Journal of Artificial Intelligence Research 54 (2015) 631-677 Submitted 7/15; published 12/15

On a Practical, Integer-Linear Programming Model for Delete-Free Tasks and its Use as a Heuristic for Cost-Optimal Planning

Tatsuya Imai TATSUYA.IMAI.30100041@GMAIL.COM
Graduate School of Information Science and Engineering
Tokyo Institute of Technology
Tokyo, Japan

Alex Fukunaga FUKUNAGA@IDEA.C.U-TOKYO.AC.JP
Graduate School of Arts and Sciences
The University of Tokyo
Tokyo, Japan

Abstract

We propose a new integer-linear programming model for the delete relaxation in cost-optimal planning. While a straightforward IP for the delete relaxation is impractical, our enhanced model incorporates variable reduction techniques based on landmarks, relevance-based constraints, dominated action elimination, immediate action application, and inverse action constraints, resulting in an IP that can be used to directly solve delete-free planning problems. We show that our IP model is competitive with previous state-of-the-art solvers for delete-free problems. The LP-relaxation of the IP model is often a very good approximation to the IP, providing an approach to approximating the optimal value of the delete-free task that is complementary to the well-known LM-cut heuristic. We also show that constraints that partially consider delete effects can be added to our IP/LP models. We embed the new IP/LP models into a forward-search based planner, and show that the performance of the resulting planner on standard IPC benchmarks is comparable 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 delete effects are eliminated from its operators. In the delete relaxation, every proposition that becomes true remains true and never becomes false again. The delete relaxation has been studied extensively in the classical planning literature because it can be used to estimate the cost of an optimal plan for the original planning problem and is therefore useful as a basis for heuristic functions for search-based domain-independent planning algorithms. A solution for the original planning problem is a solution for its delete relaxation, and the cost of the optimal solution to a delete-relaxed problem can be lower than the cost of the original problem because in the relaxation, every proposition only needs to be established once. Thus, the optimal cost of the delete relaxation of a planning problem (denoted h^+) is a lower bound on the optimal cost of the original planning problem. Despite the fact that computing h^+ is easier than solving the original planning problem, computing h^+ is itself NP-equivalent (Bylander, 1994) and poses a challenging problem.

In addition to its importance as a basis for heuristic functions for standard classical planning, the delete relaxation is also interesting in its own right, because there are some problems that can be naturally modeled as *delete-free problems* (i.e., problems where there are no actions with delete effects). For example, the "minimal seed set" problem, a problem in systems biology which seeks the minimal set of nutrients that are necessary for an organism to fully express its metabolism, can be mapped to a delete-free planning problem (Gefen & Brafman, 2011). Another application is in relational database query plan generation (Robinson, McIlraith, & Toman, 2014), where the problem of determining join orders can be modeled as a delete-free problem.

In this paper, we propose a new, integer programming (IP) approach to computing h^+.[1] We show that this model allows fast computation of h^+, and that the linear programming (LP) relaxation of this model can be used successfully as the heuristic function for an A*-based planner. The rest of this paper is structured as follows: We begin with a review of previous work on the delete relaxation as well as applications of LP to planning. Then we introduce $\mathrm{IP}(T^+)$, a basic integer programming model for a delete-free planning problem (Section 3) and show that it correctly computes h^+. Since the straightforward $\mathrm{IP}(T^+)$ model is often intractable and not useful in practice for computing h^+, we develop an enhanced model, $\mathrm{IP}^{\mathrm{e}}(T^+)$, which reduces the number of variables in the IP by using techniques such as landmark-based constraints, relevance analysis (Section 4). We evaluate the performance of the basic $\mathrm{IP}(T^+)$ and enhanced $\mathrm{IP}^{\mathrm{e}}(T^+)$ models in Section 5, and show that $\mathrm{IP}^{\mathrm{e}}(T^+)$ is competitive with the state-of-the-art methods for computing h^+.

While our objective is to use our IP models as a basis for a heuristic for forward state-space search based planning, solving an IP at every node in the search algorithm is computationally daunting, so in Section 6, we propose and evaluate two relaxations to our $\mathrm{IP}(T^+)$-based IP models. We consider the $\mathrm{LP}(T^+)$ and $\mathrm{LP}^{\mathrm{e}}(T^+)$, LP-relaxation of $\mathrm{IP}(T^+)$ and $\mathrm{IP}^{\mathrm{e}}(T^+)$, and show that the LP-relaxations usually closely approximate h^+. We also introduce a *time-relaxation* of the IP and LP models ($\mathrm{IP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$ and $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$, respectively) which further reduces the number of variables, at the cost of sometimes underestimating h^+, and show that these time-relaxations usually closely approximate h^+. We experimentally compare how closely these relaxed, delete-free models approximate h^+ with the LM-cut heuristic (Helmert & Domshlak, 2009) and show that these approaches are complementary.

Next, in Section 7, we evaluate the utility of our IP and LP models as heuristics for forward-search based planning by embedding them into an A*-based planner. Our results show that although $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$ is not competitive with the LM-cut heuristic overall, there are some domains where $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$ yields state-of-the-art performance, outperforming LM-cut.

We then turn to strengthening our IP and LP models by partially considering delete effects (Section 8). We add constraints that enforce lower bounds on the number of times an action must be used. These correspond to the *net change* constraints that were recently proposed by Pommerening et al. (2014), as well as the action order relaxation by van den Briel et al. (2007). This tightened bound $\mathrm{IP}_{\mathrm{c}}(T)$ dominates $\mathrm{IP}(T^+)$. Counting constraints can also be added to the LP-relaxation $\mathrm{LP}^{\mathrm{e}'}_{\mathrm{c}}(T)$, as well as the time-relaxed LP-relaxation $\mathrm{LP}^{\mathrm{e}'}_{\mathrm{ctr}}(T)$. However, the additional counting constraints makes the IP and LP more difficult, so in a A*-based planner that uses these bounds, there is a tradeoff between a tighter bound (fewer nodes searched by A*) and the time spent per node. As a result, we find that although counting constraints result in enhanced performance on some domains, it significantly degrades performance on other domains. We experimentally compare our counting-constraint enhanced models with the LMC-SEQ LP model of Pommerening et al. (2014) which combines landmark and net-change constraints, and show that, like LM-cut vs our delete-free LP's, these models are complementary.

1. This paper revises and extends the work originally reported by the authors in a paper presented at ECAI2014 (Imai & Fukunaga, 2014). Formal results and proofs which were not in the ECAI paper are included, and this paper contains a much more thorough experimental evaluation of our models (all of the experimental data is new).

Table 1 provides an overview of all of the IP/LP models discussed in Sections 3-8, and also serves as a roadmap of this paper . For each model, we indicate the section in the text where the model is introduced, the constraints used in the model, and the variable elimination optimizations used in the model. Figure 1 is a directed graph showing the dominance relationships among the optimal costs of the IP/LP models.

Finally, because there is no clear dominance relationship among our LP models (with respect to the performance of A*-based planners that use these LP models as the heuristic function), we propose and evaluate a simple automatic configuration heuristic which selects the LP to use as the heuristic for A* (Section 9). This simple automated bound selection significantly boosts performance, resulting in a ensemble-based LP-heuristic that is competitive with state-of-the-art heuristics. Section 10 concludes the paper with a summary and discussion of our results and some directions for future work.

2. Background and Related Work

This section first introduces the notation for planning tasks which will be used in the rest of the paper, then surveys related work on solving delete-free planning tasks as well as previous applications of IP/LP to domain-independent planning.

2.1 Preliminary Definitions

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*. A solution to a planning task is a sequence of actions that 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 basic STRIPS planning task can be extended to *STRIPS planning with action costs*, where each action $a \in A$ has an associated (non-negative) cost $c(a)$. The objective of *cost-optimal planning* in a STRIPS model with action costs is to find a plan π that minimizes the sum of the costs of its actions $\sum_{i=0}^{i=n} c(a_i)$.

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.

2.2 Previous Work on Computing h^+ and its Relaxations

The delete relaxation has been used as the basis for planning heuristics since the beginning of the recent era of interest in forward-state space search based planning (Bonet & Geffner, 2001). Unfortunately, computing h^+ is known to be NP-equivalent by reduction from vertex cover (Bylander,

Model	Constraints	Variable Eliminations	
$\mathrm{IP}(T^+)$ (Sec. 3)	C1, C2, C3, C4, C5, C6,	None	Basic delete-free task IP model (computes h^+)
$\mathrm{IP}^{\mathrm{e}}(T^+)$ (Sec. 4)	C1, **C2a** C3, C4, C5, C6	Landmarks (4.1), relevance (4.2), dominated action elimination (4.3), immediate action application (4.4)	Enhanced IP model (computes h^+)
$\mathrm{LP}(T^+)$ (Sec. 6.1)	Same as $\mathrm{IP}(T^+)$	None	LP relaxation of $\mathrm{IP}(T^+)$
$\mathrm{LP}^{\mathrm{e}}(T^+)$ (Sec. 6.1)	Same as $\mathrm{IP}^{\mathrm{e}}(T^+)$	Same as $\mathrm{IP}^{\mathrm{e}}(T^+)$	LP relaxation of $\mathrm{IP}^{\mathrm{e}}(T^+)$
$\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$ (Sec. 6.2)	C1, **C2a** C3, C4,	Same as $\mathrm{IP}^{\mathrm{e}}(T^+)$	LP-relaxation of time-relaxation of $\mathrm{IP}^{\mathrm{e}}(T^+)$
$\mathrm{IP}_{\mathrm{c}}(T)$ (Sec. 8)	C1, C2, C3, C4, C5, C6, **C7 C8**	None	Basic delete-free task IP model enhanced with counting constraints
$\mathrm{IP}^{\mathrm{e}'}_{\mathrm{c}}(T^+)$ (Sec. 8)	C1, **C2a** C3, C4, C5, C6 **C7 C8**	Landmarks (4.1), relevance (4.2), modified dominated action elimination (Definition 2)	Enhanced IP model with counting constraints
$\mathrm{LP}_{\mathrm{c}}(T)$ (Sec. 8)	Same as $\mathrm{IP}_{\mathrm{c}}(T)$	None	LP relaxation of $\mathrm{IP}_{\mathrm{c}}(T)$
$\mathrm{LP}^{\mathrm{e}'}_{\mathrm{c}}(T)$ (Sec. 8)	Same as $\mathrm{IP}^{\mathrm{e}'}_{\mathrm{c}}(T)$	Same as $\mathrm{IP}^{\mathrm{e}'}_{\mathrm{c}}(T)$	LP relaxation of $\mathrm{IP}^{\mathrm{e}'}_{\mathrm{c}}(T)$
$\mathrm{LP}^{\mathrm{e}'}_{\mathrm{ctr}}(T)$ (Sec. 8)	C1, **C2a** C3, C4, **C7 C8**	Same as $\mathrm{IP}^{\mathrm{e}'}_{\mathrm{c}}(T)$	LP-relaxation of time-relaxation of $\mathrm{IP}^{\mathrm{e}'}_{\mathrm{c}}(T)$
A*/autoconf (Sec. 9)	Selects among $\mathrm{LP}^{\mathrm{e}}(T^+)$, $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$, $\mathrm{LP}^{\mathrm{e}'}_{\mathrm{c}}(T)$, $\mathrm{LP}^{\mathrm{e}'}_{\mathrm{ctr}}(T)$.		Automatic LP Model Selection

