A Practical, Integer-Linear Programming Model for the Delete-Relaxation in Cost-Optimal Planning

Tatsuya Imai[1] and **Alex Fukunaga**[2]

Abstract. We propose a new integer-linear programming model for the delete relaxation in cost-optimal planning. While a naive formulation of the delete relaxation as IP is impractical, our model incorporates landmarks and relevance-based constraints, resulting in an IP that can be used to directly solve the delete relaxation. We show that our IP model outperforms the previous state-of-the-art solver for delete-free problems. We then use LP relaxation of the IP as a heuristics for a forward search planner, and show that our LP-based solver is competitive with the state-of-the-art for cost-optimal planning.

1 Introduction

The *delete relaxation* of a classical planning problem is a relaxation of a planning problem such that all deletions are eliminated from its operators. It is clear that h^+, the optimal value of the delete relaxation of a planning instance is an admissible, lower bound on the cost of the optimal cost plan for the instance.

In cost-optimal planning, while h^+ is known to be more accurate than commonly used heuristics such as landmark-cut [11], current planners to not directly compute h^+ because the extra search efficiency gained from using h^+ is offset by the high cost of computing h^+. In fact, computing h^+ is known to be NP-complete [3]. As far as we are aware, the first use of h^+ inside a cost-optimal planner was by Betz and Helmert [1], who implemented domain-specific implementations of h^+ for several domains. Haslum evaluated the use of a domain-independent algorithm for h^+ [10] as the heuristic function for cost-optimal planning, and found that the performance was relatively poor [8]. In recent years, there have been several advances in the computation of h^+ [7, 14, 10].

A somewhat separate line of research is the increasing use of integer/linear programming (ILP) in domain-independent planning. The earliest use of linear programming (LP) in domain-independent planning that we are aware of was by Bylander, who used an LP encoding of planning as a heuristic function for a partial order planner [4]. Briel and Kambhampati formulated and solved planning as an integer program (IP) [18]. Recently, instead of modeling and directly solving planning as an IP, LP relaxations have been used to compute admissible heuristics in a search algorithm, including a network flow a LP heuristic for branch-and-bound [19], a heuristic for A* based on the state equations in SAS+ [2], and most recently, an LP encoding of a broad framework for operator-counting heuristics [15]. IP has also been used to compute hitting sets as part of the computation of h^+ in delete-free planning (in an improved version of the algorithm described in [10], [9].

[1] Tokyo Institute of Technology, Japan
[2] The University of Tokyo, Japan

In this paper, we propose a new, integer/linear programming approach to computing h^+. While a straightforward ILP model for h^+ is often intractable and not useful in practice, we developed an enhanced model, $\mathrm{IP}^{\mathrm{e}}(T^+)$, which incorporates landmark constraints for the delete relaxation, as well as relevance analysis to significantly decrease the number of variables. We show that $\mathrm{IP}^{\mathrm{e}}(T^+)$ allows significantly faster computation of h^+ compared to the state of the art.

Then, we consider the use of h^+ as a heuristic for A* in a cost-optimal, domain-independent planner. We further augment $\mathrm{IP}^{\mathrm{e}}(T^+)$ with constraints that consider some delete effects, as well as constraints for cycle avoidance, resulting in a new admissible heuristic which dominates h^+. Since $\mathrm{IP}^{\mathrm{e}}(T^+)$ is an IP, its LP relaxation, $\mathrm{LP}^{\mathrm{e}}(T^+)$, is also an admissible heuristic for domain-independent problem. Since even $\mathrm{LP}^{\mathrm{e}}(T^+)$ can be quite expensive, the ILP model can be further relaxed by omitting a subset of its constraints, resulting in $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$, an LP for the "relaxed" delete relaxation.

We empirically evaluate the ILP models by embedding them as heuristics in an A*-based planner. We implemented a simple method for automatically selecting which LP formulation to use as the heuristic, based on a comparison of their values at the root node. The resulting planner performs comparably to the state-of-the-art, cost-optimal planners, Fast-Downward with the landmark-cut heuristic [11] and Fast-Downward using the hybrid bisimulation merge-and-shrink heuristic [13].

The rest of the paper is organized as follows. Section 2 proposes the basic ILP model for h^+. Section 3 describes enhancements to the ILP model which significantly speeds up computation of h^+. Section 4 augments the ILP model by adding counting constraints, which results in a IP bound that dominates h^+. Section 5 summarizes the relationship among ILP models, and describes a simple method for selecting which model to apply to a given problem instance. Section 6, experimentally evaluates the proposed ILP models, as well as a portfolio approach that automatically selects one of the ILP models.

2 ILP model for h^+

A STRIPS planning task is defined by a 4-tuple $T = \langle P, A, I, G\rangle$. P is a set of *propositions*. A is a set of *actions*. A *state* is represented by a subset of P, and applying an action to a state adds some propositions and removes some propositions in the state. Each action $a \in A$ is composed of three subsets of P, $\langle \mathrm{pre}(a), \mathrm{add}(a), \mathrm{del(a)}\rangle$ which are called the *preconditions*, *add effects*, and *delete effects*. An action a is applicable to a state S iff it satisfies $\mathrm{pre}(a) \subseteq S$. By applying a to S, propositions in S change from S to $S(a) = ((S \setminus \mathrm{del}(a)) \cup \mathrm{add}(a))$. For a sequence of actions $\pi = (a_0, \cdots, a_n)$, we use $S(\pi)$ to denote $((((S \setminus \mathrm{del}(a_0)) \cup \mathrm{add}(a_0)) \setminus \mathrm{del}(a_1)) \cup \cdots) \cup \mathrm{add}(a_n)$.

Let $I \subseteq P$ be the *initial state* and $G \subseteq P$ the *goal*. The target

of a planning task is to find a sequence of actions to transform I to a state S that satisfies $G \subseteq S$. Formally, a feasible solution, i.e., a *plan*, is a sequence of actions $\pi = (a_0, \cdots, a_n)$ that satisfies (i) $\forall i, \mathrm{pre}(a_i) \subseteq I((a_0, \cdots, a_{i-1}))$, and (ii) $G \subseteq I(\pi)$. The target of a cost-optimal STRIPS planning is to find a shortest plan, or to find a plan π that minimizes $\sum_{a \in \pi} c(a)$ when the non-negative *cost* $c(a)$ of each action a is defined.

The delete relaxation of a task T, denoted by T^+, is a task $\langle P, A^+, I, G \rangle$ where A^+ is a set of delete-free actions defined as $A^+ = \{\langle \mathrm{pre}(a), \mathrm{add}(a), \emptyset \rangle \mid a \in A\}$. We also use T^+ to denote a task that is delete-free from the beginning without being relaxed.

