# Incremental Lower Bounds for Additive Cost Planning Problems

**Patrik Haslum** and **John Slaney** and **Sylvie Thiébaux**
Australian National University & NICTA
*firstname.lastname*@anu.edu.au

**Abstract**

We define an incremental lower bound function for additive cost planning, based on repeatedly solving and strengthening the delete-relaxation of the problem. We show that it is monotonically increasing, and thus in the limit will produce an optimal plan. In most domains, however, it is not effective compared to alternatives such as limited search.

## Introduction

Many potential applications of planning require planners to produce plans of high quality, according to a metric like cost, makespan, net benefit, or other. Even when generating guaranteed optimal plans is not computationally feasible, there is a need to be able to measure, in absolute terms, how good the plans found by non-optimal planners are. Current planning technology does not offer many tools suitable for this purpose. A lower bound function (i.e., an admissible heuristic) gives an absolute assessment, since the optimal cost is known to lie between the lower bound and the cost of the best known plan. But if this gap is too large to give confidence in the quality of the solution, and further search fails to turn up a better plan, there is not much that can be done. What is needed in this situation is an incremental lower bound: a function that can produce increasingly higher admissible estimates, given more time and memory.

Given any admissible heuristic $h$ (including the "blind" heuristic $h = 0$) such an incremental lower bound can be obtained by running A*, IDA*, or any search algorithm that approaches the optimum from below, with $h$ for a limited time, taking the highest proven $f$-value. But this is not always effective. As an example in point, columns (b) and (c) in table 1 shows, for a set of problems, the value of the LM-Cut heuristic (Helmert and Domshlak 2009) in the initial state, and the highest lower bound proven by running A* with this heuristic (Fast Downward implementation) until it exhausts memory. In only one problem does this result in a bound higher than the initial, and in no problem does the search uncover a plan.

In this paper, we propose a different approach to incremental lower bound computation, by repeatedly finding optimal solutions to relaxations of the planning problem, which are increasingly less and less relaxed. If we find a relaxed plan that is also a plan for the original, unrelaxed problem, we know it is optimal. If not, we use clues from the failure of the relaxed plan to construct the next relaxation, in such a way that the same relaxed plan will not be found again. The relaxation we consider is the standard delete-relaxation, in which negative effects of actions are ignored. (This means that we solve the NP-hard problem of finding an optimal delete-relaxed plan in each iteration.) The delete-relaxation has the property that a plan achieving atoms $p$ and $q$ also achieves the conjunction $\{p, q\}$. The way in which we make it less relaxed is by constructing a modified problem, in which certain conjunctions are made explicit, in the form of new atoms.[1] This carries information about negative interactions between subgoals (achievement of conjuncts) into

| | (a) | (b) | (c) | (d) | (e) | (f) |
|---|---|---|---|---|---|---|
| chunk-089 | 137 | 104 | | | 104 | 108 |
| chunk-091 | 12 | 10 | | 12⋆ | 10 | 12⋆ |
| window-331 | 42 | 31 | | | 31 | 32⋆ |
| window-332 | 52 | 40 | | | 40 | 42 |
| window-333 | 68 | 47 | | | 47 | 50 |
| window-334 | 68 | 47 | | | 47 | 50 |
| window-335 | 63 | 45 | | | 45 | 48 |
| window-336 | 42 | 35 | | | 35 | 37 |
| window-337 | 32 | 27 | | | 27 | 29 |
| window-338 | 25 | 22 | 23 | | 22 | 24 |
| window-339 | 25 | 22 | | | 22 | 23 |
| window-340 | 25 | 21 | | | 21 | 22 |
| window-341 | 22 | 20 | | | 20 | 21 |
| window-409 | 10 | 9 | | 10⋆ | 9 | 10⋆ |

Table 1: Comparison of lower bounds on problems from a PDDL encoding of DES diagnosis. (All systems run with a memory limit of 3Gb, and no time limit.)
(a) Cost of best solution found by a non-optimal planner.
(b) LM-Cut-value of the initial state.
(c) Highest lower bound ($f$-value) proven by A* search with LM-Cut (where higher than (b)).
(d) Optimal solution costs found by Gamer (Edelkamp and Kissmann 2008).
(e) $h^+$-value of the initial state.
(f) Highest lower bound proven by $h^{++}$ A star indicates that the bound is optimal, i.e., that the final relaxed plan is valid.

[1] It is basically the $P^m_\star$ construction of Haslum (2009), applied to a select set of conjunctions instead of all of size $m$.

the delete-relaxation of the modified problem. We choose the conjunctions to encode by analysing the failure of the current relaxed plan, in a way that guarantees we find a different relaxed plan in the next iteration. This incremental lower bound function is tentatively named $h^{++}$.

Columns (e) and (f) in table 1 show the $h^+$ and $h^{++}$ values computed, respectively. As can be seen, it achieves better bounds than A*/LM-Cut, and in a few cases even finds an optimal solution. It must, however, be noted that this is an exception rather than a rule. We have found few other domains where $h^{++}$ is as effective, compared to the A*/LM-Cut combination. (Two examples will be presented later.)

## Related Work

The idea of incrementally refining relaxations is by no means new. It is widely used in optimisation algorithms, for example in the form of incremental generation of valid cuts in integer-linear programming (e.g. Cornuéjols 2008). An instance of the idea that is closer to planning is *counterexample-guided abstraction refinement*, in which the relaxation is an abstraction of the problem. An optimal solution (plan) for the abstract problem is checked for failures wrt to the parts of the original problem ignored by the current abstraction, and those parts that cause it to fail are candidates for inclusion in the next abstraction. This idea has been applied in verification (model checking), and to planning in adversarial, probabilistic domains (Chatterjee et al. 2005), but, as far as we are aware, not to deriving lower bounds for planning with additive cost. The most closely related instance of the idea is perhaps in the work of van den Briel et al. (2007), who formulate a relaxation of planning problems as an integer programming model of a flow problem. (The problem is further simplified by solving the LP relaxation of the IP.) It is a relaxation because certain ordering constraints, due to non-deleted action preconditions, are ignored. They use composition of state variables to refine the relaxation, though not in an incremental fashion.[2]

## Background

We adopt the standard definition of a propositional STRIPS planning problem, without negation in action preconditions or the goal (see, e.g., Ghallab, Nau, and Traverso 2004, chapter 2). We assume that action effects are consistent, meaning that for each action $a$, $\mathrm{del}(a) \cap \mathrm{add}(a) = \emptyset$.[3] As usual, a sequence of actions (or *plan*) *achieves* condition $c$ from state $s$ iff the sequence is executable in $s$ and leads to a state where $c$ holds. We assume an additive cost objective, i.e., each action $a$ has a non-negative cost, $\mathrm{cost}(a)$, and the cost of a plan is the sum of the cost of actions in it. The initial state is denoted by $s_I$ and the goal by $G$.