Table 1: Overview of delete-relaxation based IP/LP models in this paper

$$\begin{tikzcd}
\mathrm{LP}(T^+) \arrow[r] & \mathrm{LP}^{\mathrm{e}}(T^+) \arrow[r] & \mathrm{IP}(T^+) = \mathrm{IP}^{\mathrm{e}}(T^+) = \boldsymbol{h}^+ \arrow[r] & \mathrm{IP}^{\mathrm{e}'}_{\mathrm{c}}(T) & \mathrm{LP}^{\mathrm{e}'}_{\mathrm{c}}(T) \arrow[l] \\
 & \mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+) \arrow[u] \arrow[r] & \mathrm{IP}^{\mathrm{e}}_{\mathrm{tr}}(T^+) \arrow[u] & \mathrm{IP}^{\mathrm{e}'}_{\mathrm{ctr}}(T) \arrow[u] & \mathrm{LP}^{\mathrm{e}'}_{\mathrm{ctr}}(T) \arrow[u] \arrow[l] \\
\mathrm{LP}_{\mathrm{tr}}(T^+) \arrow[uu] \arrow[ur] \arrow[r] & \mathrm{IP}_{\mathrm{tr}}(T^+) \arrow[ur] & & &
\end{tikzcd}$$

Figure 1: Dominance relationships among our IP/LP models. Edge $model_i \rightarrow model_j$ indicates "the optimal cost of $model_i \leq$ the optimal cost of $model_j$". The 4 highlighted LP's are the components of the A*/autoconf model.

1994), and therefore, from the beginning, researchers avoided direct computation of h^+, and instead sought approximations to h^+.

In satisficing planning, where optimal solutions are not required, a successful approach to deriving heuristics has been to approximate the delete relaxation. The *additive heuristic* (h^{add}) assumes that subgoals are independent and computes the sum of achieving each subgoal in the delete-relaxed model (Bonet & Geffner, 2001). The FF heuristic (Hoffmann & Nebel, 2001) constructs a planning graph (Blum & Furst, 1997) for the delete-relaxed problem, extracts a relaxed plan, and computes the number of actions in the relaxed plan, which is an upper bound on h^+.

In the case of cost-optimal planning, where each action is assigned a cost and the objective is to find a minimal cost plan, lower bounds on h^+ are the basis for several admissible heuristic functions that have been used in the literature. Bonet and Geffner (2001) proposed the h^{max} heuristic, which computes the highest cost associated with achieving the most costly, single proposition. While h^{max} is admissible, it is often not very informative (i.e, the gap between h^{max} and h^+ is large) because it only considers the single most costly goal proposition. The admissible landmark cut (LM-cut) heuristic (Helmert & Domshlak, 2009), approximates h^+ as follows. For state s, the LM-cut heuristic first computes $h^{max}(s)$, and if this is zero or infinite, then h^+ is zero or infinite, so $h^{LMcut}(s) = h^{max}(s)$. Otherwise, a disjunctive action landmark L (a set of actions at least one of which must be included in any relaxed plan) is computed, and the cost of all actions in L is reduced by $c(m)$, the cost of the minimal-cost action $m \in L$, and h^{LMcut} is increased by $c(m)$. This process is repeated until $h^{max}(s)$ (for the remaining, reduced problem) becomes 0. Other approximations to h^+ that are more informative than h^{max} include the set-additive heuristic (Keyder & Geffner, 2008) and cost-sharing approximations to h^{max} (Mirkis & Domshlak, 2007).

Previous planners have avoided direct computation of h^+ because the extra search efficiency gained from using h^+ is offset by the high cost of computing h^+. As far as we are aware, the first actual use of h^+ inside a cost-optimal planner was by Betz and Helmert (2009), who implemented *domain-specific* implementations of h^+ for several domains. More recently, Haslum et al. evaluated the use of a domain-independent algorithm for h^+ (Haslum, Slaney, & Thiébaux, 2012) as the heuristic function for A*-based cost-optimal planning, but found that the performance was relatively poor (Haslum, 2012).

In recent years, there have been several advances in the computation of h^+. Since, as described above, the LM-cut heuristic (Helmert & Domshlak, 2009) is a lower bound on h^+, a cost-optimal planner using the A* search algorithm and the LM-cut heuristic can be directly applied to the delete relaxation of a classical planning problem in order to compute h^+. It is possible to improve upon this by developing methods that exploit the delete-free property and are specifically tailored for solving the delete relaxation. Pommerening and Helmert (2012) developed an approach which uses IDA* or branch-and-bound with an incrementally computed LM-cut heuristic. Gefen and Brafman (2012) proposed action pruning for delete-free problems.

A different approach to computing h^+ is based on the observation that h^+ could be formulated as the problem of finding a minimal hitting set for sets of disjunctive action landmarks (Bonet & Helmert, 2010). This led to methods for computing h^+ by searching for minimum-cost hitting set for a complete set of action landmarks for the delete-relaxed planning problem (Bonet & Castillo, 2011; Haslum et al., 2012). While the original implementation of Haslum et al.'s hitting-set based h^+ solver used a problem-specific branch-and-bound algorithm (Haslum et al., 2012), an improved implementation (which we use in our experimental evaluation in Section 5) uses integer programming to solve the hitting set problem (Haslum, 2014a).

2.3 Integer/Linear Programming for Classical Planning

Another related line of research is the modeling of classical planning as integer/linear programs (ILP). The use of very high-performance, general problem solvers to solve planning problems was pioneered by Kautz and Selman, who solved planning problems by encoding them as propositional satisfiability (SAT) and applied state-of-the-art SAT solvers. The basic approach is to instantiate a SAT formula for which a satisfying assignment implies a t-step plan. SATPLAN starts with a small value of t (e.g., trivially, 1, or some other lower bound), instantiates a propositional formula $F(t)$ which is satisfiable if and only if a plan of t parallel steps or less exists. If $F(t)$ is satisfiable, then a minimal parallel makespan plan has been found. Otherwise, t is incremented, and this process is repeated until a plan is found. While the initial encodings were modestly successful (Kautz & Selman, 1992), advances in both SAT solver technology as well as improvements to the encoding and the integration of planning graphs (Blum & Furst, 1997) led to dramatic performance improvements (Kautz & Selman, 1996, 1999). Recent work on SAT-based planning includes improved encodings as well as execution strategies for SAT strategies that improve upon simply incrementing t as above (Rintanen, Heljanko, & Niemelä, 2006). In addition, improvements to SAT solvers which specifically target domain-independent planning have been investigated (Rintanen, 2012)

Since the expressiveness of integer programming (IP) subsumes SAT, SAT encodings can be straightforwardly translated to IP. However, direct translation of SAT encodings to IP resulted in poor performance, and a *state-change formulation* which replaces the original fluents in the SAT encoding with a set of variables that directly expresses the addition, deletion, and persistence of fluents was shown to be more successful as the basis for an IP model for planning (Vossen, Ball, Lotem, & Nau, 1999). This formulation was strengthened with additional mutual exclusion constraints (Dimopoulos, 2001). The Optiplan model (van den Briel & Kambhampati, 2005) combined the state-change IP formulation with the planning-graph based model refinement strategies and improvements by Dimopoulous (2001). As with the SAT-based approaches described above, IP models which are feasible if and only if a plan of up to t steps exists are constructed. However, unlike the SAT formulation, it is easy to directly encode action costs into the objective function for the IP model, so the IP models can be used to directly solve the cost-optimal planning problems. Another approach decomposes a planning instance into a set of network flow problems, where each subproblem corresponds to a state variable in the original planning problem (van den Briel, Vossen, & Kambhampati, 2008).

Instead of modeling and directly solving a classical planning problem as an IP, another approach, which we adopt in this paper, uses ILP models to provide a heuristic function which guides a state-space search planning algorithms such as A*. An early instance of this approach (which, to our knowledge, is also the earliest application of LP to classical planning) is LPlan, where an LP encoding of the classical planning problem is used as a heuristic function for a partial order planner (Bylander, 1997). Van den Briel et al. (2007) developed an admissible heuristic based on an LP model which represents a planning problem where the order in which actions are executed is relaxed, and each variable represents the number of times an action is executed. Delete effects are considered, in that there are constraints such that the number of actions that delete values can be incremented only if there are actions that add the value. Although this LP-based heuristic was not integrated into a planning system, they compared the relaxed problem cost found by their model with Bylander's LPlan LP model, as well as an LP model for h^+.

To our knowledge, the h^+ implementation by van den Briel et al. (2007) is the first implementation of an IP model of h^+. First, a relaxed planning graph (Blum & Furst, 1997) was expanded until quiescence, which results in the instantiation of all actions that are relevant for the optimal delete-free task as well as an upper bound on the number of steps in the optimal delete-free task. Then, h^+ was computed using a delete-relaxed, step-based encoding of the planning problem of Optiplan (van den Briel, 2015).

Cooper et al. (2011) showed that the optimal solution to the dual of an LP model which relaxes the action ordering corresponds to the best lower bound that can be obtained by applying transformations to the original planning problem that shift costs among actions that affect the same fluents.

