq V^{\max}(P')$ if $P \subseteq P'$. The last line exploits that effect e' of action a' establishes the value $V^{\max}_{\Pi'}(f')$ and that $cost(a') = cost(a) - c_{min}$.

Overall we have shown that $h^{\max}(\Pi) \leq V^{\max}_{\Pi'}(f') + c_{min}$. Since we know from above that $h^{\max}(\Pi') \geq V^{\max}_{\Pi'}(f')$, it holds that $h^{\max}(\Pi) \leq h^{\max}(\Pi') + c_{min}$. ∎

We have seen that $h^{\text{LM-Cut}}_{\text{context}}$ preserves the desired properties of the LM-Cut heuristic for STRIPS. In the next section we will evaluate whether it also preserves its good performance.

8 EXPERIMENTAL EVALUATION

For the evaluation we use the same sets of domains T0 and FSC as Haslum [4]. The T0 domains are generated by a compilation from conformant to classical planning by Palacios and Geffner [10]; the set FSC has been generated by the finite-state controller synthesis compilation by Bonet et al. [2]. In addition, we include tasks from the briefcase world from the IPP benchmark collection [8]. We also use the Miconic Simple-ADL version from the benchmark set of the International Planning Competition (IPC-2000) because it has conditional effects but no derived predicates after grounding with Fast Downward.

We compare $h^{\max}$ and three variants of the LM-Cut heuristic:

- our version $h^{\text{LM-Cut}}_{\text{context}}$ using relaxed context splitting,
- the version $h^{\text{LM-Cut}}_{\text{basic}}$ mentioned by Keyder et al. [7] that reduces the action cost of every action with an effect in the cut and does not dominate $h^{\max}$, and
- the standard LM-Cut version $h^{\text{LM-Cut}}_{\text{standard}}$ [6], which does not support conditional effects. For this variant, we transform the tasks with the exponential compilation by Nebel [9].

All heuristics were implemented in the Fast Downward planning system [5], which separates the preprocessing phase from the actual search phase. For each phase, we set a time limit of 30 minutes and a memory limit of 2 GB per task. The experiments were conducted on Intel Xeon E5-2660 processors (2.2 GHz).

We first compare the two LM-Cut versions that support conditional effects directly.

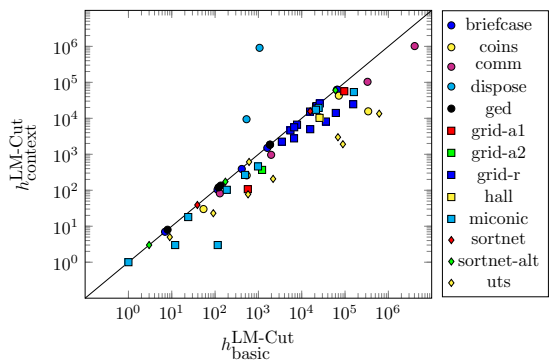
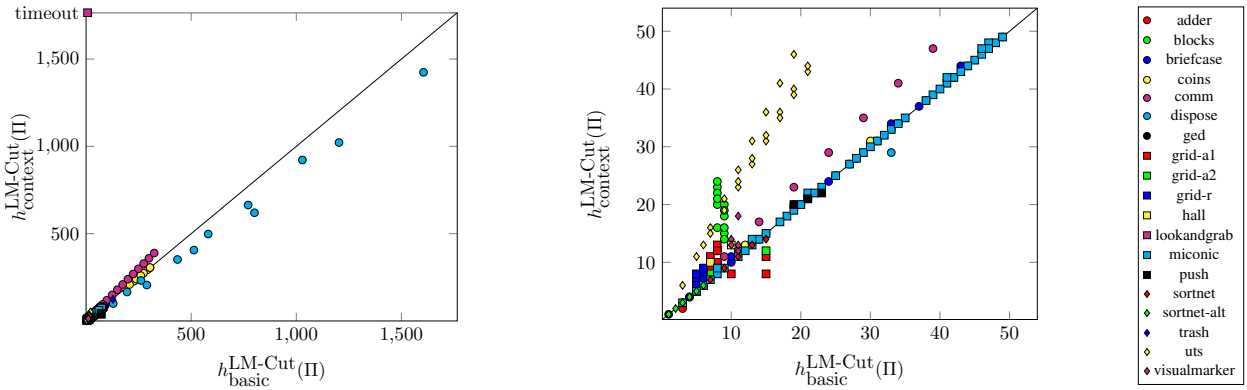


Figure 3: Number of expansions (excluding the ones on the last f-layer) of $h^{\text{LM-Cut}}_{\text{basic}}$ and $h^{\text{LM-Cut}}_{\text{context}}$ for the commonly solved tasks.

Figure 3 plots the number of A* expansions of $h^{\text{LM-Cut}}_{\text{basic}}$ vs. those of $h^{\text{LM-Cut}}_{\text{context}}$ for each task. As expected, context splitting almost always gives equal or better guidance than the basic approach. The only exception is the t0-grid-dispose domain in which $h^{\text{LM-Cut}}_{\text{basic}}$ is superior.

To get a clearer idea of the difference of the heuristic estimates, we compare the heuristic values of the initial states in Figure 4. The very high estimates in the t0-grid-dispose domain render the results of the other tasks almost indistinguishable. For this reason, Figure 4b shows the same results but only includes tasks where both heuristic estimates are below 50. Overall, we note that the estimates of $h^{\text{LM-Cut}}_{\text{context}}$ are much better than those of $h^{\text{LM-Cut}}_{\text{basic}}$ and in the t0-uts domain they are always at least twice as high.