2.1 ILP formulation of a delete-free problem

We formulate a delete free task $T^+ = \langle P, A^+, I, G \rangle$ as an integer-linear program. $\mathrm{IP}(T^+)$ denotes the IP problem derived from T^+, and we use $\pi^* = (a_0^*, \cdots, a_n^*)$ to denote an optimal plan for T^+ derived from an optimal solution of $\mathrm{IP}(T^+)$. Similarly $\mathrm{LP}(T^+)$ denotes the LP relaxation of $\mathrm{IP}(T^+)$. Note that for any feasible and non-redundant (i.e., same actions appear only once) solution of $\mathrm{IP}(T^+)$ (not just the optimal solution), we can derive a corresponding, feasible plan for T^+ that has same cost as the $\mathrm{IP}(T^+)$ solution.

First, we define the variables of $\mathrm{IP}(T^+)$. In addition to being able to derive a plan from $\mathrm{IP}(T^+)$, there always exists a injective mapping from a feasible non-redundant plan to an $\mathrm{IP}(T^+)$ solution. Thus, we also show the feasible assignments of variables that can be derived from a feasible plan of T^+, as well as the meanings and roles of the variables.

proposition: $\forall p \in P, \mathcal{U}(p) \in \{0, 1\}$. $\mathcal{U}(p) = 1$ iff $p \in I(\pi^*)$.
action: $\forall a \in A, \mathcal{U}(a) \in \{0, 1\}$. $\mathcal{U}(a) = 1$ iff $a \in \pi^*$ holds.
add effect: $\forall a \in A, \forall p \in \mathrm{add}(a), \mathcal{E}(a, p) \in \{0, 1\}$. $\mathcal{E}(a, p) = 1$ iff $a \in \pi^*$ holds and a achieves p first.
time (proposition): $\forall p \in P, \mathcal{T}(p) \in \{0, \cdots, |A|\}$. $\mathcal{T}(p) = t$ when $p \in I(\pi^*)$ and p is added by a_{t-1}^* first. $\mathcal{T}(p) = 0$ for $p \notin I(\pi^*)$.
time (action): $\forall a \in A, \mathcal{T}(a) \in \{0, \cdots, |A| - 1\}$. $\mathcal{T}(a) = t$ when $a = a_t^*$. $\mathcal{T}(a) = |A| - 1$ when $a \notin \pi^*$.
initial proposition: $\forall p \in P, \mathcal{I}(p) \in \{0, 1\}$. $\mathcal{I}(p) = 1$ iff $p \in I$.

If $p \in P$ appears more than once, use first indices for $\mathcal{T}(p)$. Variables $\mathcal{I}(p)$ are auxiliary variables for computing h^+. Although they are redundant when solving a delete-free task only one time, they are useful to avoid reconstructing constraints for each state when $\mathrm{IP}(T^+)$ or $\mathrm{LP}(T^+)$ are embedded as a heuristic function in a forward-search planner and called for each state.

The objective function seeks to minimize $\sum_{a \in A} c(a)\mathcal{U}(a)$.

Because of this objective function, the cost of an IP solution is equal to the cost of the corresponding (delete-free) plan.

Finally we define following six constraints.

1. $\forall p \in G,\ \mathcal{U}(p) = 1$.
2. $\forall a \in A,\ \forall p \in \mathrm{pre}(a), \mathcal{U}(p) \geq \mathcal{U}(a)$.
3. $\forall a \in A,\ \forall p \in \mathrm{add}(a), \mathcal{U}(a) \geq \mathcal{E}(a, p)$.
4. $\forall p \in P,\ \mathcal{I}(p) + \sum_{a \in A \text{ s.t.} p \in \mathrm{add}(a)} \mathcal{E}(a, p) \geq \mathcal{U}(p)$.
5. $\forall a \in A,\ \forall p \in \mathrm{pre}(a),\ \mathcal{T}(p) \leq \mathcal{T}(a)$.
6. $\forall a \in A,\ \forall p \in \mathrm{add}(a), \mathcal{T}(a) + 1 \leq \mathcal{T}(p) + (|A| + 1)(1 - \mathcal{E}(a, p))$.

There exists a feasible plan only if $\mathrm{IP}(T^+)$ has a feasible solution. When $\mathrm{IP}(T^+)$ is solved optimally, an optimal plan for T^+ is obtained according to following lemma. For a variable $\mathcal{V}$ of $\mathrm{IP}(T^+)$, $\mathcal{V}_F$ describes the assignment of $\mathcal{V}$ on a solution F of $\mathrm{IP}(T^+)$.

Proposition 1. *Given a feasible solution F for $\mathrm{IP}(T^+)$, the action sequence obtained by ordering actions in the set $\{a \mid \mathcal{U}(a)_F = 1\}$ in ascending order of $\mathcal{T}(a)_F$ is a feasible plan for T^+.*

Proof: At first we show that π satisfies the condition (ii) of a plan (i.e., $G \subseteq I(\pi)$) by proof of contradiction. Assume that there exists a proposition $g \in G$ that satisfies $g \notin I(\pi)$. There exists no action achieving g in π according to the assumption. Since F is a solution of $\mathrm{IP}(T^+)$, $\mathcal{U}(g)_F = 1$ holds according the constraint 1. Since $g \notin I(\pi)$ deduces $g \notin I$, $\mathcal{I}(g)_F = 0$. Therefore, to satisfy the condition 4, there must exist an action $a \in A$ that satisfies $g \in \mathrm{add}(a)$ and $\mathcal{E}(a, g)_F = 1$. However, to satisfy the constraint 3, $\mathcal{U}(a)_F = 1$ has to hold. This means $a \in \pi$, and this contradicts the assumption.

Next we show that π satisfies condition (i) (i.e., $\forall i, \mathrm{pre}(a_i) \subseteq I((a_0, \cdots, a_{i-1}))$). For the base case of inductive proof, assume that there exists a proposition $p \in P$ satisfying $p \in \mathrm{pre}(a_0)$ and $p \notin I$. Since $a_0 \in \pi$, $\mathcal{U}(a_0)_F = 1$ has to hold, and $\mathcal{U}(p)_F = 1$ has to hold according to the constraint $\mathcal{U}(p)_F \geq \mathcal{U}(a_0)_F$. Then, similar to the proof of condition (ii), there must exist an action $a \in A$ that satisfies $p \in \mathrm{add}(a)$, $\mathcal{U}(a)_F = 1$, and $\mathcal{E}(a, p)_F = 1$. However, to satisfy constraint 5, $\mathcal{T}(p) \leq \mathcal{T}(a_0)$ has to be true, and $\mathcal{T}(a) + 1 \leq \mathcal{T}(p)$ has to hold to satisfy condition 6. Therefore we have $\mathcal{U}(a)_F = 1$ and $\mathcal{T}(a) < \mathcal{T}(a_0)$, but a_0 is the first action of π, a contradiction.