### The Delete-Relaxation

The delete-relaxation of a planning problem $P$, denoted $P^+$, is a problem exactly like $P$ except that $\mathrm{del}(a) = \emptyset$ for each $a$, i.e., no action makes any atom false. The delete-relaxation heuristic, $h^+(s, c)$, is defined as the minimum cost of any plan achieving $c$ from $s$ in the delete-relaxed problem.

Let $A$ be a set of actions in $P$. We denote by $R^+(A)$ the set of all atoms that are reachable in $P^+$, starting from the initial state, using only actions in $A$. We say a set of actions $A$ is a *relaxed plan* iff the goal $G \subseteq R^+(A)$. An actual plan for the delete-relaxed problem is of course a sequence of actions. That $G \subseteq R^+(A)$ means there is at least one sequencing of the the actions in A that reaches a goal state. When we need to distinguish a particular sequence of actions, that is valid a plan for $P^+$, we will call it a *sequenced relaxed plan*, or a *(relaxed) valid sequencing of* $A$.

We assume throughout that $G \subseteq R^+(A)$ for some set of actions $A$ (which may be the set of all actions in $P^+$), i.e., that the goal is relaxed reachable. If it is not, the problem is unsolvable and we already have the highest possible lower bound $\infty$.

## Computing Minimum-Cost Relaxed Plans

Our lower bound procedure depends on being able to compute a minimum-cost relaxed plan for a planning problem. It does not matter, for our purposes, how this is done. We would of course like to do it as efficiently as possible, but as solving $P^+$ optimally is NP-complete, we cannot expect any method to be efficient in general. A possible approach is to treat $P^+$ as a planning problem, like any other, and solve it using any cost-optimal planning algorithm. Betz and Helmert (2009) present polynomial-time algorithms for a few specific planning domains.

We will, however, use an algorithm based on the correspondance between relaxed plans and disjunctive action landmarks, established by Bonet and Helmert (2010). The algorithm is iterative in nature, and inspired by Slaney and Thiébaux's (2001) optimal planning algorithm for the Blocksworld domain.[4]

### The Iterative Landmark-Based Algorithm

A *disjunctive action landmark* (landmark, for short) of a problem $P$ is a set of actions such that at least one action in the set must be included in any valid plan for $P$. For any collection of landmarks $L$ for $P$, the set of actions in any valid plan for $P$ is a hitting set for $L$, i.e., contains one action from each set in $L$ (by definition).

For any set of actions $A$ such that $G \not\subseteq R^+(A)$, the complement of $A$ (w.r.t. the whole set of actions) is a disjunctive action landmark for $P^+$. If $A$ is a maximal (w.r.t. set inclusion) such "relaxed non-plan", the landmark is minimal

[2] An incremental refinement method has recently been developed for this relaxation, and used to create a quite effective cost-optimal planner. (Menkes van den Briel, pers. comm.)

[3] The semantics of PDDL permit actions that both delete and add the same atom. The net effect of such an action is to make the atom true. The transient delete effect implies that the action cannot be concurrent with any other action requiring or adding the atom, but since we consider only sequential (or sequentialisable) plans this is of no significance.

[4] Because we are only interested in optimal delete-relaxed plans, we also use a stronger form of relevance analysis to reduce the size of the problem: only actions that can achieve a relevant atom for the first time are relevant.

(w.r.t. set inclusion). This leads to the following algorithm for computing $h^+$:

Initialise the landmark collection $L$ to $\emptyset$. Repeatedly (1) find a minimum-cost hitting set $H$ for $L$; (2) test if $G \subseteq R^+(H)$; and if not, (3) extend $H$, by adding actions, to a maximal set $H'$ such that $G \not\subseteq R^+(H')$, and add the complement of $H'$ to the landmark collection.

The algorithm terminates when the hitting set $H$ passes the test in step (2). This set is by definition a relaxed plan. There is no lower-cost relaxed plan, because any plan for $P^+$ must contain an action from every landmark in $L$, and $H$ is a minimum-cost hitting set. Finally, the algorithm eventually terminates, because each landmark generated at step (3) is distinct from every landmark currently in $L$ (every landmark in $L$ contains at least one action in $H$, which the new landmark does not) and there is only a finite set of minimal landmarks for $P^+$. Two steps in this algorithm are frequently repeated, and therefore important to do efficiently: testing if the goal is relaxed reachable with a given set of actions, and finding a hitting set with minimum cost. Their implementations are detailed below. In spite of all tricks, however, computing an optimal relaxed plan is costly, in many problems criplingly so.

**Testing Relaxed Reachability** The set of atoms relaxed reachable with a given set of actions, i.e., $R^+(A)$, can be computed in linear time by a variant of Djikstra's algorithm:[5] Keep track of the number of unreached preconditions of each action, and keep a queue of newly reached atoms, initialised with the initially true atoms, dequeueing one at a time until the queue is empty. When dequeueing an atom, decrement the precondition counter for the actions that have this atom as a precondition, and when the counter reaches zero, mark atoms added by the action as reached and place any previously unmarked ones on the queue.

When generating a new landmark, we will perform a series of reachability computations, with mostly increasing sets of actions, starting from $H$, i.e., $H \cup \{a_1\}$, $H \cup \{a_1, a_2\}$, etc. Therefore, each reachability test (except the first) can be done incrementally. Suppose we have computed $R^+(A)$, by the algorithm above, and now wish to compute $R^+(A \cup \{a\})$. If $\mathrm{pre}(a) \not\subseteq R^+(A)$, $R^+(A \cup \{a\})$ equals $R^+(A)$, and the only thing that must be done is to initialise $a$'s counter of unreached preconditions. If not, mark and enqueue any previously unreached atoms in $\mathrm{add}(a)$, and resume the main loop until the queue is again empty. If the goal becomes reachable, we must remove the last added action ($a$) from the set, and thus must restore the earlier state of reachability. This can be done by saving the state of $R^+(A)$ (including precondition counters) before computing $R^+(A \cup \{a\})$, copying it back if needed, instead of recomputing $R^+(A)$ from scratch.