Bonet proposed h^{SEQ}, an admissible, flow-based LP heuristic based on Petri Net state equations (Bonet, 2013) which was used as the heuristic for an A*-based planner. Bonet and van den Briel (2014) enhanced Bonet's flow-based LP model by adding action landmark constraint and implementing variable merging strategies, resulting in a competitive, admissible heuristic. Karpas and Domshlak (2009) proposed an LP formulation to compute optimal partitioning for landmarks. Pommerening et al. (2014) proposed an *operator counting* framework which enabled the unification of a number of ideas, including the state equation formulation (Bonet, 2013), post-hoc optimization constraints (Pommerening, Röger, & Helmert, 2013), as well as landmarks (the formulation by Bonet & Helmert, 2010, which is the dual of the formulation in Karpas & Domshlak, 2009) and state abstraction heuristics (Katz & Domshlak, 2010). They showed that combinations of constraints resulted in strong heuristics which significantly outperformed the LM-cut heuristic. A recent survey by Röger and Pommerening (2015) presents a survey of LP-based heuristics for planning which includes an earlier conference version of this paper (Imai & Fukunaga, 2014) and suggests how our delete-relaxation model could be incorporated into the operator counting framework by associating a operator-counting variable for each action variable (see below) in the delete-relaxed problem.

3. IP(T^+): The Basic IP Formulation of a Delete-Free Task

We now define the integer program IP(T^+), which is the IP formulation of the delete free task $T^+ = \langle P, A^+, I, G\rangle$. Note that for any feasible solution to IP(T^+) (not just the optimal solution), we can derive a corresponding, feasible and non-redundant (i.e., each action appears only once) plan for T^+ that has the same cost as the IP(T^+) solution.

First, we define the variables of IP(T^+). In addition to being able to derive a plan from IP(T^+), there always exists an injective mapping from a feasible non-redundant plan for an IP(T^+) solution. Thus, we also show the feasible assignments of variables that can be derived from a feasible plan for T^+, as well as the meanings and roles of the variables. We use $\pi = (a_0, \cdots, a_n)$ to denote a plan for T^+ corresponding to a solution for IP(T^+). We say that a is the *first achiever* of p in plan π if $p \notin I$, and a is the first action that achieves (establishes) p.

proposition: $\forall p \in P, \mathcal{U}(p) \in \{0,1\}$. $\mathcal{U}(p) = 1$ iff $p \in I(\pi)$. $\mathcal{U}(p)$ indicates whether proposition p is achieved in a relaxed plan for T^+.

action: $\forall a \in A, \mathcal{U}(a) \in \{0,1\}$. $\mathcal{U}(a) = 1$ iff $a \in \pi$ holds. $\mathcal{U}(a)$ indicates whether the action a is used in a relaxed plan.

add effect: $\forall a \in A, \forall p \in \text{add}(a), \mathcal{E}(a,p) \in \{0,1\}$. $\mathcal{E}(a,p) = 1$ iff $a \in \pi$ holds and a is the first achiever of p. $\mathcal{E}(a,p) = 0$ if p is true in I, or p is not achieved.

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. $\mathcal{T}(p) = 0$ if p is a member of I. $\mathcal{T}(p)$ indicates the time step where p is first achieved by its first achiever.

time (action): $\forall a \in A, \mathcal{T}(a) \in \{0, \cdots, |A|\}$. $\mathcal{T}(a) = t$ when $a = a_t$. $\mathcal{T}(a) = |A|$ when $a \notin \pi$. $\mathcal{T}(a)$ indicates the time step where a is first used.

initial proposition: $\forall p \in P, \mathcal{I}(p) \in \{0, 1\}$. $\mathcal{I}(p) = 1$ iff $p \in I$.

If $p \in P$ is achieved more than once, i.e., p appears in the add effects of multiple actions in π, we assign $\mathcal{T}(p)$ the index of the first such action in π. If p is not achieved, i.e., $p \notin I(\pi)$ holds, we can assign an arbitrary value in $\{0, \cdots, |A|\}$ to $\mathcal{T}(p)$. Given a delete-free task T^+ and its feasible and non-redundant plan π, we call the above assignment *a solution derived from* π.

We use the following fact in later proofs: a solution derived from a feasible solution satisfies (a) $\sum_{a' \in A \text{ s.t.} p \in \text{add}(a')} \mathcal{E}(a', p) \leq 1$ for any proposition p such that $\mathcal{U}(p) = 1$, and (b) $\sum_{a' \in A \text{ s.t.} p \in \text{add}(a')} \mathcal{E}(a', p) = 0$ for any proposition p such that $\mathcal{U}(p) = 0$.

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 IP(T^+) or LP(T^+) are embedded as a heuristic function in a forward-search planner and called for each state.

The objective function is defined as follows:

$$\text{minimize: } \sum_{a \in A} c(a)\mathcal{U}(a). \tag{1}$$

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

Finally we define the following six constraints.

(C1) $\forall p \in G$, $\mathcal{U}(p) = 1$. (The goals must be achieved).

(C2) $\forall a \in A$, $\forall p \in \text{pre}(a), \mathcal{U}(p) \geq \mathcal{U}(a)$. (Actions require their preconditions).

(C3) $\forall a \in A$, $\forall p \in \text{add}(a), \mathcal{U}(a) \geq \mathcal{E}(a, p)$. (An action can be the first achiever only if it is used).

(C4) $\forall p \in P$, $\mathcal{I}(p) + \sum_{a' \in A \text{ s.t.} p \in \text{add}(a')} \mathcal{E}(a', p) = \mathcal{U}(p)$. (If a proposition is achieved, it must be true in the initial state or is the effect of some action).

(C5) $\forall a \in A$, $\forall p \in \text{pre}(a)$, $\mathcal{T}(p) \leq \mathcal{T}(a)$. (Actions must be preceded by the satisfaction of their preconditions).

(C6) $\forall a \in A$, $\forall p \in \text{add}(a), \mathcal{T}(a) + 1 \leq \mathcal{T}(p) + (|A| + 1)(1 - \mathcal{E}(a, p))$. (If a is the first achiever of p, then a must precede p).

Now we can show that a solution of IP(T^+) derived from a feasible non-redundant plan of T^+ is feasible. For a variable $\mathcal{V}$ of IP(T^+), $\mathcal{V}_F$ describes the assignment of $\mathcal{V}$ on a solution F of IP(T^+).

Proposition 1. *Given a delete-free task T^+ and a feasible, non-redundant plan π for T^+, the solution F to* IP(T^+) *derived from π is a feasible solution to* IP(T^+).

Proof. F clearly satisfies constraint C1 since π satisfies $G \subseteq I(\pi)$.

Constraint C2 is not satisfied only if there exists an action $a \in A$ and a proposition $p \in \mathrm{pre}(a)$ such that $\mathcal{U}(a)_F = 1$ and $\mathcal{U}(p)_F = 0$. However, if $\mathcal{U}(a)_F = 1$, then $\mathcal{U}(p)_F = 1$ because π is a delete-free feasible plan and p has to be established at some point. We can show that F satisfies constraints C3 and C4 by similar arguments. If there exists an action $a \in A$ and a proposition $p \in \mathrm{add}(a)$ such that $\mathcal{E}(a,p)_F = 1$, $\mathcal{U}(a)_F = 1$ must hold according to the definition of F. In addition, if there exists a proposition p such that $\mathcal{U}(p)_F = 1$, there exists a first achiever $a \in A$ of p such that $\mathcal{E}(a,p)_F = 1$, or p is a member of the initial state I.

If an action $a \in A$ is a member of π, then all propositions in its precondition must be achieved before a is used. Hence, according to the definition of F, we have $\mathcal{T}(p)_F \leq \mathcal{T}(a)_F$ for any action a in the plan π. If an action $a \in A$ is not a member of π, then we have $\mathcal{T}(a)_F = |A|$. Thus, constraint C5 is satisfied for any action a not in the plan π, regardless of the values of $\mathcal{T}(p)_F$.

Finally F satisfies constraint C6 for any action $a \in A$ and for any proposition in its precondition $p \in \mathrm{pre}(a)$. If a is not the first achiever of p, i.e., $\mathcal{E}(a,p) = 0$, then constraint C6 is satisfied regardless of the values of $\mathcal{T}(p)_F$ and $\mathcal{T}(a)_F$. If a is the first achiever of p, then, according to the definition of F, we have $\mathcal{T}(p)_F = \mathcal{T}(a)_F + 1$, which satisfies constraint C6. □

In addition, 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 the following proposition.

Proposition 2. *Given a feasible solution F for $\mathrm{IP}(T^+)$, the action sequence $\pi = (a_0, \cdots, a_n)$ 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. First we show that π satisfies the condition (ii) of a plan (i.e., $G \subseteq I(\pi)$) using a proof by contradiction. Assume that there exists a proposition $g \in G$ that satisfies $g \notin I(\pi)$. Then, there exists no action achieving g in π. Since F is a solution to $\mathrm{IP}(T^+)$, $\mathcal{U}(g)_F = 1$ due to constraint C1. Since $g \notin I(\pi)$ implies $g \notin I$, $\mathcal{I}(g)_F = 0$. Therefore, to satisfy constraint C4, there must exist an action $a \in A$ such that $g \in \mathrm{add}(a)$ and $\mathcal{E}(a,g)_F = 1$. However, to satisfy constraint C3, $\mathcal{U}(a)_F = 1$ has to hold. This means $a \in \pi$, which 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 an 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 $a \in A$ such that $p \in \mathrm{add}(a)$, $\mathcal{U}(a)_F = 1$, and $\mathcal{E}(a,p)_F = 1$. However, to satisfy constraint C5, $\mathcal{T}(p) \leq \mathcal{T}(a_0)$ must be true, and $\mathcal{T}(a) + 1 \leq \mathcal{T}(p)$ has to hold to satisfy constraint C6. 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 $a \notin (a_0, \cdots, a_{i-1})$ such that $\mathcal{U}(a)_F = 1$ and $\mathcal{T}(a) < \mathcal{T}(a_i)$, contradicting the fact that a_i is the i-th action of the sequence π. □

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

The number of variables in $\mathrm{IP}(T^+)$ is $3|P| + 2|A| + \sum |\mathrm{add}(a)|$. The number of constraints is less than $2|P| + 2\sum_{a \in A} |\mathrm{pre}(a)| + 2\sum_{a \in A} |\mathrm{add}(a)|$. The number of terms is also $O(|P| + \sum |\mathrm{pre}(a)| + \sum |\mathrm{add}(a)|)$.

4. Enhanced IP Model

While $\mathrm{IP}(T^+)$ provides an IP model for exactly computing h^+, we shall see in Section 5 that $\mathrm{IP}(T^+)$ by itself is not competitive with previous methods for computing h^+. Thus, in this section, we introduce some variable elimination techniques and some modifications to constraints in order to speed up the computation h^+. As we will show in the experimental results, $\mathrm{IP}^{\mathrm{e}}(T^+)$, which incorporates these enhancements, computes h^+ significantly faster than $\mathrm{IP}(T^+)$. Some of the enhancements below are adopted into our IP framework from previous work in planning research. In particular, a landmark-based variable reduction method plays a key role.