Similar to the case of $i = 0$, when $i > 0$, if $\mathrm{pre}(a_i) \subseteq I((a_0, \cdots, a_{i-1}))$ is not true, there must exist an action $a \notin (a_0, \cdots, a_{i-1})$ that satisfies $\mathcal{U}(a)_F = 1$ and $\mathcal{T}(a) < \mathcal{T}(a_i)$, contradicting the fact that a_i is the i-th action of the sequence π. □

3 Enhancements for ILP model

In this section, we introduce some variable elimination techniques and some modifications of constraints. As we will show in the experimental results, these enhancements significantly reduce the time to solve $\mathrm{IP}(T^+)$ and $\mathrm{LP}(T^+)$. Some of the enhancements are adopted into our IP framework from previous work in planning research. In particular, landmarks, which have been extensively studied in recent years, play very important role.

Note that while some of the enhancements introduce cuts that render some solutions of $\mathrm{IP}(T^+)$ mapped from feasible plans infeasible, *at least one optimal plan will always remain*.

3.1 Landmark Extraction and Substitution

A *landmark* is an element which needs to be used in every feasible solution. We use two kinds of landmarks, called *fact landmarks* and *action landmarks* as in [7]. A fact landmark of a planning task T is a proposition that becomes true on some state of every feasible plan, and an action landmark of a planning task T is an action that is included in every feasible plan. We also say that a fact or action landmark l is a *landmark of a proposition* p if l is a landmark of the task $\langle P, A, I, \{p\} \rangle$. Similarly we also say that a landmark l is a *landmark of an action* a if l is a landmark of the task $\langle P, A, I, \mathrm{pre}(a) \rangle$. In the IP model of a delete-free task T^+, if a proposition p is a fact landmark, then we can substitute $\mathcal{U}(p) = 1$. Similarly, if an action a is an action landmark, then we can substitute $\mathcal{U}(a) = 1$.

In this work, we extract some kinds of action landmarks and fact landmarks according to following facts. The contrapositions of these propositions are clearly true.

Proposition 2. *Given a feasible delete-free task T^+, an action $a \in A$ is an action landmark of T^+ if the task $\langle P, A \setminus \{a\}, I, G \rangle$ is infeasible.*

Proposition 3. *Given a feasible delete-free task T^+, a proposition $p \in P$ is a fact landmark of T^+ if the task $\left\langle P, A \setminus A_p^{\mathrm{add}}, I \setminus \{p\}, G \right\rangle$ is infeasible, where A_p^{add} is defined as $A_p^{\mathrm{add}} = \{a \mid p \in \mathrm{add}(a)\}$.*

Zhu et al. defined a kind of fact landmark called *causal landmark* [20]. A proposition p is a causal landmark if $\left\langle P, A \setminus A_p^{\mathrm{pre}}, I \setminus \{p\}, G \right\rangle$ is infeasible, where $A_p^{\mathrm{pre}} = \{a \mid p \in \mathrm{pre}(a)\}$. If $\left\langle P, A \setminus A_p^{\mathrm{pre}}, I \setminus \{p\}, G \right\rangle$ does not have any solution, then $\left\langle P, A \setminus A_p^{\mathrm{add}}, I \setminus \{p\}, G \right\rangle$ is also infeasible, therefore using A_p^{add} instead of A_p^{pre} can extract larger set of fact landmarks. Keyder et al. proposed AND-OR graph based landmark extracting method generalized from a causal landmark extracting algorithm proposed Zhu et al. [12]. We use similar algorithm to extract both of our fact landmarks and action landmarks.

3.2 Relevance Analysis

Backchaining relevance analysis is widely used to eliminate irrelevant propositions and actions of a task. An action a is relevant if (i) $\mathrm{add}(a) \cap G \neq \emptyset$, or (ii) there exists a relevant action a' satisfying $\mathrm{add}(a) \cap \mathrm{pre}(a') \neq \emptyset$. A proposition p is relevant if (i) $p \in G$, or (ii) there exists a relevant action a and $p \in \mathrm{pre}(a)$ holds. In addition to this, as Haslum et al. noted, it is sufficient to consider relevance with respect to only a subset of first achievers of add effect [10]. Although they defined a first achiever by achievability of a proposition, it is completely equivalent to the following definition: an action a is a first achiever of a proposition p if $p \in \mathrm{add}(a)$ and p is not a fact landmark of a. When we use $\mathrm{fadd}(a)$ to denote $\{p \in \mathrm{add}(a) \mid a \text{ is a first achiever of } p\}$, it is sufficient to use fadd instead of add on the above definition of relevance.

If $a \in A$ or $p \in P$ is not relevant, we can eliminate a variable as $\mathcal{U}(a) = 0$ or $\mathcal{U}(p) = 0$. In addition to this, if $p \in \mathrm{add}(a)$ but a is not a first achiever of p, we can eliminate a variable as $\mathcal{E}(a, p) = 0$.

3.3 Dominated Action Elimination

On a delete-free task, if two actions have same add effect, then it is clearly sufficient to use at most one of two actions. Here we introduce a technique that eliminates an useless action (*dominated action*) extending this idea. If there exists a dominated action a, we can eliminate a variable as $\mathcal{U}(a) = 0$. We omit the proof due to space.

Proposition 4. *Given a feasible delete-free task T^+, there exists an optimal plan that does not contains $a \in A$ if there exists an action $a' \in A$ satisfying following: (i) $\mathrm{fadd}(a) \subseteq \mathrm{fadd}(a')$, (ii) for any $p \in \mathrm{pre}(a')$, p is a fact landmark of a or $p \in I$, and (iii) $c(a) \geq c(a')$.*

Robinson proposed similar constraints for a MaxSAT-based planner, but his condition is stricter than condition (ii) [16].

3.4 Immediate Action Application

On a delete-free task T^+, applying some types of actions to the initial state do not hurt optimality. We adopt to use an action with cost zero as [6] and an action landmark as [7] to this enhancement. For a delete-free task T^+, if an action $a \in A$ satisfies $c(a) = 0$ and $\mathrm{pre}(a) \subseteq I$, then a sequence made by connecting a before an optimal plan of $\langle P, A \setminus \{a\}, I \cup \mathrm{add}(a), G \rangle$ is an optimal plan of T^+. Similarly, if an action a is an action landmark of T^+ and a is applicable to I, you can apply a to I immediately.

For $\mathrm{IP}(T^+)$, variables $\mathcal{T}(p)$ for $p \in I$ can be eliminated by substituting zero. Given a sequence of immediate applicable actions $(a_0, \cdots, a_k)$ (it must be a correct applicable sequence), we can eliminate some variables as follows: (i) $\mathcal{U}(a_i) = 1$, (ii) $\mathcal{T}(a_i) = i$, (iii) $\forall p \in \mathrm{pre}(a_i), \mathcal{U}(p) = 1$, and (iv) $\forall p \in \mathrm{add}(a_i) \setminus I((a_0, \cdots, a_{i-1})), \mathcal{U}(p) = 1, \mathcal{T}(p) = i$ and $\mathcal{E}(a_i, p) = 1$.