**Finding a Minimum-Cost Hitting Set** Finding a (provably) minimum-cost hitting set over the collection of landmarks $L$ is an NP-hard problem. We solve it using a recursive branch-and-bound algorithm with caching. Given a set $L = \{l_1, \ldots, l_m\}$ of landmarks to hit, pick a landmark $l_i \in L$: the minimum cost of a hitting set for $L$ is

$$H^\star(L) = \min_{a \in l_i} H^\star(L - \{l \mid a \in l\}) + \mathrm{cost}(a)$$

A lower bound on $H^\star(L)$ can be obtained by selecting any subset $L' \subset L$ such that $l \cap l' = \emptyset$ for any $l, l' \in L'$, i.e., a set of pair-wise disjoint landmarks, and summing the costs of their cheapest actions, i.e., $\sum_{l \in L'} \min_{a \in l} \mathrm{cost}(a)$. Finding the set $L'$ that yields the maximum lower bound amounts to solving a weighted independent set problem, but there are reasonably good and fast approximation algorithms (e.g. Halldórsson 2000).

Because the branch-and-bound algorithm is invoked recursively on subsets of $L$, it may prove increased lower bounds on the cost of hitting these subsets. Improved bounds are cached, and used in place of the lower bound calculation whenever a subset is encountered again. In the course of computing $h^+$, we will be solving a series of hitting set problems, over a strictly increasing collection of landmarks. Cached lower bounds may be used not only if a subset of $L$ is encountered again within the same search, but also if it is encountered while solving a subsequent hitting set problem. This makes the caching mechanism quite important.

When finding a hitting set for $L \cup \{l_{i+1}\}$, we have an optimal hitting set for $L$. $H^\star(L)$ is clearly a lower bound on $H^\star(L \cup \{l_{i+1}\})$, and an initial upper bound can be found by taking $H^\star(L) + \min_{a \in l_{i+1}} \mathrm{cost}(a)$. These bounds are often very tight, which limits the amount of search. In particular, if $l_{i+1}$ contains any zero-cost action, the initial upper bound is matched by the lower bound, and thus already optimal.

Finally, as long as the hitting set is not a relaxed plan, we do not require it to be optimal, since any hitting set will do as a starting point for generating another landmark. Thus, we can use any approximation algorithm to find a non-optimal hitting set, and invoke the optimal branch-and-bound procedure only when a relaxed plan is found. This scheme not always more efficient, however, because when the branch-and-bound algorithm is called, it is typically with a larger gap between the upper and lower bounds, and with fewer sets cached, resulting in more search.

## The Relaxed Plan Dependency Graph

Although we have defined a relaxed plan as an unordered set of actions, there are some necessary ordering relations between actions in this set. These, and the reasons for them, are captured by the *relaxed plan dependency graph* defined below. This graph plays a central role in the identification of conflicts in a failed relaxed plan.

Let $A$ be relaxed plan, i.e., a set of actions such that $G \subseteq R^+(A)$. We say that $A$ is *non-redundant* if no strict subset of $A$ is a relaxed plan. A relaxed plan can be checked for redundancy by simply testing for each action $a$ in turn whether $G \subseteq R^+(A - \{a\})$, and made non-redundant (though not necessarily in a unique way) by greedily removing actions found to be redundant. The landmark-based algorithm can produce optimal plans containing redundant actions, although naturally only when those actions have a cost of zero. From now on, we will assume all relaxed plans are non-redundant.

[5] To the best of our knowledge, this method of computing relaxed reachability was first implemented in Fast Downward, but has not been described in any publication.

**Definition 1** *Let $A$ be a non-redundant relaxed plan. Construct a graph $G'$ with nodes $\{n_a \mid a \in A\} \cup \{n_G\}$, i.e., the nodes of $G'$ correspond to actions in $A$ plus an additional node which represents the goal. With some abuse of notation, we write $\mathrm{pre}(n)$ for the precondition of node $n$, which is $\mathrm{pre}(a)$ for a node $n_a$ and $G$ for node $n_G$. $G'$ has an edge from $n_a$ to $n'$ iff $\mathrm{pre}(n') \not\subseteq R^+(A - \{a\})$. This edge is labelled with $\mathrm{pre}(n') - R^+(A - \{a\})$.*

*The relaxed plan dependency graph, $\mathrm{RPDG}(A)$, is the transitive reduction[6] of $G'$.*

As will be shown shortly, the graph $G'$ is acyclic, and therefore its transitive reduction is unique. (It is also computable in polynomial time; cf. Aho, Garey, and Ullman 1972.) Thus, the RPDG is well-defined.

Intuitively, an edge from node $n$ to node $n'$ in the RPDG means that the action associated with $n$ is necessary for $\mathrm{pre}(n')$ to be relaxed reachable, and the edge label documents why that is, i.e., which atoms in $\mathrm{pre}(n')$ are added by that action only. However, the RPDG does not capture disjunctive dependencies: if several actions in $A$ add atom $p$, there will be no edge with $p$ in its label, and the fact that at least one of those actions is necessary to reach $p$ will not be visible in the graph.

If there is a path from node $n$ to node $n'$ in $\mathrm{RPDG}(A)$, we say that the nodes are *ordered*. Conversely, if there is no path from $n$ to $n'$, nor from $n'$ to $n$, we say they are *unordered*. This terminology is justified by properties (2) and (3) in the theorem below.