Note that some of the enhancements introduce constraints that render some solutions of $\mathrm{IP}(T^+)$ mapped from feasible plans for T^+ infeasible. However, we show that in such cases, *at least one optimal plan will always remain valid in the enhanced model*, so the optimal cost of the enhanced model still corresponds to h^+.

4.1 Landmark-Based IP Model Reduction

A *landmark* is an element which needs to be used in every feasible solution (Hoffmann, Porteous, & Sebastia, 2004). We use two kinds of landmarks, called *fact landmarks* and *action landmarks* as in the work of Gefen and Brafman (2012). A fact landmark of a planning task T is a proposition that becomes true in some state of every feasible plan, and an action landmark for 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 for a proposition* p if l is a landmark for the task $\langle P, A, I, \{p\}\rangle$. Similarly we say that a landmark l is a *landmark for an action* a if l is a landmark for 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 for a proposition in the goal G, then we can substitute $\mathcal{U}(p) = 1$. Similarly, if an action a is an action landmark, then we can substitute $\mathcal{U}(a) = 1$. Landmark extraction and substitution clearly do not prune any feasible solutions of $\mathrm{IP}(T^+)$.

To actually extract the set of landmarks that satisfy the above intensional definitions, a landmark extraction algorithm is necessary. It is easy to see that given a feasible delete-free task, a proposition $p \in P$ is a fact landmark if and only if $p \in I$ holds or $\left\langle P, A \setminus A_p^{\mathrm{add}}, I \setminus \{p\}, G\right\rangle$ is infeasible, where $A_p^{\mathrm{add}} = \{a \mid p \in \mathrm{add}(a)\}$. Similarly an action $a \in A$ is an action landmark if and only if $\langle P, A \setminus \{a\}, I, G\rangle$ is infeasible. Hence, for each landmark candidate, we can test whether it is a landmark by checking the feasibility of the delete-free task which excludes that candidate. The feasibility of a delete-free task can be checked using the following, straightforward algorithm based on the delete-relaxed planning graph method by Hoffmann and Nebel (2001): For each fluent, let $e(p) \in \{0, 1\}$ represent whether p is achievable or not. For each action, let $e(a) \in \{0, 1\}$ represent whether the preconditions of a are satisfied or not. Initially, $e(p) = 1$ for all $p \in I$, $e(p) = 0$ for all other fluents. and $e(a) = 0$ for all a. At each step of the algorithm, for all actions for which $e(a) = 0$ and whose preconditions are satisfied; (1) set $e(a) = 1$, and (2) set $e(p) = 1$ for all $e \in add(a)$. The algorithm terminates when it reaches quiescence, i.e., no actions for which $e(a) = 0$ and whose preconditions are satisfied can be found. This takes at most $|A|$ steps. By repeating this feasibility check for all facts and actions, we have an algorithm that collects all fact landmarks and action landmarks satisfying the definitions above in $O(|T^+|^2)$-time.

If we were only interested in computing h^+ once, a straightforward method such as the one described above would be sufficient. However, since we intend to use our h^+-based models as heuristic functions for forward state-space search planning, the landmark extraction needs to be performed repeatedly during the search, so the efficiency of the extraction procedure is important. We experimented with several methods, and describe our most effective method below.

Our method for extracting landmarks is based on the method by Zhu and Givan (2003), who proposed a planning based propagation method for collecting causal landmarks. Their method was later generalized by Keyder et al. to an AND-OR graph based landmark extraction method (Keyder, Richter, & Helmert, 2010).

Zhu and Givan (2003) define a proposition p as a *causal landmark* if $\langle P, A \setminus A_p^{\rm pre}, I \setminus \{p\}, G\rangle$ is infeasible, where $A_p^{\rm pre} = \{a \mid p \in {\rm pre}(a)\}$. They focus on causal landmarks, ignoring other (non-causal) landmarks because they are nonessential (even misleading) from the point of view of guiding a search algorithm that uses a landmark-based heuristic. In contrast, we use landmarks in order to reduce the number of variables in our IP model of the delete relaxation. Thus, instead of focusing on causal landmarks and using Zhu and Givan's criteria, we seek a larger set of landmarks by slightly modifying the criterion for landmark detection. If $\langle P, A \setminus A_p^{\rm pre}, I \setminus \{p\}, G\rangle$ does not have a solution, then $\left\langle P, A \setminus A_p^{\rm add}, I \setminus \{p\}, G\right\rangle$ must also be infeasible, and furthermore, using $A_p^{\rm add}$ instead of $A_p^{\rm pre}$ can extract a larger set of fact landmarks. In addition, while Zhu and Givan used a forward propagation algorithm based on the layered planning graph of the delete-free task T^+, we use the following, open-list based propagation algorithm.

For each proposition p, we compute a set of fact landmarks for p, using an iterative method based on the following update equations characterizing fact landmarks:

- If p is a member of the initial state I, then $\{p\}$ is the set of fact landmarks to achieve p.
- If p is not a member of I, then the set of fact landmarks for p is $\{p\} \cup \bigcap_{a \in A \text{ s.t.} p \in {\rm add}(a)} ({\rm add}(a) \cup \bigcup_{p' \in {\rm pre}(a)} (\text{fact landmarks for } p'))$.

The pseudocode for this open list based propagation algorithm is shown in Algorithm 1. In the initialization phase, the candidate set for each proposition $p \notin I$ is set to P, and the fact landmarks for each $p \in I$ is set to $\{p\}$ (Lines 1-3). In addition, an action a is inserted into a FIFO queue Q if it satisfies ${\rm pre}(a) \subseteq I$ (Lines 4-7). The main loop of the iterative method is similar to the straightforward method described above. At each iteration, an action a is retrieved from Q, and the candidate set of fact landmarks is updated for each $p \in {\rm add}(a)$ based on the second equation (Lines 12-14). Moreover, the method memorizes the achievability of p (Line 11), and action a' is inserted into Q if all members of ${\rm pre(a')}$ are achievable and if the candidate set of $p' \in {\rm pre(a')}$ is changed (Lines 15-17). This process continues until Q becomes empty. For clarity and simplicity, some implementation details/optimizations are omitted from Algorithm 1, e.g., instead of literally inserting every member of P into $L[p]$ in Line 3, we use a single flag to represent "$L[p] = P$" Updating a candidate set always reduces the number of its elements, so this method always terminates. Unlike the simpler $O|T^+|^2$ algorithm described above, this algorithm is not complete (not all landmarks will be extracted). However, the soundness of this method is guaranteed by the following proposition.

Proposition 3. *Given a delete-free STRIPS planning task $\langle P, A^+, I, G\rangle$, assume all propositions in P can be achieved. Let $L(p)$ be the set of fact landmark candidates for p computed by some landmark extracting method. If*

(i) $L(p) = \{p\}$ *for* $p \in I$*, and*

(ii) $L(p) = \{p\} \cup \bigcap_{a \in A \text{ s.t.} p \in {\rm add}(a)} ({\rm add}(a) \cup \bigcup_{p' \in {\rm pre}(a)} L(p'))$ *for* $p \notin I$

Algorithm 1 Our landmark extracting method

```
 1: // $L[p]$ are sets of candidates of fact landmarks for $p \in P$.
 2: $L[p] \leftarrow P$ for each $p \notin I$;
 3: $L[p] \leftarrow \{p\}$ for each $p \in I$;
 4: $S \leftarrow I$;
 5: for $a \in A$ do
 6:    insert $a$ into a FIFO queue $Q$ if $\mathrm{pre}(a) \subseteq S$;
 7: end for
 8: while $Q$ is not empty do
 9:    retrieve an action $a$ from $Q$.
10:    for $p \in \mathrm{add}(a)$ do
11:       $S \leftarrow S \cup \{p\}$.
12:       $X \leftarrow L[p] \cap (\mathrm{add}(a) \cup \bigcup_{p' \in \mathrm{pre}(a)} L[p'])$;
13:       if $L[p] \neq X$ then
14:          $L[p] \leftarrow X$.
15:          for $a' \in A_p^{\mathrm{pre}}$ do
16:             insert $a'$ into $Q$ if $\mathrm{pre}(a') \subseteq S$ and $a' \notin Q$;
17:          end for
18:       end if
19:    end for
20: end while
21: // At this point, $L[p]$ contain sets of fact landmarks for $p \in P$.
```

are satisfied, then all elements of $L(p)$ are fact landmarks for p.

Proof. Assume that some proposition q satisfies $q \in L(p)$ and q is not a fact landmark for p. We have $p \neq q$ since any proposition is a fact landmark for itself. Then, $L(p)$ has more than one proposition, and from condition (i) and (ii), $p \notin I$ holds. Since q is not a landmark, there exists a non-empty feasible plan for the delete-free task $\langle P, A^+, I, \{p\}\rangle$ that does not achieve q.

Let $\pi = (a_0, \cdots, a_n)$ be such a plan, and let a_i be the action in π that achieves p first. We have $p \neq q$ as stated above, and we have $q \notin \mathrm{add}(a_i)$ since π does not achieve q. Hence, according to condition (ii), we have $q \in \bigcup_{p' \in \mathrm{pre}(a_i)} L(p')$. Let p' be a member of $\mathrm{pre}(a_i)$ that satisfies $q \in L(p')$. Since π is a feasible plan that does not achieve q, p' is achieved by π, and thus $p' \neq q$ holds. Then, $L(p')$ has more than one proposition, and again, $p' \notin I$ holds. Hence, $\pi' = (a_0, \cdots, a_{i-1})$ is a non-empty feasible plan for a delete-free task $\langle P, A^+, I, \{p'\}\rangle$ that does not achieve q.

This argument can be extended *ad infinitum*, but the length of π is clearly finite, so we have a contradiction. Thus, all members of $L(p)$ are fact landmarks for p for each proposition $p \in P$. $\square$

In addition to the fact landmarks which are extracted using the above procedure, our algorithm extracts action landmarks using the criterion: if a proposition p is a fact landmark of G, and if only one action a can achieve p, then a is used as an action landmark of G.