3.5 Iterative Application of Variable Eliminations

The variable elimination techniques described above can interact synergistically with each other resulting in a cascade of eliminations. For example, landmarks increase non relevant add effects, which increases dominated actions, which can result in new landmarks. Therefore, we used a iterative variable eliminating algorithm which applies eliminations until quiescence.

A full landmark extraction pass after each variable elimination would be extremely expensive, but landmark extraction can be implemented incrementally. Hence we perform a complete landmark extraction once for each state, and after that, the incremental extraction is executed after each variable reduction.

3.6 Inverse action constraints

We define the following inverse relationship between a pair of actions for a delete-free task T^+. For two actions $a_1, a_2 \in A$, a_1 is an *inverse action* of a_2 if it satisfies following: (i) $\mathrm{add}(a_1) \subseteq \mathrm{pre}(a_2)$, and (ii) $\mathrm{add}(a_2) \subseteq \mathrm{pre}(a_1)$. By the definition, it is clear that if a_1 is an inverse action of a_2, then a_2 is an inverse action of a_1. Inverse actions satisfy following fact (proof omitted due to space).

Proposition 5. *For a delete-free task T^+, a feasible solution $\pi = (a_0, \cdots, a_n)$ is not optimal if $a_i \in \pi$ is an inverse action of $a_j \in \pi$ and both of a_i and a_j have non-zero cost.*

Let $\mathrm{inv}(a, p)$ denote the set of inverse actions of an action a which have p as add effect. There are several possible ways to use above proposition (e.g., $\mathcal{U}(a) + \mathcal{U}(a') \leq 1$, for all $a' \in \mathrm{inv}(a)$). On $\mathrm{IP}(T^+)$, due to avoid adding a huge number of constraints, we modify constraint 2 as follows:

2. $\forall a \in A,\ \forall p \in \mathrm{pre}(a), \mathcal{U}(p) - \sum_{a' \in \mathrm{inv}(a,p)} \mathcal{E}(a', p) \geq \mathcal{U}(a)$.

We use e (e.g. $\mathrm{LP}^{\mathrm{e}}(T^+)$) to denotes the ILP after all of the reductions in Sections 3.1-3.6 have been applied.

3.7 Constraint Relaxation

So far in this section, we have presented enhancements which seek to speed up the computation of h^+. As we show experimentally in Section 6, computing $\mathrm{IP}(T^+)$ or $\mathrm{LP}(T^+)$ remains relatively expensive, even if we use all of the enhancements described above.

Thus, we introduce a relaxation for $\mathrm{IP}(T^+)$. We call $\mathrm{IP}(T^+)$ without constraints 5 and 6 *time-relaxed* $\mathrm{IP}(T^+)$, denoted $\mathrm{IP}_{\mathrm{tr}}(T^+)$. Similarly we call $\mathrm{LP}(T^+)$ without same constraints *time-relaxed* $\mathrm{LP}(T^+)$, denoted $\mathrm{LP}_{\mathrm{tr}}(T^+)$. It can be seen that if the relevance of propositions and actions has an ordering (i.e. it does not have a cycle) on T^+, then the optimal costs of $\mathrm{IP}(T^+)$ and $\mathrm{LP}(T^+)$ are the same as the optimal costs of $\mathrm{IP}_{\mathrm{tr}}(T^+)$ and $\mathrm{LP}_{\mathrm{tr}}(T^+)$ respectively. We shall show experimentally in Section 6.1 that the relaxation is quite tight (i.e., $\mathrm{IP}(T^+)$ and $\mathrm{IP}_{\mathrm{tr}}(T^+)$ often have the same cost), and that $\mathrm{IP}_{\mathrm{tr}}(T^+)$ can be computed significantly faster than $\mathrm{IP}(T^+)$. $\mathrm{LP}(T^+), \mathrm{LP}^{\mathrm{e}}(T^+), \mathrm{IP}^{\mathrm{e}}(T^+)$ have same behavior.

4 Counting Constraints

So far, we have concentrated on efficient computation of h^+, and all of our relaxations are bounded by h^+. However, our IP model can be extended with constraints regarding delete effects. By adding variables and constraints related to delete effects of actions, our model can also calculate lower bounds on the number of times each action must be applied. New variables are defined as follows:

- $\forall a \in A, \mathcal{N}(a) \in \{0, 1, \cdots\} : \mathcal{N}(a) = n$ iff a is used n times.
- $\forall p \in P, \mathcal{G}(p) \in \{0, 1\} : \mathcal{G}(p) = 1$ iff $p \in G$.

$\mathcal{G}(p)$ is also an auxiliary variable as $\mathcal{I}(p)$. New constraints are defined as follows:

7. $\forall a \in A, \mathcal{N}(a) \geq \mathcal{U}(a)$.
8. $\forall p \in P, \mathcal{G}(p) + \sum_{p \in \mathrm{predel}(a)} \mathcal{N}(a) \leq \mathcal{I}(p) + \sum_{p \in \mathrm{add}(a)} \mathcal{N}(a)$,

where $\mathrm{predel}(a) = \mathrm{pre}(a) \cap \mathrm{del}(a)$. Finally, the objective function is modified so as to minimize $\sum_{a \in A} c(a)\mathcal{N}(a)$.

These constraints correspond to the *net change* constraints that were recently proposed in [15], as well as the action order relaxation in [17], (both are based on SAS$^+$ formulations). Intuitively, the final constraint states that the number of times actions adding p are used must be equal to or larger than the number of times actions requiring and deleting p same time are used. Given a non delete-free task T, we use $\mathrm{IP}(T)$ to denote an IP problem composed of $\mathrm{IP}(T^+)$ and above new variables and constraints. We also use LP and tr as same as corresponding relaxations for $\mathrm{IP}(T^+)$. For any T and any feasible plan π for T, there exists a feasible solution of $\mathrm{IP}(T)$ with same cost as π, since the delete relaxation of π is a feasible plan of T^+. Hence the optimal cost of naive $\mathrm{IP}(T)$ is an admissible heuristic for T.

Unfortunately these new constraints conflict with dominated action elimination and zero cost immediate action application. When counting constraint is used, it is necessary to disable zero cost immediate action applying and to modify the condition of dominated action: an action a is a dominated action of action a' if (i) $\mathrm{add}(a) \subseteq \mathrm{add}(a')$, (ii) for any $p \in \mathrm{pre}(a')$, p is a fact landmark of a or $p \in I$, (iii) $c(a) \geq c(a')$, and (iv) $\mathrm{pre}(a') \cap \mathrm{del}(a') \subseteq \mathrm{pre}(a) \cap \mathrm{del}(a)$. On the other hand, following fact ensures that other enhancements do not hurt admissibility of $\mathrm{IP}(T)$. We omit detailed discussion due to space. We also use e (e.g. $\mathrm{LP}^{\mathrm{e}}(T)$) to denotes the ILP after all of the valid reductions have been applied.