**Theorem 2**
*(1) The graph $G'$ in definition 1 is acyclic, and hence so is $\mathrm{RPDG}(A)$.*
*(2) If there is a path from $n_a$ to $n_b$ in $\mathrm{RPDG}(A)$, $a$ appears before $b$ in every valid sequencing of $A$.*
*(3) Let $n_a$ and and $n_b$ be two unordered nodes, i.e., such that neither $n_a$ is reachable from $n_b$ nor $n_b$ from $n_a$ in $\mathrm{RPDG}(A)$. Then there exists a valid sequencing of $A$ in which $a$ appears before $b$.*
*(4) If atom $p$ appears in the label of an outgoing edge from node $n_a$ in $\mathrm{RPDG}(A)$, then $p \in \mathrm{add}(a)$.*
*(5) For any two action nodes $n_a$ and $n_b$ in $\mathrm{RPDG}(A)$, the labels of any pair of outgoing edges from $a$ and $b$, respectively, are disjoint.*
*(6) Any two unordered nodes $n$ and $n'$ in $\mathrm{RPDG}(A)$ have a common descendant, $n''$.*

**Proof:** (1) follows directly from property (2) and the fact that $A$ is non-redundant.

(2) Consider first the case where the path is a single edge from $n_a$ to $n_b$, and suppose that there is a valid sequencing with $b$ before $a$. This means that the preconditions of $b$ are reachable using only those actions that appear before $b$ in the sequence, which do not include $a$. This contradicts the fact that $\mathrm{pre}(b)$ is not contained in $R^+(A - \{a\})$. The general case follows by induction.

(3) Construct the sequence as follows: Let $A'$ be the set of all actions except $a$, $b$ and their descendants. Begin the sequence by applying all actions in $A'$ (in any valid order). After this, apply $a$, then $b$, then the remaining actions in any valid order. Suppose that this fails, because $\mathrm{pre}(a)$ does not hold after applying all actions in $A'$. Let $p \in \mathrm{pre}(a)$ be an unsatisfied precondition. Since $p$ does not hold initially (if it did, it would still hold after applying the actions in $A'$), there must be at least one action in $A$ with $p \in \mathrm{add}(a)$ which is not a descendant of $a$, and no such action is in $A'$. Thus, it must be $b$ or a descendant of $b$. But by definition this implies a path from $b$ to this action, and thus from $b$ to $a$.

(4) That $p$ belongs to the label of an edge from $n_a$ to some other node $n'$ means that $p \in \mathrm{pre}(n')$ and becomes unreachable without action $a$. This cannot be because $p$ is added by some other action, $b$, and $\mathrm{pre}(b)$ becomes unreachable without $a$, because if so, there would be edges from $n_a$ to $n_b$ and from $n_b$ to $n$, and thus the edge from $n_a$ to $n$ would not be in the transitive reduction. Hence, $p$ must be belong to $\mathrm{add}(a)$.

(5) Suppose there are two nodes, $n_a$ and $n_b$, in $\mathrm{RPDG}(A)$, that both have outgoing edges with labels that include $p$ (both must be action nodes, since the goal node has no outgoing edges). By property (4), $p$ must belong to both $\mathrm{add}(a)$ and $\mathrm{add}(b)$. If there is a path from $n_a$ to $n_b$, then $a$ appears before $b$ in any sequencing of $A$ (by property (2)), and removing $b$ cannot make $p$ unreachable. If $n_a$ and $n_b$ are unordered, either $a$ or $b$ can appear first in a valid sequencing (by property (3)). Thus, removing just one of $a$ or $b$ cannot make $p$ unreachable.

(6) Observe first that from every action node $n_a$ there is a path to the goal node $n_G$. Since the graph is acyclic, every path must end in a leaf (node with no outgoing edges). A leaf node that is not the goal node is a redundant action, since removing this action does not make the goal unreachable. Thus, any pair of unordered nodes have a common descendant, the goal node if no other. □

Note that property (3) holds only for pairs of nodes. The reason is that the RPDG does not capture disjunctive precedences. Suppose, for example, that actions $a$ and $b$ both add atom $p$, and that $p$ is a precondition of action $c$. ($a$ and $b$ both also add some other relevant atoms, as otherwise at least one of them would be redundant.) There are valid sequencings with $c$ before $a$ (if $c$ is preceded by $b$) and $c$ before $b$ (if preceded by $a$), but not with $c$ before both $a$ and $b$. Because of this, every valid sequencing of $A$ is a topological sort of $\mathrm{RPDG}(A)$, but not every topological sort of $\mathrm{RPDG}(A)$ is a valid sequencing of $A$.

## Incrementally Strengthening the Relaxation

The idea behind the incremental lower bound function $h^{++}$ is as follows: Given a minimal-cost sequenced relaxed plan $A$, if it is valid also for the non-relaxed problem $P$, then the optimal plan cost for $P$ is equal to the cost of $A$ (and we have a plan to prove it). If not, then we can identify a set of "conflicts", $C$, which are conjunctive conditions that are required but fail to hold at some point in the execution of the plan. We construct a new problem, called $P^C$, in which these conjunctions are explicitly represented by new atoms. The new problem has the property that $A$ is not a relaxed plan for it (or, to be precise, no sequence of representatives

[6]The transitive reduction of a graph is the smallest edge-subgraph that has the same transitive closure.

of each action in $A$ is a relaxed plan). The minimum cost of a relaxed plan for the new problem is a lower bound also on the cost of solving $P$. There is no guarantee that it will be higher than the $h^+$ value for the original problem, but since the set of non-redundant minimal-cost sequenced relaxed plans is finite, repeated application of this procedure will eventually result in a higher lower bound. In the limit it will even result in an optimal plan, unless time or memory limits halt the process.

## The $P^C$ Construction

Let $C = \{c_1, \ldots, c_n\}$ be a set of (non-unit) conjunctions over the atoms in $P$. We construct a problem $P^C$, in which these distinguished conjunctions are explicitly represented by new atoms, called "meta-atoms". The set of actions in $P^C$ is also modified, so that the truth of the meta-atoms will accurately reflect the truth of the conjunctions they represent. This is essentially the $P^m_*$ construction (Haslum 2009), applied to a specific set of conjunctive conditions.