Since the results of the t0-grid-dispose domain stick out negatively, we had a closer look at this domain to understand the different performance. A deeper analysis of one task reveals that the variant with relaxed context splitting makes unfavorable decisions when selecting one of several candidates with maximal $V^{\max}$ for



(a) All values. (b) Heuristic values below 50.

Figure 4: Heuristic values of the initial state for $h^{\text{LM-Cut}}_{\text{basic}}$ and $h^{\text{LM-Cut}}_{\text{context}}$.

the condition choice function. As a result, effects that achieve different sub-goals end up in one cut, and they all become cheaper in the next round. A similar effect can also be observed with $h^{\text{LM-Cut}}_{\text{standard}}$ in the STRIPS freecell domain.

Table 1: Coverage results. Best results in each domain in bold text.

	$h^{\text{LM-Cut}}_{\text{standard}}$	$h^{\max}$	$h^{\text{LM-Cut}}_{\text{basic}}$	$h^{\text{LM-Cut}}_{\text{context}}$
briefcaseworld (9,50)	6	7	**9**	8
fsc-grid-a1 (0,16)	-	**2**	**2**	**2**
fsc-grid-a2 (0,2)	-	**1**	**1**	**1**
fsc-grid-r (0,16)	-	**15**	**15**	13
fsc-hall (0,2)	-	**1**	**1**	**1**
gedp-ds2ndp (0,24)	-	**18**	12	12
miconic(149,150)	78	70	**141**	**141**
t0-coins (20,30)	**14**	10	**14**	**14**
t0-comm (25,25)	**5**	4	**5**	**5**
t0-grid-dispose (0,15)	-	0	**3**	2
t0-grid-lookandgrab (0,1)	-	**1**	**1**	0
t0-sortnet (0,5)	-	**2**	**2**	**2**
t0-sortnet-alt (1,6)	1	**4**	**4**	**4**
t0-uts (6,29)	5	6	8	**10**
Sum (210,371)	109	141	**218**	215

Table 1 shows the number of solved instances for all heuristics (omitting domains where no task was solved by any heuristic). Note that $h^{\text{LM-Cut}}_{\text{standard}}$ cannot be directly compared to the other heuristics based on these numbers because it requires a compilation of the task that removes conditional effects. The small numbers behind the domain names state for how many tasks the Fast Downward preprocessing phase completed with and without the compilation. It is apparent that–at least with the exponential transformation–compiling away conditional effects and using a standard heuristic is not competitive.

Except for the Miconic domain, which dominates the summary results with its large number of tasks, the three remaining heuristics are surprisingly close to each other and each one is better than the others in some domain. While $h^{\max}$ performs worst as expected, the better guidance of $h^{\text{LM-Cut}}_{\text{context}}$ does not translate to higher coverage than $h^{\text{LM-Cut}}_{\text{basic}}$ because it does not offset the additional time for the heuristic evaluations. However, in the conclusion we will explain how this might be resolvable in future work.

9 CONCLUSIONS AND FUTURE WORK

We presented an extension of the LM-Cut heuristic to conditional effects that is admissible and dominates the maximum heuristic. For this purpose we introduced context splitting as a new general concept of which we belief that it will prove useful also for other applications.

One obstacle for the new heuristic is that it adds many new actions in every round of its computation, which causes computational overhead in the following rounds. However, we hope that we can resolve this to some extent in future work: in certain respects, the computation of $h^{\text{LM-Cut}}_{\text{context}}$ is based on the individual conditional effects plus their action precondition. From this perspective, the context split adds many "equivalent" effects in every round. If it is possible to represent them only once (similar to the way it is done in an efficient $h^{\max}$ implementation), we expect a significant speed-up of the computation.

To avoid unfavorable selections of the condition choice function, it might be beneficial to deploy additional strategies, such as preferring conditions that were not added by a context split. As this paper focuses on the theoretical properties of the heuristics, we leave this topic for future work.

ACKNOWLEDGEMENTS

This work was supported by DFG grant HE 5919/2-1.

REFERENCES

[1] Blai Bonet and Héctor Geffner, 'Planning as heuristic search', *AIJ*, **129**(1), 5–33, (2001).
[2] Blai Bonet, Héctor Palacios, and Héctor Geffner, 'Automatic derivation of memoryless policies and finite-state controllers using classical planners', in *Proc. ICAPS 2009*, pp. 34–41, (2009).
[3] Tom Bylander, 'The computational complexity of propositional STRIPS planning', *AIJ*, **69**(1–2), 165–204, (1994).
[4] Patrik Haslum, 'Optimal delete-relaxed (and semi-relaxed) planning with conditional effects', in *Proc. IJCAI 2013*, pp. 2291–2297, (2013).
[5] Malte Helmert, 'The Fast Downward planning system', *JAIR*, **26**, 191–246, (2006).
[6] Malte Helmert and Carmel Domshlak, 'Landmarks, critical paths and abstractions: What's the difference anyway?', in *Proc. ICAPS 2009*, pp. 162–169, (2009).
[7] Emil Keyder, Jörg Hoffmann, and Patrik Haslum, 'Semi-relaxed plan heuristics', in *Proc. ICAPS 2012*, pp. 128–136, (2012).
[8] Jana Köhler, 'Handling of conditional effects and negative goals in IPP', Technical Report 128, Institute for Computer Science, Albert-Ludwigs-Universität, Freiburg, Germany, (1999).
[9] Bernhard Nebel, 'On the compilability and expressive power of propositional planning formalisms', *JAIR*, **12**, 271–315, (2000).
[10] Hector Palacios and Hector Geffner, 'Compiling uncertainty away in conformant planning problems with bounded width', *JAIR*, **35**, 623–675, (2009).