Proposition 6. *Given a task T, let $\mathrm{IP}^{\mathrm{e}}(T^+)$ be a variable-reduced IP for T^+, and $\mathrm{IP}^{\mathrm{e}}(T)$ be an IP made from $\mathrm{IP}^{\mathrm{e}}(T^+)$ with counting constraints. For any feasible solution π of T, if there exists a solution of $\mathrm{IP}^{\mathrm{e}}(T^+)$ derived from a subsequence of π^+, then there exists a feasible solution of $\mathrm{IP}^{\mathrm{e}}(T)$ that has same cost as π.*

5 Relationship among the ILP bounds

Based on the definitions, we know that: $\mathrm{IP}_{\mathrm{tr}}(T^+) \leq \mathrm{IP}^{\mathrm{e}}_{\mathrm{tr}}(T^+) \leq \mathrm{IP}(T^+) = \mathrm{IP}^{\mathrm{e}}(T^+) \leq \mathrm{IP}(T) = \mathrm{IP}^{\mathrm{e}}(T)$. As for the LP relaxations, we know that $\mathrm{LP}(T^+) \leq \mathrm{LP}^{\mathrm{e}}(T^+)$, $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+) \leq \mathrm{LP}^{\mathrm{e}}(T^+)$, $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T) \leq \mathrm{LP}^{\mathrm{e}}(T)$, and $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T) \leq \mathrm{LP}^{\mathrm{e}}(T)$. However, $\mathrm{LP}^{\mathrm{e}}(T)$ does not always dominate $\mathrm{LP}^{\mathrm{e}}(T^+)$ since sets of eliminated variables are different because of dominated action elimination and zero-cost immediate action application. Figure 1 illustrates the dominance relationships among the bounds.

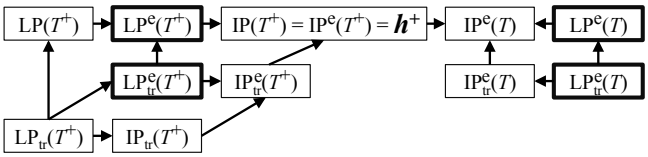

Figure 1. Dominance relationships. Edge $L_i \rightarrow L_j$ indicates "$L_i \leq L_j$". The 4 highlighted LP's are used in the A*/autoconf in Tables 2-3.

5.1 Automatic bound selection for each problem

While $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$ and $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T)$ are dominated by $\mathrm{LP}^{\mathrm{e}}(T^+)$ and $\mathrm{LP}^{\mathrm{e}}(T)$, respectively, the time-relaxed LPs are significantly cheaper to compute than their non-relaxed counterparts. Similarly, although $\mathrm{IP}^{\mathrm{e}}(T)$ dominates $\mathrm{IP}^{\mathrm{e}}(T^+)$, it is possible for $\mathrm{LP}^{\mathrm{e}}(T^+)$ to be larger than $\mathrm{LP}^{\mathrm{e}}(T)$. Thus, we have a set of 4 viable LP heuristics, none of which dominate the others when considering both accuracy and time. The "best" choice to optimize this tradeoff between heuristic accuracy and node expansion rate depends on the problem instance.

We implemented a simple mechanism for automatically selecting the LP to be used for each problem. First, we compute $\mathrm{LP}^{\mathrm{e}}(T^+)$, $\mathrm{LP}^{\mathrm{e}}(T)$, $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$, $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T)$ for the problem instance (i.e., at the root node of the A* search). We then select one based on the following rule: Choose the heuristic with the highest value. Break ties by choosing the heuristic that is cheapest to compute. Although the "cheapest" heuristic could be identified according to the cpu time to compute each heuristic, for many problems, the computations are too fast for robust timing measurements, so we simply break ties in order of $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$, $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T)$, $\mathrm{LP}^{\mathrm{e}}(T^+)$, $\mathrm{LP}^{\mathrm{e}}(T)$ (because this ordering usually accurately reflects the timing order). A more sophisticated method for heuristic selection may result in better performance (c.f. [5]), and is an avenue for future work.

6 Experimental Evaluation

Below, all experiments used the CPLEX 12.6 solver to solve integer linear programs. All experiments were single-threaded and executed on a Xeon E5-2650, 2.6GHz. We used a set of 1,366 IPC benchmark problems (from 1998 to 2011) distributed with Fast Downward. Our planner can currently handle the subset of PDDL which includes STRIPS, types, and action-costs. The full list of domains and # of instances per domain is shown in Table 3.

6.1 Comparison of ILP Bounds

We evaluate the quality of the integer/linear programming bounds by evaluating the optimal costs computed for these bounds.

First, we compute the ratio between the optimal cost of the LP relaxation and the IP (Figure 2). We take the ceiling of the LP cost, because the IPC benchmarks have integer costs. As shown in Table 2, the gap between the LP and IP relaxation are quite small. In fact, for the majority of problems, the gap between the rounded-up LP value and IP value is 0 for $\mathrm{IP}^{\mathrm{e}}(T^+)$, $\mathrm{IP}^{\mathrm{e}}(T)$, $\mathrm{IP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$, $\mathrm{IP}^{\mathrm{e}}_{\mathrm{tr}}(T)$, so the LP relaxation is frequently a perfect approximation of h^+.

Next, to understand the impact of various sets of constraints in the ILP formulations, Table 1 compares pairs of IP and LP formulations. The IP ratio for $\mathrm{IP}(T^+)$ vs $\mathrm{IP}^{\mathrm{e}}(T^+)$ is always 1 because they both compute h^+. However, on almost every single domain, the LP value of the extended formulation $\mathrm{LP}^{\mathrm{e}}(T^+)$ is significantly better (higher) than the basic formulation $\mathrm{LP}(T^+)$, indicating that variable elimination and the additional constraints serve to tighten the LP

bound. Thus, the enhancements to the basic model described in Section 3 provide a significant benefit. $\mathrm{LP}^{\mathrm{e}}(T)$ tends to be higher than $\mathrm{LP}^{\mathrm{e}}(T^+)$, indicating that that counting constraints enhances accuracy; note that in some cases $\mathrm{LP}^{\mathrm{e}}(T^+)$ is higher than $\mathrm{LP}^{\mathrm{e}}(T)$. The time-relaxations $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$ and $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T)$ are usually very close to $\mathrm{LP}^{\mathrm{e}}(T^+)$ and $\mathrm{LP}^{\mathrm{e}}(T)$, indicating that the time relaxation achieves a good tradeoff between computation cost and accuracy.