**Definition 3** *The set of atoms of $P^C$ contains all atoms of $P$, and for each $c \in C$ an atom $\pi_c$. $\pi_c$ is initially true iff $c$ holds in the initial state of $P$, and is a goal iff $c \subseteq G$ in $P$. For each action $a$ in $P$ and for each subset $C'$ of $C$ such that for each $c \in C'$ (i) $\mathrm{del}(a) \cap c = \emptyset$ and $\mathrm{add}(a) \cap c \neq \emptyset$; and (ii) any $c' \in C$ such that $c' \subset c$ and $c'$ satisfies (i) is also in $C'$, $P^C$ has an action $\alpha_{a,C'}$ with*

$$\begin{aligned}
&\mathrm{pre}(\alpha_{a,C'}) = \mathrm{pre}(a) \cup \textstyle\bigcup_{c \in C'}(c - \mathrm{add}(a)) \cup \\
&\qquad \{\pi_c \mid c \subseteq (\mathrm{pre}(a) \cup \textstyle\bigcup_{c \in C'}(c - \mathrm{add}(a))), c \in C\} \\
&\mathrm{add}(\alpha_{a,C'}) = \mathrm{add}(a) \cup \{\pi_c \mid c \in C'\} \cup \\
&\qquad \{\pi_c \mid c \subseteq \mathrm{add}(a), c \in C\} \\
&\mathrm{del}(\alpha_{a,C'}) = \mathrm{del}(a) \cup \{\pi_c \mid c \cap \mathrm{del}(a) \neq \emptyset, c \in C\}
\end{aligned}$$

*and* $\mathrm{cost}(\alpha_{a,C'}) = \mathrm{cost}(a)$.

Each action $\alpha_{a,C'}$ in $P^C$ is constructed from an action $a$ in $P$. We call this the *original action* for $\alpha_{a,C'}$. Conversely, for each action $a$ in $P$ we call the actions in $P^C$ whose original action is $a$ the *representatives of* $a$. Note that $P^C$ always has at least one representative of each action $a$, namely $\alpha_{a,\emptyset}$.

Intuitively, meta-action $\alpha_{a,C'}$ corresponds to applying $a$ in a state such that each conjunction $c \in C'$ will be made true by applying $a$, which is why $\alpha_{a,C'}$ adds $\pi_c$. This means that $a$ does not delete any part of $c$, and the part of $c$ not made true by $a$ is already true. For any conjunction $c$ that is made true by $a$ alone (i.e., such that $c \subseteq \mathrm{add}(a)$), $\pi_c$ is added by every representative of $a$.

The size of $P^C$ is (potentially) exponential in the size of $C$, i.e., the number of conditions, but not in their size.

**Theorem 4** *Given any plan for $P^C$, the corresponding sequence of original actions is a plan for $P$. Conversely, given any plan for $P$, there is a plan for $P^C$ made up of a representative of each action in the plan for $P$ (and hence of equal cost).*

**Proof:** The first claim follows directly from that the preconditions and effects of each action in $P^C$ on atoms present in $P$ are identical to those of the original action in $P$.

For the second, we choose by induction a representative of each action in the plan such that in each state resulting from the execution of the plan for $P^C$, $\pi_c$ is true whenever $c$ is true. It is then easy to see that each action in this plan will be executable, since the precondition of an action in $P^C$ includes a meta-atom $\pi_c$ if and only if it includes all of $c$, and that the goal condition will hold at the end.

The correspondence holds in the initial state by definition. Suppose the current state is $s$, and that $a$ is the next action in the plan for $P$. Let $s'$ be the next state in the execution of the plan in $P$, i.e., the state that results from applying $a$ in $s$. Let $C'$ be the subset of conditions in $C$ that hold in $s'$. For each $c \in C'$, one of the following cases must hold: (1) $c$ holds in $s$ and $c \cap \mathrm{del}(a) = \emptyset$; (2) $c \subseteq \mathrm{add}(a)$; or (3) $c \not\subseteq \mathrm{add}(a)$, $c \cap \mathrm{add}(a) \neq \emptyset$, $c - \mathrm{add}(a)$ holds in $s$ and $c \cap \mathrm{del}(a) = \emptyset$. Let $C'' = C' - (\{c \mid c \subseteq \mathrm{add}(a)\} \cup \{c \mid c \text{ holds in } s\})$, i.e., $C''$ is exactly the subset of conditions in $C'$ that hold in $s'$ because of case (3), and choose the representative $\alpha_{a,C''}$. There is such an action in $P^C$ because condition (*i*) of the definition is implied by case (3), and condition (*ii*) by the choice of $C''$ as the set of all $c \in C$ that hold in $s'$ by this case. If $c$ holds in $s'$ by case (1), $\pi_c$ holds in the current state of execution in $P^C$ by inductive assumption, and it is not deleted by the chosen representative. If $c$ holds in $s'$ by cases (2) or (3), it is added by the chosen representative. Finally, for each $c$ that holds by case (3), $c - \mathrm{add}(a)$ must hold in $s$: thus, all atomic preconditions of the chosen representative are true in the current state, and therefore, by inductive assumption, so are any precondition meta-atoms representing conjunctions of the atomic preconditions. □

As of corollary, any lower bound on the cost of solving $P^C$ is also a lower bound on the cost of any plan for $P$.

Compared to $P$, $P^C$ contains no additional information. The reason why this construction is nevertheless useful is that the delete-relaxation of $P^C$ may be more informative than the delete-relaxation of $P$. If we have any additional source of information about unreachability in $P$, such as static mutexes or invariants, this can be used in the construction of $P^C$ to further strengthen $(P^C)^+$, by not including $\pi_c$ in the add effects of any action for any condition $c \in C$ that is known to be unreachable, in $P$. (The correspondance shown in theorem 4 holds also with this modification, since the chosen representatives only add meta-atoms for conditions made true by the plan.) Our current implementation uses static mutexes found by $h^2$.

## Conflict Extraction for a Sequenced Relaxed Plan

Consider a sequenced relaxed plan $A$. Deciding if it is a valid plan for the real problem, $P$, is easily done by simulating its execution. If the sequence is not a valid plan, then simulating it will at some point fail, meaning that the precondition of the next action to be applied does not hold in the current state. Call this the *failed condition*, and let $n_f$ be the corresponding node in $\mathrm{RPDG}(A)$. (Note that the failed condition may also be the goal.) Let $p$ be some unsatisfied atom in the failed condition. Since the sequence is valid in the relaxed sense, $p$ was either true initially or added by some action preceding the point of failure. Thus, $p$ was true

in some earlier state also in the real execution of the plan, but deleted by some action taking place between that point and the point of failure. Call that action the *deleter*, and let $n_d$ be its node in $\mathrm{RPDG}(A)$. We distinguish two cases:

Case 1: There is a path from $n_d$ to $n_f$ in $\mathrm{RPDG}(A)$. In this case, for each edge on the path from $n_d$ to $n_f$ choose an atom $q$ from the edge label and form the conflict $\{q, p\}$.