4.2 Relevance Analysis

Backchaining relevance analysis is widely used to eliminate propositions and actions that are irrelevant to 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, as noted by Haslum et al. (2012), it is sufficient to consider relevance with respect to only a subset of first achievers of an add effect. Although they defined a first achiever by achievability of a proposition, it is 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 for a. Let $\mathrm{fadd}(a)$ denote $\{p \in \mathrm{add}(a) \mid a \text{ is a first achiever of } p\}$. It is sufficient to use fadd instead of add in 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 $\mathcal{E}(a, p) = 0$. It is possible for a fact landmark fact to be irrelevant, in which case we set $\mathcal{U}(p) = 1$. While this variable elimination prunes some *feasible* solutions, it clearly does not prune any *optimal* solutions.

4.3 Dominated Action Elimination

In a delete-free task, if two actions have the same add effects, then it is clearly sufficient to use at most one of these two actions. This idea can be generalized to the following reduction, which eliminates useless (*dominated*) actions.

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

Proof. For any plan $\pi = (a_0, \cdots, a_{i-1}, a, a_{i+1}, \cdots, a_n)$ of T^+, we show that a sequence of actions $\pi' = (a_0, \cdots, a_{i-1}, a', a_{i+1}, \cdots, a_n)$ is also a feasible plan. Each proposition of $\mathrm{pre}(a')$ is a fact landmark for a, hence, if $\mathrm{pre}(a) \subseteq I((a_0, \cdots, a_{i-1}))$, then $\mathrm{pre}(a') \subseteq I((a_0, \cdots, a_{i-1}))$ also holds. By the definition of first achievers, $\mathrm{add}(a) \setminus \mathrm{fadd}(a) \subseteq I((a_0, \cdots, a_{i-1}))$, so we also have $I((a_0, \cdots, a_{i-1}, a)) \subseteq I((a_0, \cdots, a_{i-1}, a')$. Therefore $G \subseteq I(\pi')$ (π' is a feasible plan).

Finally, $c(\pi) \geq c(\pi')$ because $c(a) \geq c(a')$. Therefore, if a plan contains a, it is not optimal, or there exists another optimal plan which does not contain a. □

If there exists a dominated action a, we can eliminate a variable by setting $\mathcal{U}(a) = 0$. This variable elimination prunes some feasible solutions of $\mathrm{IP}(T^+)$. Moreover, it sometimes prunes some optimal solutions if $c(a) = c(a')$ holds for the condition (iii). However, as shown in the proof above, at least one optimal solution remains.

This is a slight generalization of a similar set of constraints by Robinson (2012)[Definition 5.3.4, p. 108] for a MaxSAT-based planner. Robinson's dominance condition checks whether (R1) $\mathrm{add}(a) \setminus I \subseteq \mathrm{add}(a') \setminus I$, (R2) $\mathrm{pre}(a') \setminus I \subseteq \mathrm{pre}(a) \setminus I$, and (R3) $c(a) \geq c(a')$. While our condition (iii) and (R3) are equivalent, our condition (i) is less strict than condition (R1) because instead of checking all add effects, condition (i) only tests whether the propositions for which a is a first achiever is subsumed by those of a'. Furthermore, our condition (ii) subsumes (R2) because if each proposition of $\mathrm{pre}(a')$ is a fact landmark for a, then if $\mathrm{pre}(a) \subseteq I((a_0, \cdots, a_{i-1}))$, $\mathrm{pre}(a') \subseteq I((a_0, \cdots, a_{i-1}))$ also holds, satisfying (R2).

4.4 Immediate Action Application

On a delete-free task T^+, some actions can be immediately applied to the initial state without affecting the optimality of the relaxed plan. We adopt immediate application of zero-cost actions (Gefen & Brafman, 2011) as well as immediate application of action landmarks (Gefen & Brafman, 2012). For a delete-free task T^+, if an action $a \in A$ satisfies $c(a) = 0$ and $\text{pre}(a) \subseteq I$, then a sequence made by placing a before an optimal plan for $\langle P, A \setminus \{a\}, I \cup \text{add}(a), G\rangle$ is an optimal plan for T^+. Similarly, if an action a is an action landmark for T^+ and a is applicable to I, a can be applied to I immediately.

In the IP(T^+) model, variables $\mathcal{T}(p)$ for $p \in I$ can be eliminated by substituting zero for their values. Given a sequence of immediately 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) $\forall p \in \text{pre}(a_i), \mathcal{U}(p) = 1$, (iv) $\forall p \in \text{add}(a_i) \setminus I((a_0, \cdots, a_{i-1})), \mathcal{U}(p) = 1, \mathcal{T}(p) = i$ and $\mathcal{E}(a_i, p) = 1$, and (v) $\forall p \in \text{add}(a_i) \setminus I((a_0, \cdots, a_{i-1})), \forall a \in A \setminus \{a_0, \cdots, a_i\}, \mathcal{E}(a, p) = 0$.

4.5 Iterative Application of Variable Eliminations

The variable elimination techniques described above can interact synergistically with each other resulting in a cascade of eliminations. Therefore, we used an iterative variable elimination algorithm which applies eliminations until quiescence. The order in which each elimination is applied is shown in Algorithm 2. A full landmark extraction pass after each variable elimination would be extremely expensive. Therefore, we perform a landmark extraction only once before the iterative application of the other eliminations.

Algorithm 2 Iterative Variable Elimination

```
relevance analysis;
landmark extraction;
While a variable can be eliminated do
    immediate action application;
    dominated actions elimination;
    relevance analysis;
```

4.6 Inverse Action Constraints

We define the following inverse relationship between a pair of actions for a delete-free task T^+.

Definition 1 (inverse action). *For two actions $a_1, a_2 \in A$, a_1 is an inverse action of a_2 if: (i) $\text{add}(a_1) \subseteq \text{pre}(a_2)$, and (ii) $\text{add}(a_2) \subseteq \text{pre}(a_1)$.*

By 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 the following fact.

Proposition 5. *Given a delete-free task T^+, let $\pi = (a_0, \cdots, a_n)$ be a feasible plan. If $a_i \in \pi$ is an inverse action of $a_j \in \pi$, and if $i < j$ holds, then $\pi' = (a_0, \cdots, a_{j-1}, a_{j+1}, \cdots, a_n)$ is also a feasible plan.*

Proof. Since π is a feasible plan for T^+, $\text{pre}(a_i) \subseteq I((a_0, \cdots, a_{i-1})) \subseteq I((a_0, \cdots, a_{j-1}))$. By the definition of inverse actions, $\text{add}(a_j) \subseteq \text{pre}(a_i)$ holds, and $\text{add}(a_j) \subseteq \text{pre}(a_i) \subseteq I((a_0, \cdots, a_{j-1})) =$

$I((a_0, \cdots, a_j))$. Hence $(a_{j+1}, \cdots, a_n)$ is applicable to $I((a_0, \cdots, a_{j-1}))$, and $G \subseteq I(\pi') = I(\pi)$. □

Corollary 2. *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.*

There are several possible ways to use the above proposition (e.g., $\mathcal{U}(a) + \mathcal{U}(a') \leq 1$, for all $a' \in \mathrm{inv}(a)$, where $\mathrm{inv}(a)$ is the set of inverse actions of a). In order to avoid adding a large number of constraints to the $\mathrm{IP}(T^+)$ model ($|A/2|^2$ in the worst case where half of the actions are inverses of the other), we modify constraint C2 as follows:

(C2a) $\forall a \in A,\ \forall p \in \mathrm{pre}(a), \mathcal{U}(p) - \sum_{a' \in \mathrm{inv}(a,p)} \mathcal{E}(a', p) \geq \mathcal{U}(a)$, where $\mathrm{inv}(a, p)$ denotes the set of inverse actions of a which have p as an add effect.

Proposition 6. *Given a delete-free task T^+, if* $\mathrm{IP}(T^+)$ *with constraint C2 has a feasible solution, then an optimal solution to* $\mathrm{IP}(T^+)$ *with constraint C2 is also feasible for* $\mathrm{IP}(T^+)$ *with constraint C2a.*

Proof. Let F^* be an optimal solution to $\mathrm{IP}(T^+)$ with constraint C2 derived from an optimal plan for T^+. Since F^* satisfies all the constraints of $\mathrm{IP}(T^+)$ with constraint C2, it suffices to show that F^* satisfies constraint C2a for any action $a \in A$ and proposition $p \in \mathrm{pre}(a)$.

Recall that a feasible solution derived from a feasible plan satisfies $\sum_{a' \in A \text{ s.t.} p \in \mathrm{add}(a')} \mathcal{E}(a', p) \leq 1$ for any proposition p such that $\mathcal{U}(p) = 1$, and it also satisfies $\sum_{a' \in A \text{ s.t.} p \in \mathrm{add}(a')} \mathcal{E}(a', p) = 0$ for any proposition p such that $\mathcal{U}(p) = 0$. Since $\sum_{a' \in A \text{ s.t.} p \in \mathrm{add}(a')} \mathcal{E}(a', p) \geq \sum_{a' \in \mathrm{inv}(a,p)} \mathcal{E}(a', p)$ for any action $a \in A$ and proposition $p \in \mathrm{pre}(a)$, F^* clearly satisfies constraint C2a if $\mathcal{U}(p)_{F^*} = 1$ and $\mathcal{U}(a)_{F^*} = 0$, or if $\mathcal{U}(p)_{F^*} = 0$ and $\mathcal{U}(a)_{F^*} = 0$ hold.

To show that $\sum_{a' \in \mathrm{inv}(a,p)} \mathcal{E}(a', p)_{F^*} = 0$ holds when $\mathcal{U}(a)_{F^*} = \mathcal{U}(p)_{F^*} = 1$, assume there exists an action $a' \in \mathrm{inv}(a, p)$ such that $\mathcal{E}(a', p)_{F^*} = 1$. According to constraint C3, $\mathcal{U}(a')_{F^*} = 1$. However, since F^* is derived from an optimal plan for T^+, there must exist an optimal plan for T^+ that contains both a and a'. This contradicts Corollary 2.

Since F^* is a feasible solution, there does not exist any action $a \in A$ and proposition $p \in \mathrm{pre}(a)$ such that $\mathcal{U}(a)_{F^*} = 1$ and $\mathcal{U}(p)_{F^*} = 0$. Hence F^* satisfies constraint C2a for any $a \in A$ and $p \in \mathrm{pre}(a)$. □

4.7 $\mathrm{IP}^{\mathrm{e}}(T^+)$: The Enhanced IP Model for h^+

We define $\mathrm{IP}^{\mathrm{e}}(T^+)$ as the integer programming model that is the result of first adding the inverse action constraints described in Section 4.6 to the basic $\mathrm{IP}(T^+)$ model and then applying the iterative reduction algorithm in Algorithm 2 (which applies the reductions in Sections 4.1-4.4) until quiescence. As previously noted, $\mathrm{IP}^{\mathrm{e}}(T^+)$ computes h^+. As we shall see below, the cumulative effects of these enhancements is quite significant, resulting in a much more practical IP model for computing h^+. See Table 1 for a summary of the relationship between $\mathrm{IP}^{\mathrm{e}}(T^+)$ and $\mathrm{IP}(T^+)$.