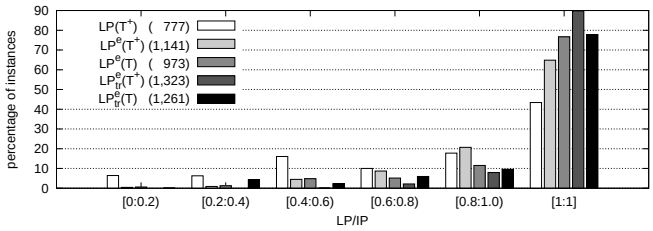

Figure 2. Ratio between the optimal costs of the IP's and their LP relaxations, categorized into buckets. [x:y) = "% of instances where the LP/IP ratio is in the range [x:y).

Table 1. Comparison of bounds: $il^+ = \mathrm{ILP}(T^+)$, $il^{e+} = \mathrm{ILP}^{\mathrm{e}}(T^+)$, $il^e = \mathrm{ILP}^{\mathrm{e}}(T)$, $il^{e+}_{tr} = \mathrm{ILP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$, $il^e_{tr} = \mathrm{ILP}^{\mathrm{e}}_{\mathrm{tr}}(T)$.

	il^+ / il^{e+}		il^e / il^{e+}		il^{e+}_{tr} / il^{e+}		il^e_{tr} / il^e	
	LP	IP	LP	IP	LP	IP	LP	IP
airport	.53	1.00	.99	1.00	.99	.99	1.00	.99
blocks	.92	1.00	.92	.92	1.00	1.00	1.00	1.00
depot	.54	1.00	.93	.99	.99	.92	1.00	.99
driverlog	.97	1.00	.91	.95	.96	.84	1.00	.96
elevators-opt08	.39	1.00	1.16	.96	.97	.64	1.00	.70
elevators-opt11	.36	1.00	1.17	.96	.96	.62	1.00	.73
floortile-opt11	.99	1.00	.93	.94	1.00	.97	1.00	.98
freecell	.48	1.00	1.01	1.00	.97	.92	1.00	.98
grid	-	-	.79	.85	.98	.79	1.00	.88
gripper	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
logistics98	.54	1.00	.89	1.00	.98	.88	1.00	1.00
logistics00	.47	1.00	.99	1.00	.99	.99	1.00	1.00
miconic	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
movie	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
no-mprime	.58	1.00	1.10	.97	.88	.66	1.00	.94
no-mystery	.58	1.00	1.03	.98	.92	.72	1.00	.96
nomystery-opt11	.97	1.00	.97	.97	1.00	1.00	1.00	1.00
openstacks	.38	1.00	1.00	1.00	1.00	1.00	1.00	1.00
openstacks-opt08	0	1.00	1.00	1.00	1.00	1.00	1.00	1.00
openstacks-opt11	-	-	1.00	1.00	1.00	1.00	1.00	1.00
parcprinter-08	.99	1.00	.92	.92	1.00	1.00	1.00	1.00
parcprinter-opt11	.99	1.00	.94	.94	1.00	1.00	1.00	1.00
parking-opt11	.90	1.00	.97	.97	.94	.87	.94	.86
pegsol-08	0	1.00	.81	.72	1.00	.68	1.00	.86
pegsol-opt11	0	1.00	.88	.73	1.00	.67	1.00	.86
pipes-notankage	.62	1.00	.94	.95	.92	.83	.97	.90
pipes-tankage	.62	1.00	.95	.96	.98	.87	1.00	.96
psr-small	.87	1.00	.38	.38	1.00	1.00	1.00	1.00
rovers	.63	1.00	.86	.77	1.00	1.00	1.00	1.00
satellite	.99	1.00	.99	.99	1.00	1.00	1.00	1.00
scanalyzer-08	1.00	1.00	1.00	1.00	1.00	.96	1.00	1.00
scanalyzer-opt11	1.00	1.00	1.00	1.00	1.00	.96	1.00	1.00
sokoban-opt08	.37	1.00	.88	.87	.99	.95	.99	.94
sokoban-opt11	.34	1.00	.90	.88	.99	.97	1.00	.96
storage	.55	1.00	.95	.91	1.00	1.00	1.00	1.00
transport-opt08	.26	1.00	3.42	1.00	.99	.36	1.00	.58
transport-opt11	-	-	-	-	.99	.43	-	-
visitall-opt11	1.00	1.00	.95	.93	.99	.97	.99	.95
woodworking08	.81	1.00	.94	.94	1.00	1.00	1.00	1.00
woodworking11	.80	1.00	.94	.94	1.00	1.00	1.00	1.00
zenotravel	.99	1.00	.92	.98	.96	.90	1.00	.99

6.2 Evaluating ILP for Delete-free planning

To evaluate the speed of our ILP approach, we compared $\mathrm{IP}^{\mathrm{e}}(T^+)$ with Haslum et al.'s h^+ algorithm [10] ("HST"), which is one of the state-of-the art solvers for the delete relaxation, of a set of 1,346 IPC benchmarks from the Fast Downward benchmark suite. Both solvers were run with a 15-minute time limit on each instance. The most recent version of HST was configured to use CPLEX to solve the hitting set subproblem, as suggested by Haslum [9].

The number of delete-free, relaxed instances that are solved by both planner is 905. HST solved 1,117 instances, and $\mathrm{IP}^{\mathrm{e}}(T^+)$ solved 1,186 instances. $\mathrm{IP}^{\mathrm{e}}(T^+)$ was faster than HST on 575 instances, and HST was faster than $\mathrm{IP}^{\mathrm{e}}(T^+)$ on 330 instances. Figure 3 shows the ratio of runtimes of HST to our solver, sorted in increasing order of the ratio, time(HST's h^+)/time($\mathrm{IP}^{\mathrm{e}}(T^+)$). The horizontal axis is the cumulative number of instances. Overall, $\mathrm{IP}^{\mathrm{e}}(T^+)$ outperform the state-of-the-art delete-free solver and indicates that direct computation of h^+ using integer programming is a viable approach (at least for computing h^+ once for each problem).

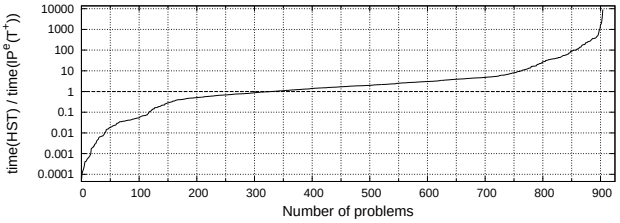

Figure 3. Computation of h^+: Comparison of $\mathrm{IP}^{\mathrm{e}}(T^+)$ and HST on delete-free, relaxed problems

6.3 Evaluating h^+-based heuristics in a cost-optimal planner

We embedded the ILP model into a A*-based, cost-optimal forward search planner. We first compared various configurations of our planner, as well as several configurations of Fast Downward (FD), given 5 minutes per problem instance and a 2GB memory limit. For the FD bisimulation merge-and-shrink heuristic, we use the IPC2011 hybrid bisimulation m&s configuration (`seq-opt-merge-and-shrink`).[3] The # of problems solved by each configuration is shown in Table 2.