Case 2: There is no path from $n_d$ to $n_f$ in $\mathrm{RPDG}(A)$. Clearly there cannot be a path from $n_f$ to $n_d$, since $n_d$ appeared before $n_f$ in the sequence, which means the nodes are unordered. Let $n_c$ be the one of their nearest common descendants that appears first in the sequence, and let $L_d$ and $L_f$ be atom sets constructed by choosing one atom from each edge label on the path from $n_d$ to $n_c$ and the path from $n_f$ to $n_c$, respectively. Form the set of conflicts $\{\{q, q'\} \,|\, q \in L_d, q' \in L_f \cup \{p\}\}$. By property (5) of theorem 2, each conflict in the set is proper, in the sense that each in pair $q$ and $q'$ are distinct atoms.

**Theorem 5** *Let $A = a_1, \ldots, a_n$ be a non-redundant sequenced relaxed plan for $P$ which is not a real plan for $P$, and let $C$ be a set of conflicts extracted as described above. No sequence $A' = \alpha_1, \ldots, \alpha_n$, where each $\alpha_i$ is a representative of $a_i$, is a relaxed plan for $P^C$.*

**Proof:** In the execution of $A$, we have the two nodes $n_d$ and $n_f$, as described above. We examine the two possible cases:

Case 2 ($n_d$ and $n_f$ are unordered): Let $n_c$ be the nearest common descendant that is first in the sequence. The edges of the path from $n_d$ to $n_c$ are labeled with atoms $q_1, \ldots, q_m$ (possibly among others), and the edges of the path from $n_f$ to $n_c$ are labeled with atoms $q'_1, \ldots, q'_l$. Since $p$ is a precondition of the failed action, we can prepend it to the second path. The following picture illustrates:

$$n_d \underset{q_1}{\longrightarrow} n_1 \underset{q_2}{\rightarrow} \cdots \underset{q_{m-1}}{\longrightarrow} n_{m-1} \overset{q_m}{\searrow} n_c$$
$$\underset{p}{\rightarrow} n_f \underset{q'_1}{\longrightarrow} n'_1 \underset{q'_2}{\rightarrow} \cdots \rightarrow n'_{l-1} \overset{q'_l}{\nearrow}$$

Each pair of nodes where one is taken from the first path and the other from the second (including $n_f$) are unordered, and each conjunction $\{q_i, q'_j\}$ is represented by a meta-atom in $P^C$. The final condition, $\{q_m, q'_l\}$ is contained in $\mathrm{pre}(n_c)$, and thus $\pi_{\{q_m, q'_l\}}$ belongs to the precondition of any representative of $n_c$ (resp. to the goal, if $n_c$ is the goal node). We show that none of these meta-atoms hold at any point prior to $n_c$ in the relaxed execution of the sequence. This implies it is not a valid sequenced relaxed plan for $P^C$.

Let $a_d$ be the deleter (the action associated with $n_d$). Immediately after $a_d$, $\pi_{\{q_1, p\}}$ is false, since it was not true before ($q_1$ was not true before $a_d$, as otherwise it could not label the outgoing edge from $n_d$) and no representative of $a_d$ adds it (by definition, since $p \in \mathrm{del}(a_d)$). Suppose we have taken some steps along both paths: the next actions are $a_i$ and $a'_j$, respectively. By inductive assumption, $\pi_{\{q_i, q'_j\}}$ does not hold. Every representative of $a_i$ in $P^C$ has $q_i$ in its precondition. No representative of $a_i$ adds $q'_j$. (Some actions on the first path might add $p$, but if so, they come after $a_f$ in the sequence. This is ensured by the way the conflict was extracted: $a_d$ lies between the last point where $p$ was true and $a_f$.) Thus, any representative that adds $\pi_{\{q_{i+1}, q'_j\}}$ must be of the form $\alpha_{a_i, C'}$ with $\{q_{i+1}, q'_j\} \in C'$, and thus its precondition must include $q'_j$. Therefore, its precondition also includes $\pi_{\{q_i, q'_j\}}$ and no such representative is relaxed applicable. Hence, $\pi_{\{q_{i+1}, q'_j\}}$ is false after taking the next step along the first path, i.e., after executing $a_i$. By the same argument, after taking a step along the second path, i.e., after executing $a_j$, $\pi_{\{q_i, q'_{j+1}\}}$ is false. Thus, by induction, $\pi_{\{q_m, q'_l\}}$ is false after executing the last in sequence of the actions represented by $n_{m-1}$ and $n'_{l-1}$.

Next, suppose that $\pi_{\{q_m, q'_l\}}$ is added by some meta-action, say $\alpha$, between the last of $n_{m-1}$ and $n'_{l-1}$, and $n_c$. $\alpha$ must also add at least one of $q_m$ and $q'_l$. (This is true also if $q_m$ and $q'_l$ are meta-atoms, due to condition (*ii*) in definition 3.) Suppose it adds $q_m$: then $n_\alpha$ and $n_{m-1}$ cannot be unordered, and neither can $n_\alpha$ be ordered before $n_{m-1}$, because if so, $q_m$ would not label the edge from $n_{m-1}$ to $n_c$. (Recall the meaning of an edge in the RPDG: removing the action associated with $n_{m-1}$ makes the precondition $q_m$ of $n_c$ relaxed unreachable.) Thus, $n_\alpha$ follows $n_{m-1}$. Similarly, if $\alpha$ adds $q'_l$, $n_\alpha$ must follow $n_{l-1}$. If $\alpha$ does not add $q'_l$, it must have $q'_l$ as a precondition. This implies that there is an edge, labeled with $q'_l$, from $n_{l-1}$ to $n_\alpha$, because there is no other action adding $q'_l$ before $n_c$ (if there were $q'_l$ would not label the edge from $n_{l-1}$ to $n_c$), and thus not before $n_\alpha$. In summary, whether $\alpha$ adds either one of $q_m$ and $q'_l$ or both, $n_\alpha$ is a common descendant of $n_{m-1}$ and $n_{l-1}$. Since $\alpha$ must appear before $n_c$ in the sequence, this contradicts the choice of $n_c$ as the first-in-sequence nearest common descendant.