5. Experimental Evaluation of IP Models for Delete-Free Planning (Exact Computation of h^+)

In this section, we evaluate the effectiveness of our integer programming model of the delete relaxation as a method for solving delete-free tasks and computing h^+ exactly. We evaluate the following models:

- $\mathrm{IP}(T^+)$: our basic IP model (Section 3).
- $\mathrm{IP}(T^+)$+LM: $\mathrm{IP}(T^+)$ with the landmark-based variable reduction method (Section 4.1).
- $\mathrm{IP}^{\mathrm{e}}(T^+)$: the enhanced model which includes all of the enhancements described in Sections 4.1-4.6 which are designed to speed up the computation of h^+ (landmark-based reduction, relevance analysis, dominated action elimination, immediate action application, inverse action constraints).

We emphasize that (unlike the other models which will be evaluated in later sections) all of these IP models compute h^+ *exactly*.

Following previous work on solvers for delete-free problems, our main results are based on an evaluation using delete-free versions of standard IPC benchmark problems (Section 5.1). In addition, in Section 5.2, we also present results of a much smaller scale study on a set of natural, delete-free problems from systems biology (Gefen & Brafman, 2011).

5.1 Evaluation on Delete-Free Versions of IPC Benchmark Instances

Following the methodology for evaluating delete-free planning in previous work (Haslum et al., 2012; Pommerening & Helmert, 2012; Gefen & Brafman, 2012), we evaluate our IP models by solving International Planning Contest (IPC) benchmark instances for which the delete effects of all actions are ignored. Below, all experiments used the CPLEX 12.61 solver to solve integer and linear programs. All experiments were single-threaded and executed on a Xeon E5-2680, 2.8GHz.

Because previous work on computing h^+ has been evaluated using several different sets of experimental settings (different CPU limits and different problem instances), we present the results of 4 sets of comparisons. In the first 3 sets of comparisons, we compare benchmark results reported in previous publications with results obtained by running our solvers on the same problem instances, while the fourth set of results compares our models with an improved implementation of the minimal hitting set based approach (Haslum et al., 2012) by one of the the original authors.

- Comparison with the results by Pommerening and Helmert (2012) (experimental setup described in Section 5.1.1, results shown in Table 2).
- Comparison with the results by Gefen and Brafman (2012) (experimental setup described in Section 5.1.2, results shown in Table 3).
- Comparison with the results by Haslum et al. (2012) (experimental setup described in Section 5.1.2, results shown in Table 4).
- Comparison with HST/CPLEX, an improved implementation of the algorithm in (Haslum et al., 2012) (experimental setup described in Section 5.1.3, results shown in Table 5 and Figures 2-3).

The results copied from previous work (Pommerening & Helmert, 2012; Haslum et al., 2012; Gefen & Brafman, 2012) in Tables 2-4 were obtained using hardware available several years ago when these original papers were written, while our results for $\mathrm{IP}(T^+)$, $\mathrm{IP}^{\mathrm{e}}(T^+)$, and HST/CPLEX were obtained with slightly more recent hardware. Since coverage is a coarse metric based on binary results (solved/unsolved), it can be significantly impacted by differences in machine speed, e.g., if many problems are at the threshold such that a slightly faster machine (equivalent to running slightly longer) results in many more instances being solved. In order to eliminate the possibility that improvements in hardware since 2010 (when the first of the results we compared against were published) explain the improvements obtained using our approach, we also include results of running our best IP model ($\mathrm{IP}^{\mathrm{e}}(T^{+})$) with a significantly *shorter* CPU time limit than the previous experiments, in addition to results that use the same CPU time limit as previous experiments.

5.1.1 COMPARISON WITH RESULTS BY POMMERENING AND HELMERT (2012) ON DELETE-FREE VERSIONS OF IPC BENCHMARKS

The first comparison is with the results by Pommerening and Helmert (2012). Table 2 shows the results of running $\mathrm{IP}(T^{+})$, $\mathrm{IP}(T^{+})$+LM, and $\mathrm{IP}^{\mathrm{e}}(T^{+})$ with a 5 minute time limit and 2GB memory limitation. Coverage (# of problem instances solved) on each domain is shown. The columns where the solver name contains "PH12" in Table 2 are copied from the paper by Pommerening and Helmert (2012). "FD/PH12" is Fast Downward using A* and the LM-cut heuristic applied to the delete-relaxed problems, "BC/PH12" is the hitting set based approach by Bonet and Castillo (2011), and "BnB/PH12" and "IDA*/PH12" are the best performing strategies using the incremental LM-cut heuristic for delete-free problems proposed by Pommerening and Helmert (2012). Pommerening and Helmert obtained their results using a AMD Opteron 2356 processor with a 2GB memory limit and 5 minute time limit.

Table 2 includes a column "$\mathrm{IP}^{\mathrm{e}}(T^{+})$/1min", which shows the results for 1-minute runs of $\mathrm{IP}^{\mathrm{e}}(T^{+})$. All other columns in Table 4 are for 5 minute runs.

5.1.2 COMPARISONS WITH RESULTS BY GEFEN AND BRAFMAN (2012) AND HASLUM ET AL. (2012) ON DELETE-FREE VERSIONS OF IPC BENCHMARKS

Next, we evaluated our h^{+} solvers with previous results that were obtained with a 30-minute time limit and 2GB memory limit. Table 3 compares $\mathrm{IP}(T^{+})$, $\mathrm{IP}(T^{+})$+LM, and $\mathrm{IP}^{\mathrm{e}}(T^{+})$ with some results from (Gefen & Brafman, 2012, p. 62, Table 2). The LM-cut/GB12 column is A* with the LM-cut heuristic (Helmert & Domshlak, 2009) applied directly to delete-free instances in order to compute h^{+}. The LM-cut+Pruning/GB12 column is A* with LM-cut using the pruning techniques for delete-free instances proposed by Gefen and Brafman (2012). Table 4 compares $\mathrm{IP}(T^{+})$ and $\mathrm{IP}^{\mathrm{e}}(T^{+})$ with some results by Haslum et al. (2012, p. 356, Table 1). The BC/HST12 column is the method by Bonet and Castillo (2011). The ML/HST12 column is the minimal landmark method proposed by Haslum et al.. In the original work by Haslum et al. (2012), the minimum-cost hitting set problem was solved using a specialized branch-and-bound algorithm, and the ML/HST12 column reflects the performance of this original algorithm. However, the Minimal Landmark method was later significantly improved by replacing the hitting set solver with a CPLEX-based solver (Haslum, 2014b), so Table 4 also includes the HST/CPLEX column, which shows the results of Minimal Landmark method using the CPLEX hitting set solver. We obtained these HST/CPLEX results by running the HST/CPLEX code on the same machine used to run our IP models.

Table 4 includes a column "$\mathrm{IP}^{\mathrm{e}}(T^{+})$/5min", which shows the results for 5-minute runs of $\mathrm{IP}^{\mathrm{e}}(T^{+})$ (all other columns in Table 4 are for 30 minute runs).

Note that in Table 4, the instances from IPC2008 and IPC2011 are from the sequential satisfying track (i.e., "-sat08" and "-sat11" in the domain names), in accordance with the original paper (Haslum et al., 2012).

5.1.3 COMPARISON WITH HST/CPLEX ON DELETE-FREE VERSIONS OF IPC BENCHMARKS

The most detailed comparison is with an improved implementation of the hitting-set based method of Haslum et al. (2012). Although the original version of this algorithm used a problem-specific branch-and-bound method to solve the hitting set problems, we used a more recent version of Haslum's h^+ solver (source dated 2014-1-17), configured to use CPLEX 12.61 to solve the hitting set subproblem. This configuration is abbreviated as "HST/CPLEX". As shown in Table 4, HST/CPLEX significantly outperforms the original HST implementation described in (Haslum et al., 2012), and compares favorably vs. other previous methods.

Tables 5-6 and Figures 2-3 compare IP(T^+), IP$^e(T^+)$, IP(T^+)+LM, and HST/CPLEX on 1376 IPC benchmark instances. All algorithms were run with a 2GB memory limit. Table 5 shows results with a 30 minute time limit, while Table 6 shows results with a 5 minute time limit. Tables 5 and 6 compares coverage and runtimes per domain, while Figure 2 compares the cumulative number of instances solved as a function of time, and Figure 3 compares the runtimes of all individual instances.

In contrast to the previous set of experiments described in Section 5.1.2, we used optimal track instances ("-opt08" and "-opt11" in the domain names) when both satisficing and optimal track instances were available in the benchmark sets. This is because in the subsequent sections, we focus on applying our models as the basis for heuristics for forward-search, cost-optimal planning.

5.1.4 DISCUSSION OF RESULTS ON DELETE-FREE VERSIONS OF IPC BENCHMARKS

Not surprisingly, the basic IP(T^+) model is not competitive with previous state-of-the-art methods that were specifically developed for computing h^+ (Haslum et al., 2012; Pommerening & Helmert, 2012). However, Table 3 shows that the basic IP(T^+) model is at least competitive with A* with LM-cut enhanced with Gefen and Brafman's pruning methods for delete-free instances ("Prune/GB12"). IP(T^+) also significantly outperforms standard A* with LM-cut (Table 3, "LM-cut/GB12" and Table 2, "FD/PH12").

On the other hand, enhancing IP(T^+) with our landmark-based model reduction method results in significant improvement, and IP(T^+)+LM is competitive with all previous methods except for HST/CPLEX.

The IP$^e(T^+)$ model, which includes all of the enhancement described in Section 4 for reducing the model in order to compute h^+ faster, performs very well overall, and is competitive with all previous methods. For example, in Table 4, IP$^e(T^+)$ has the highest coverage (or is tied for highest) on 19/28 domains. Table 5, Figure 2, and Figure 3 show that while IP$^e(T^+)$ and HST/CPLEX have similar coverage with a 30-minute time limit, IP$^e(T^+)$ tends to be somewhat faster overall. However, there is no clear dominance relationship between IP$^e(T^+)$ and HST/CPLEX, since there are some domains where IP$^e(T^+)$ clearly performs better (e.g., rovers, satellite, freecell) , and other domains where HST/CPLEX performs better (e.g., airport, pegsol, scanalyzer, transport). Thus, the IP-based approach and minimal landmark approaches seem to have complementary strengths with respect to solving delete-free problems.