Table 2. IPC benchmark problems: # solved with 5 minute time limit.

Configuration	# solved	Description
FD/LM-cut	746	Landmark Cut (`seq-opt-lmcut`)
FD/M&S IPC2011	687	IPC 2011 Merge-and-Shrink [13]
FD/$h^{\max}$	551	$h^{\max}$
A*/h^+	342	hsp_f planner using A* and h^+ heuristic [10, 8]
A*/IP(T^+)	358	basic IP formulation for h^+
A*/LP(T^+)	477	LP relaxation of IP(T^+)
A*/IP(T^+)+land	425	IP(T^+) + Landmarks
A*/LP(T^+)+land	564	LP relaxation of IP(T^+)
A*/IP$^{\mathrm{e}}$(T^+)	582	IP(T^+) with all enhancements in Sections 3.1-3.6
A*/LP$^{\mathrm{e}}$(T^+)	652	LP relaxation of IP$^{\mathrm{e}}$(T^+)
A*/IP$^{\mathrm{e}}$(T)	463	IP$^{\mathrm{e}}$(T^+) with counting constraints (Section 4)
A*/LP$^{\mathrm{e}}$(T)	608	LP relaxation of IP$^{\mathrm{e}}$(T)
A*/IP$^{\mathrm{e}}_{\mathrm{tr}}$($T^+$)	606	time-relaxation (Section 3.7) of IP$^{\mathrm{e}}$(T^+)
A*/LP$^{\mathrm{e}}_{\mathrm{tr}}$($T^+$)	674	LP relaxation of IP$^{\mathrm{e}}_{\mathrm{tr}}$($T^+$)
A*/IP$^{\mathrm{e}}_{\mathrm{tr}}$($T$)	554	time-relaxation of IP$^{\mathrm{e}}$(T)
A*/LP$^{\mathrm{e}}_{\mathrm{tr}}$($T$)	661	LP relaxation of IP$^{\mathrm{e}}_{\mathrm{tr}}$($T$)
A*/autoconf	722	Automated selection of LP at root node(Section 5.1)

As shown in Table 2, the basic IP model performs the worst, and is comparable to A*/h^+. As noted in [8], straightforward use of h^+ as a heuristic is unsuccessful (significantly worse than FD using $h^{\max}$). However, the addition of landmark constraints is sufficient to significantly increase the number of solved problems compared to A*/h^+, and A*/IP$^{\mathrm{e}}$(T^+), outperforms $h^{\max}$ and can be considered a somewhat useful heuristic. The time-relaxation results in significantly increases performance compared to A*/IP$^{\mathrm{e}}$(T^+) and A*/IP$^{\mathrm{e}}$(T). In addition, for all IP models, A* search using their corresponding LP relaxations as the heuristic function performs significantly better than directly using the IP as the A* heuristic. A*/LP$^{\mathrm{e}}$(T^+), A*/LP$^{\mathrm{e}}_{\mathrm{tr}}$($T^+$), and A*/LP$^{\mathrm{e}}_{\mathrm{tr}}$($T$), are all competitive with the bisimulation merge-and-shrink heuristic. While A*/LP$^{\mathrm{e}}$(T), does not perform quite as well, there are some problems where A*/LP$^{\mathrm{e}}$(T) per-

[3] While this is tuned for 30 minutes and suboptimal for 5 minutes, we wanted to use the same configuration as in the 30-minute experiments below.

forms best. Finally, A*/autoconf, which uses LP heuristic selection (Section 5.1) performs quite well, significantly better than its 4 components ($\mathrm{LP}^{\mathrm{e}}(T^+)$, $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$, $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T)$, $\mathrm{LP}^{\mathrm{e}}(T)$).

Table 3. 30 minutes, 2GB RAM: "evals"=# of calls to heuristic function

	Fast Downward LM-Cut		Fast Downward M&S		A*/autoconf	
Domain (# problems)	solved	evals	solved	evals	solved	evals
airport(50)	**28**	13403	23	461855	25	4640
barman-opt11(20)	**4**	1614605	**4**	5944586	3	473561
blocks(35)	28	95630	28	880799	**29**	51523
depot(22)	**7**	261573	**7**	1746549	**7**	34046
driverlog(20)	**14**	245920	13	4355507	13	56933
elevators-opt08(30)	**22**	1189951	14	10132421	13	66011
elevators-opt11(20)	**18**	1196979	12	11811143	10	65695
floortile-opt11(20)	**7**	2354266	**7**	10771362	**7**	152836
freecell(80)	15	180560	19	6291413	**45**	2177
grid(5)	2	94701	**3**	11667600	**3**	14197
gripper(20)	7	1788827	**20**	3131130	6	404857
logistics98(35)	6	169645	5	6825245	**7**	143897
logistics00(28)	**20**	212998	**20**	3007288	**20**	212985
miconic(150)	**141**	16635	77	3872365	**141**	15087
movie(30)	**30**	29	**30**	29	**30**	31
no-mprime(35)	**24**	55549	22	1490714	18	7260
no-mystery(30)	16	880031	**17**	3725239	12	1105
nomystery-opt11(20)	14	20744	**19**	9951860	14	754
openstacks(30)	**7**	157100	**7**	202732	**7**	4973
openstacks-opt08(30)	19	3254361	**21**	6347048	11	165070
openstacks-opt11(20)	14	4412937	**16**	8326670	6	294006
parcprinter-08(30)	19	699592	17	3129238	**29**	668
parcprinter-opt11(20)	14	949416	13	4091925	**20**	854
parking-opt11(20)	3	435359	**7**	8044843	1	2991
pegsol-08(30)	27	224149	**29**	705639	26	85760
pegsol-opt11(20)	17	370401	**19**	1092529	16	151110
pipes-notankage(50)	**17**	234717	**17**	1777823	13	6021
pipes-tankage(50)	12	361767	**16**	2447552	7	1926
psr-small(50)	49	178328	**50**	221152	**50**	4056
rovers(40)	7	77783	8	3395947	**11**	209551
satellite(36)	7	155990	7	1890912	**10**	26897
scanalyzer-08(30)	**15**	259961	14	6785907	8	4374
scanalyzer-opt11(20)	**12**	324943	11	8636568	5	6975
sokoban-opt08(30)	**30**	669669	24	3938226	23	75743
sokoban-opt11(20)	**20**	173004	19	3338708	19	77681
storage(20)	**15**	86439	**15**	1006600	**15**	21598
transport-opt08(30)	**11**	16807	**11**	1158282	10	58616
transport-opt11(20)	6	30550	**7**	4473292	5	116375
trucks(30)	10	462320	8	8478357	**15**	61067
visitall-opt11(20)	11	1255455	16	129229	**17**	20378
woodworking08(30)	17	759825	14	876479	**28**	767
woodworking11(20)	12	1076372	9	1357935	**18**	699
zenotravel(20)	**13**	318142	12	6727643	12	16571
Total (1366)	787		727		785	

Table 3 compares the coverage following algorithms on the IPC benchmark suite with 30 minute CPU time limit and 2GB memory limit: (1) A*/autoconf, which uses the LP heuristic selection mechanism described in Section 5.1 to choose among $\mathrm{LP}^{\mathrm{e}}(T^+)$, $\mathrm{LP}^{\mathrm{e}}(T)$, $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$, $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T)$, (2) FD using the Landmark Cut heuristic [11], and (3) FD using the IPC2011 bisimulation merge-and-shrink configuration (`seq-opt-merge-and-shrink`)[13].