Case 1 ($n_d$ precedes $n_f$): This case is similar to the previous, but simpler in that we have only one path. The edge labels along this path include atoms $q_1, \ldots, q_m$, and the conflict set contains $\{q_i, p\}$ for each $q_i$, so $P^C$ has a corresponding meta-atom $\pi_{\{q_i, p\}}$. The precondition of the failed node contains $\{q_m, p\}$, and thus the precondition of any representative of this action (or the goal, if the failed node is $n_G$) in $P^C$ includes $\pi_{\{q_m, p\}}$. As in the previous case, none of these meta-atoms will hold in the relaxed execution of the sequence, and thus the failed action is not relaxed applicable. $\pi_{\{q_1, p\}}$ does not hold immediately after $a_d$. No action taking place between $a_d$ and $a_f$ in the sequence adds $p$ (again, this is because $a_d$ lies between the last point where $p$ was true and $a_f$) and thus any representative of any of these actions which adds $\pi_{\{q_{i+i}, p\}}$ requires $\pi_{\{q_i, p\}}$. □

## Conflict Extraction for Non-sequenced Plans

A non-sequenced relaxed plan is valid for the real problem iff there exists a sequencing of it that is a valid plan. Enumerating the sequencings of a relaxed plan is straightforward: starting with an empty sequence, non-deterministically choose the next action to apply from those whose preconditions are initially true or added by previously applied actions, never applying an action more than once. Backtracking systematically over the non-deterministic choices yields all sequencings. (When choosing the next action to apply, we divide the candidates into sets of mutually commutative

actions and branch only on which set to apply. Changing the order of actions in a commutative set will not change the relaxed or real validity of a sequence. It may, however, affect which conflicts are found, as described below.)

The conflict extraction procedure above is defined for sequenced relaxed plans, and there is a spectrum ways to apply it to a non-sequenced plan: At one end, we may choose one arbitrary sequencing and add only the conflicts geneated by this sequence. We may then find for the modified problem a relaxed plan comprised of the same set of actions, but if so, it will not permit them to be sequenced in the same way. At the other end of the spectrum, we may add the union of conflicts extracted from all sequencings of the relaxed plan, ensuring that the modified problem requires a different relaxed plan. Both methods have their drawbacks: Eliminating only one sequencing at a time can result in many iterations, in each of which we must compute a minimum-cost relaxed plan. On the other hand, adding a large number of conjunctions can cause the size of the modified problem to blow out.

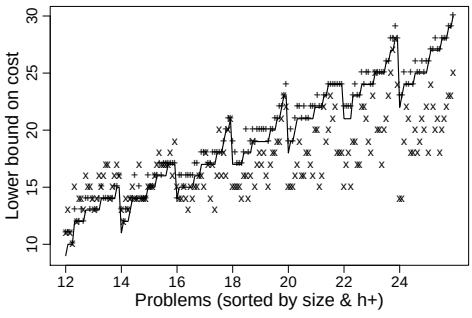
We adopt a middle ground: We enumerate all sequencings (modulo interleaving of commutative actions), and identify in each all triplets $(n_d, n_f, p)$ of deleter, failed node and false atom $p \in \mathrm{pre}(n_f)$, then choose a set of such triplets containing at least one from every sequence, and take the union of conflict sets generated by each chosen triplet. The choice is made with the aim of minimising the size of the final conflict set. (This is again a weighted hitting set problem, which we solve with an approximation algorithm.) The number of conjunctions generated by $(n_d, n_f, p)$ is estimated by the length of the shortest path from $n_d$ to $n_f$, if they are ordered (case 1), and the smallest product of the length of their shortests paths to a nearest common descendant, if they are not (case 2). If the deleter and failed node are unordered and have several nearest common descendants, we choose only one. (The nearest common descendants are all unordered, so there exist sequencings with each one of them appearing first. This is where the order of commutative actions may affect which conflicts are generated.) By theorem 5, there is at least one choice that generates a conjunction not already represented by a meta-atom.

## Iterating the Procedure

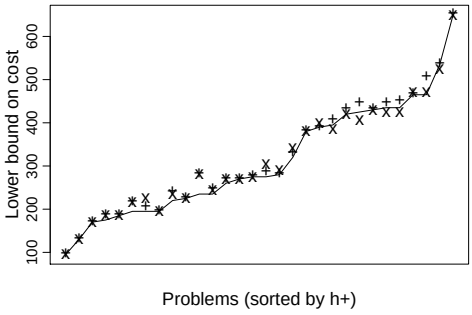
The $h^{++}$ procedure iterates finding a relaxed plan, extracting a set of conflicts, and adding meta-atoms representing them to the problem, until the relaxed plan is also a real plan (or we run out of patience or memory). In the process, we construct a sequence of problems, $P$, $P^{C_1}$, $(P^{C_1})^{C_2}$, etc. The conflict extraction procedure described above generates only binary conjunctions, but from the second iteration on, the atoms in such a pair may themselves be meta-atoms.

In fact, what we do is slightly different: Instead of $(\cdots(P^{C_1})^{\cdots})^{C_k}$, we construct the problem $P^{(C_1 \cup \ldots \cup C_k)}$, which is not the same. That is, instead of creating meta-atoms representing conjunctions of meta-atoms, we take the union of the sets of original atoms they represent as the new conjunction. Let $\mathrm{atoms}(\pi_c) = c$, when $\pi_c$ is a meta-atom, and $\mathrm{atoms}(p) = \{p\}$, when $p$ is an original atom. When we generate a conflict $\{\pi, \pi'\}$, where $\pi$ and $\pi'$ may be either original or meta-atoms, the new conjunction is $\mathrm{atoms}(\pi) \cup \mathrm{atoms}(\pi')$.

There are two reasons for this: First, if we build up conjunctions by combining meta-atoms pair-wise, we can end up with several meta-atoms that in fact represent the same conjunction of original atoms, which is clearly redundant. Second, the delete-relaxation of $P^{(C_1 \cup \ldots \cup C_k)}$ is in fact a stronger relaxation of $P$ than the delete-relaxation of $(\cdots(P^{C_1})^{\cdots})^{C_k}$. The reason for this is that in the second, "incremental", construction only meta-atoms representing conditions in the set $C_i$ will be added to the precondition of new action representatives created at that step. (Note that in definition 3, $\mathrm{pre}(\alpha_{a,C'})$ contains $\pi_c$ for *every* $c \in C$ contained in the set of original atoms in its precondition.) In practice, $P^{(C_1 \cup \ldots \cup C_k)}$ is of course constructed incrementally, not rebuilt from scratch each iteration. This can be done by keeping track of conjunctions already represented by meta-atoms, so that these can be added to the preconditions of new actions as required.