Aside from coverage, Figure 3 shows that many delete-free instances are solved much faster by IP$^e(T^+)$ than HST/CPLEX. The difference between solving an "easy" delete-free instance in 0.1 vs. 0.5 seconds may not seem very important if we only need to solve the instance once. However, the speed difference between IP$^e(T^+)$ and HST/CPLEX on such easy delete-free instances has a significant implication when we consider using the h^+ solvers as heuristic functions for A*-based planners, where we may need to solve delete-free problems many thousands of times in the course of a single A* search. As a result, we see below in Section 7, A* using $\mathrm{IP^e}(T^+)$ as a heuristic significantly outperforms A* using HST/CPLEX as a heuristic.

In order to eliminate the possibility that CPU speed differences account for the qualitative improvements in coverage obtained by our IP models compared to previously published results, Table 2 includes a column "$\mathrm{IP^e}(T^+)$/1min", which is the result for 1-minute runs of $\mathrm{IP^e}(T^+)$, and Table 4 includes a column "$\mathrm{IP^e}(T^+)$/5min", which is the result for 5-minute runs of $\mathrm{IP^e}(T^+)$ In effect, these simulate machines that run at 1/5 and 1/6 (respectively) of the speed of the machine we used in our experiments in Tables 2 and 4. This more than offsets improvements in single-core CPU performance between 2010-2015. The coverage achieved by $\mathrm{IP^e}(T^+)$/1min (753) in Table 2 is higher than that of all other solvers in Table 2 which were given 5 minutes. Similarly, the coverage achieved $\mathrm{IP^e}(T^+)$/5min (847) in Table 4 is higher than that of than all other solvers in Table 4 which were given 30 minutes.

Therefore, overall, $\mathrm{IP^e}(T^+)$ is competitive with previous state-of-the-art delete-free solvers, and our results indicate that direct computation of h^+ using integer programming is a viable approach, at least for computing each delete-free task once.

5.2 Comparison with HST/CPLEX on Minimal Seed Set Problem

To assess the performance of our best IP model, $\mathrm{IP^e}(T^+)$ on a natural, delete-free task, we also compared $\mathrm{IP^e}(T^+)$ with HST/CPLEX on a set of minimal seed set problem instances from systems biology (Gefen & Brafman, 2011). These consist of 22 instances originally evaluated by Gefen and Brafman, as well as three additional versions of these 22 instances which were also provided by the original authors, where each version uses a different set of action costs (Gefen & Brafman, 2011, p. 322), for a total of $22 \times 4 = 88$ instances. The solvers were run with a 1 hour CPU time limit per instance and a 2GB RAM limit.

Figure 4 shows a scatter plot comparing the runtimes on each problem instance. The coverage of $\mathrm{IP^e}(T^+)$ was 87 instances, while the coverage of HST/CPLEX was 88 instances. On one hand, Figure 4 shows that the majority of instances were solved significantly faster by $\mathrm{IP^e}(T^+)$, and $\mathrm{IP^e}(T^+)$ solves 22 instances more than 10 times faster than HST/CPLEX. On the other hand, there was one instance on which HST/CPLEX was more than 10 times faster than $\mathrm{IP^e}(T^+)$, and there was one instance which was solved in 40.7 seconds by HST/CPLEX but was not solved within the time limit by $\mathrm{IP^e}(T^+)$ (The "dre" instance with the "type 2" preprocessing by Gefen & Brafman, 2011, p. 322).

6. Relaxations of the h^+ Models

Although delete-free planning problems are interesting in their own right, our main motivation for developing an efficient IP model for delete-free problems is to be able to use it as the basis for a heuristic function for a forward-state space search based domain-independent planner. So far, we have presented $\mathrm{IP}(T^+)$, a basic IP model which computes h^+, and then proposed $\mathrm{IP^e}(T^+)$, which incorporates a number of enhancements which, as shown in the experimental results in Section 5, significantly increase the scalability of the model and provide a new approach to computing h^+ which is competitive with the previous state-of-the-art methods. It is possible to simply use $\mathrm{IP^e}(T^+)$ as the heuristic function for a forward search based planner. However, as shown in Section 5, computing h^+ remains relatively expensive even using $\mathrm{IP^e}(T^+)$, which is not surprising, given that

					(Pommerening & Helmert, 2012, Table 2)			
	$\mathrm{IP}(T^+)$	$\mathrm{IP}(T^+)$+LM	$\mathrm{IP}^e(T^+)$	$\mathrm{IP}^e(T^+)$/1min	FD/PH12	BC/PH12	BnB/PH12	IDA*/PH12
Domain (# problems)	solved	solved	solved	solved	solved	solved	solved	solved
airport(50)	22	36	36	35	34	**50**	**50**	**50**
blocks(35)	**35**	**35**	**35**	**35**	**35**	**35**	**35**	**35**
depot(22)	6	19	**21**	**21**	7	5	14	14
driverlog(20)	14	14	**15**	14	14	2	**15**	**15**
freecell(80)	11	17	**80**	**80**	6	1	2	3
grid(5)	0	4	**5**	**5**	1	1	2	2
gripper(20)	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**
logistics00(28)	24	**28**	**28**	**28**	23	26	**28**	**28**
logistics98(35)	8	21	**27**	24	9	7	16	15
miconic(150)	**150**	**150**	**150**	**150**	**150**	**150**	**150**	**150**
no-mprime(35)	15	20	**31**	30	27	14	27	26
no-mystery(30)	15	21	**30**	28	26	16	28	28
openstacks-opt08(30)	2	**30**	**30**	**30**	5	0	5	4
pathways-noneg(30)	**30**	**30**	**30**	**30**	5	4	5	5
pipes-notankage(50)	8	13	11	10	17	3	18	**19**
pipes-tankage(50)	5	9	9	9	**10**	2	9	**10**
psr-small(50)	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**
rovers(40)	**40**	**40**	**40**	**40**	13	12	19	19
satellite(36)	31	30	**34**	**34**	6	6	8	9
tpp(30)	11	24	**30**	**30**	13	12	23	24
trucks(30)	**30**	**30**	**30**	**30**	7	3	9	9
zenotravel(20)	14	14	**20**	**20**	13	8	13	13
Total coverage (876)	541	655	762	753	491	427	546	548
# Best Domains	7	9	19	15	5	5	7	9

Table 2: Coverage (# of instances solved) for delete-free problems (exact computation of h^+). 5-minute time limit (except for $\mathrm{IP}^e(T^+)$/1min which was run with a 1-minute time limit), 2GB RAM. Comparison with data from Table 2 in the paper by Pommerening and Helmert (2012). "# Best domains" is the number of domains for which a each solver achieves the highest coverage (including ties).

				(Gefen & Brafman, 2012, Table 2)	
	$\mathrm{IP}(T^+)$	$\mathrm{IP}(T^+)$+LM	$\mathrm{IP}^e(T^+)$	LM-cut/GB12	Prune/GB12
Domain (# problems)	solved	solved	solved	solved	solved
blocks(35)	**35**	**35**	**35**	**35**	**35**
depot(22)	8	19	**21**	7	12
driverlog(20)	14	14	**15**	14	**15**
freecell(80)	12	20	**80**	6	2
gripper(20)	**20**	**20**	**20**	**20**	**20**
logistics00(28)	24	**28**	**28**	23	**28**
logistics98(35)	8	23	**28**	10	16
miconic(150)	**150**	**150**	**150**	**150**	**150**
no-mystery(30)	21	23	**30**	26	26
pipesworld-notankage(50)	11	**17**	**17**	**17**	9
pipesworld-tankage(50)	7	9	9	**10**	9
rovers(40)	**40**	**40**	**40**	13	23
Total coverage (560)	350	398	473	331	345
# Best Domains	4	6	11	5	5

Table 3: Coverage (# of instances solved) for delete-free problems (exact computation of h^+). 30-minute time limit, 2GB RAM. Comparison with data from Table 2 in the paper by Gefen and Brafman (2012).

							(Haslum et al, 2012, Table 2)	
	$\mathrm{IP}(T^+)$	$\mathrm{IP}(T^+)$+LM	$\mathrm{IP}^{\mathrm{e}}(T^+)$	HST/CPLEX	$\mathrm{IP}^{\mathrm{e}}(T^+)$ 5min	HST/CPLEX 5min	ML/HST12	BC/HST12
Domain (# problems)	solved	solved	solved	solved	solved	solved	solved	solved
airport(50)	22	40	39	**50**	36	**50**	**50**	**50**
barman-sat11(20)	7	8	9	**20**	6	**20**	18	5
blocks(35)	**35**	**35**	**35**	**35**	**35**	**35**	**35**	**35**
depot(22)	8	19	**21**	20	**21**	20	18	12
driverlog(20)	14	14	**15**	14	**15**	14	13	8
elevators-sat08(30)	1	5	**30**	**30**	**30**	**30**	27	11
floortile-sat11(20)	19	**20**	**20**	12	19	12	12	9
freecell(80)	12	20	**80**	76	**80**	48	17	0
gripper(20)	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**
logistics98(35)	8	23	**28**	20	27	18	15	6
logistics00(28)	24	**28**	**28**	**28**	**28**	**28**	27	27
miconic(150)	**150**	**150**	**150**	**150**	**150**	**150**	**150**	99
no-mprime(35)	20	23	**34**	31	31	26	28	17
nomystery-sat11(20)	11	13	**19**	7	**19**	4	5	4
parcprinter-08(30)	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**
pegsol-08(30)	25	24	26	**30**	25	**30**	**30**	**30**
pipesworld-notankage(50)	11	17	17	**24**	11	17	20	9
pipesworld-tankage(50)	7	9	9	10	9	10	**15**	6
psr-small(50)	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**
rovers(40)	**40**	**40**	**40**	32	**40**	31	18	19
satellite(36)	31	31	**34**	14	**34**	11	8	5
scanalyzer-08(30)	10	10	10	**21**	9	16	15	4
sokoban-sat08(30)	25	29	29	**30**	29	**30**	**30**	**30**
transport-sat08(30)	2	3	7	**15**	6	12	6	6
trucks(30)	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**
visitall-sat11(20)	8	7	8	**16**	7	10	2	0
woodworking-sat08(30)	29	**30**	**30**	29	**30**	29	19	9
zenotravel(20)	15	15	**20**	14	**20**	12	13	10
Total coverage (1041)	664	743	868	858	847	793	721	541
# Best Domains	7	10	19	16	16	12	10	8

Table 4: Coverage (# of instances solved) for delete-free problems (exact computation of h^+). 30-minute time limit (except for $\mathrm{IP}^{\mathrm{e}}(T^+)$/5min and HST/CPLEX/5min which were run with a 5-minute time limit), 2GB RAM. Comparison with data from Table 2 in the paper by Haslum et al. (2012).