Our results indicate that A*/autoconf is competitive with both Fast Downward using Landmark Cut, as well as the IPC2011 Merge-and-shrink portfolio configuration. None of these planners dominate the others, and each planner performs the best on some subset of domains. Compared to the two other methods, A*/autoconf seems to perform particularly well on the freecell, parcprinter, rovers, trucks, and woodworking domains. A*/h^+[10] solved 443 problems with a 30-minute time limit, which is significantly less coverage than than our LP-based planners with a 5-minute time limit (Table 2).

As described in Section 5.1, A*/autoconf selects the LP heuristic to use for each problem based on a comparison of LP values at the root node. $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$ was selected on 755 problems, $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T)$ on 447 problems, $\mathrm{LP}^{\mathrm{e}}(T^+)$ on 119 problems, and $\mathrm{LP}^{\mathrm{e}}(T)$ on 25 problems. On the remaining 20 problems, A*/autoconf timed out during LP computations for the bound selection process at the root node, indicating that for some difficult problems, the LP computation can be prohibitively expensive.

7 Conclusion

This paper proposed a new, integer-linear programming formulation of the delete relaxation h^+ for cost-optimal, domain-independent planning. The major contribution of this paper are: (1) We propose an enhanced IP model for h^+ using landmarks, relevance analysis, and action elimination, which is outperforms one of the previous state-of-the-art techniques for computing h^+ [10]; (2) We showed that the LP relaxations of the IP models are quite tight; and (3) We embedded our relaxed LPs in a A*-based forward search planner, A*/autoconf. We showed that A* search using $\mathrm{LP}^{\mathrm{e}}(T^+)$, $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$, or $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T)$ as its heuristic is competitive with the hybrid bisimulation merge-and-shrink heuristic [13]. Using a simple rule to select from among $\mathrm{LP}^{\mathrm{e}}(T^+)$, $\mathrm{LP}^{\mathrm{e}}(T)$, and $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$, $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T)$, A*/autoconf is competitive with the landmark cut heuristic. A*/autoconf performs well in some domains where other planners perform poorly, so our ILP-based methods are complementary to previous heuristics.

While it has long been believed that h^+ is too expensive to be useful as a heuristic for forward-search based planning, our work demonstrates that an LP relaxation of h^+ can achieve the right trade-off of speed and accuracy to be the basis of a new class of heuristics for domain-independent planning. Integrating additional constraints to derive heuristics more accurate than h^+ (e.g., the inclusion of net change constraints [15] in Section 4) offers many directions for future work.

Acknowledgments: Thanks to Patrik Haslum for assistance with his code for computing h^+ and his hsp_f planner. This research was supported by a JSPS Grant-in-Aid for JSPS Fellows and a JSPS KAKENHI grant.

REFERENCES

[1] C. Betz and M. Helmert, 'Planning with h^+ in theory and practice', in *KI 2009*, 9–16, Springer, (2009).
[2] B. Bonet, 'An admissible heuristic for SAS+ planning obtained from the state equation', in *Proc. IJCAI*, pp. 2268–2274, (2013).
[3] T. Bylander, 'The Computational Complexity of Propositional STRIPS Planning', *Artificial Intelligence*, **69**(1–2), 165–204, (1994).
[4] T. Bylander, 'A linear programming heuristic for optimal planning', in *AAAI/IAAI*, pp. 694–699. Citeseer, (1997).
[5] C. Domshlak, E. Karpas, and S. Markovitch, 'Online speedup learning for optimal planning', *JAIR*, **44**, 709–755, (2012).
[6] A. Gefen and R. Brafman, 'The minimal seed set problem', in *ICAPS*, pp. 319–322, (2011).
[7] A Gefen and R. Brafman, 'Pruning methods for optimal delete-free planning', in *ICAPS*, pp. 56–64, (2012).
[8] P. Haslum, 'Incremental lower bounds for additive cost planning problems', in *ICAPS*, pp. 74–82, (2012).
[9] P. Haslum. Personal communication, 2014.
[10] P. Haslum, J. Slaney, and S. Thiébaux, 'Minimal landmarks for optimal delete-free planning', in *ICAPS*, pp. 353–357, (2012).
[11] M. Helmert and C. Domshlak, 'Landmarks, critical paths and abstractions: What's the difference anyway?', in *ICAPS*, pp. 162–169, (2009).
[12] E. Keyder, S. Richter, and M. Helmert, 'Sound and complete landmarks for and/or graphs', in *ECAI*, pp. 335–340, (2010).
[13] R. Nissim, J. Hoffmann, and M. Helmert, 'Computing perfect heuristics in polynomial time: On bisimulation and merge-and-shrink abstraction in optimal planning', in *IJCAI*, pp. 1983–1990, (2011).
[14] F. Pommerening and M. Helmert, 'Optimal planning for delete-free tasks with incremental LM-cut', in *ICAPS*, pp. 363–367, (2012).
[15] F. Pommerening, G. Röger, M. Helmert, and B. Bonet, 'LP-based heuristics for cost-optimal planning', in *ICAPS*, (2014).
[16] N. Robinson, *Advancing Planning-as-Satisfiability*, Ph.D. dissertation, Griffith University, 2012.
[17] M.H.L. van den Briel, J. Benton, S. Kambhampati, and T. Vossen, 'An LP-based heuristic for optimal planning', in *Proc. CP-2007*, (2007).
[18] M.H.L. van den Briel and S. Kambhampati, 'Optiplan: A planner based on integer programming', *JAIR*, **24**, 919–931, (2005).
[19] M.H.L. van den Briel, T. Vossen, and S. Kambhampati, 'Loosely coupled formulation for automated planning: An integer programming perspective', *JAIR*, **31**, 217–257, (2008).
[20] L. Zhu and R. Givan, 'Landmark extraction via planning graph propagation', *ICAPS Doctoral Consortium*, 156–160, (2003).