(a)




(b)

Figure 2: Comparison between $h^{++}$ and A*/LM-Cut on problems from the (a) Blocksworld and (b) Woodworking (IPC 2008) domains. The graphs show the $h^+$ value (–) and the highest lower bound proven by $h^{++}$ (+) and A* (×), within a 1 hour CPU limit. Problems are sorted by increasing size and/or $h^+$ value.

## Additional Results

As noted in the introduction, we have found few planning domains in which $h^{++}$ exceeds, or even comes close to,

the performance of A* search with the LM-Cut heuristic, as measured by the highest lower bound proven within time and memory limits. Figure 2 shows results for two such domains: Blocksworld (3ops) and Woodworking (IPC 2008). We can observe an interesting trend: Small or easy problems are quickly solved by A*, but often not by $h^{++}$, but as problems grow larger and/or have costlier (presumably longer) plans, the relative efficiency of A*/LM-Cut drops, such that it is eventually dominated even by $h^+$, while $h^{++}$ continues to sometime make a modest improvement over $h^+$. (Another comparison that should be made is of course with A* search using $h^+$ as the heuristic.)

The properties of a problem that affect the efficiency of $h^{++}$ are different from those that affect A*. For instance, a flat $f$-landscape, like that often caused by zero-cost actions, has little impact on $h^{++}$ (an abundance of zero-cost actions is even helpful, since it makes finding an optimal hitting set easier). On the other hand, the efficiency of $h^{++}$ is sensitive to the structure of the problem delete-relaxation: First, because the time taken to compute $h^{++}$ is (nearly always) dominated by the $h^+$ computation, which is in many problems prohibitively expensive. Second, if the delete-relaxation is inaccurate, the initial $h^+$ value will be far from the optimal real plan cost, and if there are many alternative relaxed plans, or many ways to sequence them, the number of iterations of problem modification needed to boost the $h^{++}$ value above the initial $h^+$ value will be large. It is interesting to note that in the three problem domains we found where $h^{++}$ is competitive with A*/LM-Cut, although they are not delete-free, optimal plans require most atoms to be made true only once.

## Conclusions & Open Questions

Incremental lower bound functions are vital for inspiring confidence in the quality of plans produced by planners that do not offer any optimality guarantees, and thus important for acceptance of automated planning in many applications. The idea of repeatedly solving increasingly less and less relaxed problem relaxations seems an attractive approach to constructing such functions, but has not been widely explored. We have proposed one realisation of this idea, based on the common delete-relaxation. Its present incarnation, however, performs well, compared to alternatives such as bounded A* search, in only a few domains; for most problems, it is hopelessly inefficient.

There are many open questions, and options to explore, concerning the effectiveness $h^{++}$, and incremental lower bound functions more generally. First, in many problems not even a first optimal relaxed plan can be computed in reasonable time: is this due to the intrinsic hardness of $h^+$, or is the iterative landmark-based algorithm we use inefficient compared to other conceivable methods? Betz and Helmert (2009) present polynomial-time algorithms for computing $h^+$ in a few specific domains: are there efficient $h^+$ algorithms for larger classes of domains? (e.g., characterised by properties of the problems causal or domain transition graphs). Second, the size of $P^C$ often grows exponentially with the number of conditions in $C$. A slightly different encoding of conjunctions can be done with only a linear size increase, using conditional action effects. This encoding has a weaker delete-relaxation, but still sufficient for theorem 5 to hold.[7] To make use of it, however, we need a method of computing optimal delete-relaxed plans for problems with conditional effects.

Finally, incremental lower bound functions can be devised in an analogous manner based on other relaxations, such as abstraction or the flow-based order relaxation used van den Briel et al. There are surely domains in which these will be more accurate than the delete-relaxation, just as there are a few in which the delete-relaxation is best.

## References


Aho, A.; Garey, M.; and Ullman, J. 1972. The transitive reduction of a directed graph. *SIAM Journal on Computing* 1(2):131–137.

Betz, C., and Helmert, M. 2009. Planning with $h^+$ in theory and practice. In *Proc. of the ICAPS'09 Workshop on Heuristics for Domain-Independent Planning*.

Bonet, B., and Helmert, M. 2010. Strengthening landmark heuristics via hitting sets. In *Proc. 19th European Conference on Artificial Intelligence (ECAI'10)*, 329–334.

Chatterjee, K.; Henzinger, T.; Jhala, R.; and Majumdar, R. 2005. Counterexample-guided planning. In *Proc. 21st International Conference on Uncertainty in Artificial Intelligence (UAI'05)*, 104–111.

Cornuéjols, G. 2008. Valid inequalities for mixed integer linear programs. *Mathematical Programming* 112(1):3–44. `http://dx.doi.org/10.1007/s10107-006-0086-0`.

Edelkamp, S., and Kissmann, P. 2008. GAMER: Bridging planning and general game playing with symbolic search. In *IPC 2008 Planner Abstracts*.

Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated Planning: Theory and Practice*. Morgan Kaufmann Publishers. ISBN: 1-55860-856-7.

Halldórsson, M. 2000. Approximations of weighted independent set and hereditary subset problems. *Journal of Graph Algorithms and Applications* 4(1):1–16.

Haslum, P. 2009. $h^m(P) = h^1(P^m)$: Alternative characterisations of the generalisation from $h^{\max}$ to $h^m$. In *Proc. of the 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In *Proc. 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*.

Slaney, J., and Thiebaux, S. 2001. Blocks world revisited. *Artificial Intelligence* 125. `http://users.cecs.anu.edu.au/~jks/bw.html`.

van den Briel, M.; Benton, J.; Kambhampati, S.; and Vossen, T. 2007. An LP-based heuristic for optimal planning. In *Proc. 13th International Conference on Principles and Practice of Constraint Programming (CP'07)*, 651–665.


[7] Thanks to Joerg Hoffmann and Emil Keyder for pointing this out.