	IP(T^+)/30min		IP(T^+)+LM/30min		IP^e(T^+)/30min		HST/CPLEX/30min	
Domain (# problems)	solved	time	solved	time	solved	time ± sd	solved	time ± sd
airport(50)	22	253.97	40	173.58	39	134.68 ± 452.99	**50**	**9.99 ± 36.34**
barman-opt11(20)	8	1616.97	8	1522.41	**20**	14.29 ± 40.80	**20**	**0.04 ± 0.08**
blocks(35)	**35**	0.08	**35**	**0.00**	**35**	**0.00 ± 0.00**	**35**	**0.00 ± 0.00**
depot(22)	8	151.07	19	12.75	**21**	**0.92 ± 1.90**	20	3.50 ± 8.70
driverlog(20)	14	19.05	14	15.77	**15**	**5.47 ± 18.32**	14	17.30 ± 56.93
elevators-opt08(30)	2	294.94	20	201.74	**30**	0.38 ± 0.46	**30**	**0.09 ± 0.07**
elevators-opt11(20)	1	525.76	13	179.16	**20**	0.32 ± 0.42	**20**	**0.07 ± 0.04**
floortile-opt11(20)	**20**	4.67	**20**	1.76	**20**	**1.08 ± 3.02**	15	54.56 ± 193.72
freecell(80)	12	130.82	20	259.96	**80**	**0.32 ± 0.21**	76	320.71 ± 433.35
grid(5)	0	0	4	5.59	**5**	6.50 ± 11.29	**5**	**1.41 ± 1.61**
gripper(20)	**20**	0.02	**20**	0.02	**20**	**0.00 ± 0.00**	**20**	0.01 ± 0.01
logistics98(35)	8	194.01	23	89.77	**28**	**39.07 ± 132.25**	20	146.85 ± 339.48
logistics00(28)	24	12.21	**28**	0.03	**28**	**0.01 ± 0.02**	**28**	0.03 ± 0.06
miconic(150)	**150**	0.08	**150**	0.09	**150**	**0.01 ± 0.01**	**150**	0.04 ± 0.05
no-mprime(35)	20	202.01	23	221.48	**34**	**53.06 ± 132.87**	31	106.60 ± 242.37
no-mystery(30)	21	187.66	23	129.89	**30**	12.84 ± 44.49	**30**	**12.43 ± 29.27**
nomystery-opt11(20)	13	180.88	17	224.40	**20**	**0.11 ± 0.11**	8	0.36 ± 0.49
openstacks(30)	5	114.55	25	82.48	**30**	**0.39 ± 1.09**	27	81.80 ± 258.73
openstacks-opt08(30)	3	506.63	**30**	0.08	**30**	**0.01 ± 0.01**	**30**	0.04 ± 0.04
openstacks-opt11(20)	0	0	**20**	0.04	**20**	**0.01 ± 0.01**	**20**	0.03 ± 0.02
parcprinter-08(30)	**30**	0.08	**30**	0.04	**30**	**0.02 ± 0.01**	**30**	0.07 ± 0.12
parcprinter-opt11(20)	**20**	0.06	**20**	0.03	**20**	**0.01 ± 0.01**	**20**	0.04 ± 0.05
parking-opt11(20)	2	529.75	18	172.21	**20**	**0.30 ± 0.23**	**20**	15.97 ± 30.89
pathways-noneg(30)	**30**	1.50	**30**	1.13	**30**	**0.05 ± 0.03**	**30**	2.55 ± 3.08
pegsol-08(30)	25	229.13	24	39.01	26	40.72 ± 126.79	**30**	**0.01 ± 0.01**
pegsol-opt11(20)	13	360.87	14	105.91	15	86.91 ± 183.21	**20**	**0.01 ± 0.01**
pipesworld-notankage(50)	11	370.96	17	198.51	17	221.80 ± 306.90	**24**	**223.55 ± 358.14**
pipesworld-tankage(50)	7	154.58	9	22.87	9	18.39 ± 44.42	**10**	**4.32 ± 11.94**
psr-small(50)	**50**	0.03	**50**	0.02	**50**	**0.01 ± 0.05**	**50**	**0.01 ± 0.05**
rovers(40)	**40**	11.77	**40**	0.34	**40**	**0.13 ± 0.22**	32	34.36 ± 123.88
satellite(36)	31	35.88	31	38.40	**34**	**0.96 ± 1.64**	14	205.10 ± 384.71
scanalyzer-08(30)	10	306.24	10	292.41	10	86.52 ± 173.64	**21**	**242.91 ± 460.55**
scanalyzer-opt11(20)	7	442.44	7	439.54	7	129.49 ± 213.41	**13**	**338.77 ± 536.07**
sokoban-opt08(30)	29	34.12	29	0.61	**30**	56.97 ± 305.42	**30**	**0.07 ± 0.12**
sokoban-opt11(20)	**20**	39.14	**20**	0.47	**20**	0.23 ± 0.28	**20**	**0.07 ± 0.13**
tpp(30)	13	256.03	24	55.71	**30**	**4.58 ± 9.54**	28	142.13 ± 272.08
transport-opt08(30)	4	289.63	4	45.00	15	151.31 ± 421.56	**27**	**100.16 ± 146.57**
transport-opt11(20)	0	0	0	0	16	203.80 ± 424.60	**20**	**18.30 ± 35.03**
trucks(30)	**30**	1.94	**30**	0.70	**30**	**0.03 ± 0.02**	**30**	1.32 ± 2.10
visitall-opt11(20)	**20**	3.97	**20**	1.76	**20**	1.07 ± 2.93	**20**	**0.21 ± 0.38**
woodworking-opt08(30)	**30**	2.04	**30**	0.52	**30**	**0.02 ± 0.01**	**30**	0.15 ± 0.27
woodworking-opt11(20)	**20**	2.40	**20**	0.47	**20**	**0.02 ± 0.01**	**20**	0.09 ± 0.07
zenotravel(20)	15	35.54	15	36.69	**20**	**3.21 ± 9.13**	14	179.65 ± 453.63
Total coverage (1376)	843		1044		1214		1202	
# Best Domains	14		17		34		31	

Table 5: Detailed comparison of IP(T^+), IP(T^+)+LM, IP^e(T^+), and HST/CPLEX on 1376 delete-free tasks (exact computation of h^+). 30-minute time limit, 2GB RAM. Coverage and mean ± standard deviation of runtimes (average of successful runs only, excludes unsuccessful runs).

	$\text{IP}(T^+)$/5min		$\text{IP}(T^+)$+LM/5min		$\text{IP}^{\text{e}}(T^+)$/5min		HST/CPLEX/5min	
Domain (# problems)	solved	time ± sd	solved	time	solved	time ± sd	solved	time ± sd
airport(50)	22	0.82	36	0.33	36	4.10 ± 23.83	**50**	**9.44 ± 34.94**
barman-opt11(20)	0	0	0	0	**20**	13.60 ± 38.23	**20**	**0.04 ± 0.08**
blocks(35)	**35**	0.08	**35**	**0.00**	**35**	**0.00 ± 0.00**	**35**	**0.00 ± 0.00**
depot(22)	6	29.35	19	12.85	**21**	**0.93 ± 1.95**	20	3.47 ± 8.57
driverlog(20)	14	17.33	14	17.04	**15**	**5.86 ± 19.83**	14	17.03 ± 56.02
elevators-opt08(30)	1	25.40	16	41.09	**30**	0.39 ± 0.47	**30**	**0.08 ± 0.07**
elevators-opt11(20)	0	0	11	30.13	**20**	0.31 ± 0.41	**20**	**0.07 ± 0.04**
floortile-opt11(20)	**20**	4.74	**20**	1.64	**20**	**1.05 ± 2.93**	14	2.80 ± 4.21
freecell(80)	11	73.07	17	43.14	**80**	**0.30 ± 0.20**	48	60.87 ± 80.81
grid(5)	0	0	4	5.39	**5**	6.35 ± 11.05	**5**	**1.37 ± 1.54**
gripper(20)	**20**	0.02	**20**	0.02	**20**	**0.00 ± 0.00**	**20**	0.01 ± 0.00
logistics98(35)	8	20.45	21	19.56	**27**	**13.94 ± 36.73**	18	34.28 ± 67.68
logistics00(28)	24	11.64	**28**	0.03	**28**	**0.01 ± 0.02**	**28**	0.03 ± 0.06
miconic(150)	**150**	0.08	**150**	0.08	**150**	**0.01 ± 0.01**	**150**	0.04 ± 0.05
no-mprime(35)	15	28.01	20	30.02	**31**	**14.91 ± 51.52**	26	11.27 ± 23.42
no-mystery(30)	15	9.35	21	27.74	**30**	**12.04 ± 41.80**	**30**	12.88 ± 30.69
nomystery-opt11(20)	11	37.85	14	42.15	**20**	**0.10 ± 0.10**	8	0.34 ± 0.46
openstacks(30)	5	66.39	24	31.37	**30**	**0.37 ± 1.00**	24	12.20 ± 35.90
openstacks-opt08(30)	2	16.89	**30**	0.08	**30**	**0.01 ± 0.01**	**30**	0.04 ± 0.04
openstacks-opt11(20)	0	0	**20**	0.04	**20**	**0.01 ± 0.01**	**20**	0.03 ± 0.02
parcprinter-08(30)	**30**	0.07	**30**	0.03	**30**	**0.01 ± 0.01**	**30**	0.07 ± 0.11
parcprinter-opt11(20)	**20**	0.05	**20**	0.03	**20**	**0.01 ± 0.01**	**20**	0.04 ± 0.05
parking-opt11(20)	0	0	15	69.50	**20**	**0.29 ± 0.22**	**20**	15.07 ± 28.61
pathways-noneg(30)	**30**	1.53	**30**	1.14	**30**	**0.04 ± 0.03**	**30**	2.47 ± 2.92
pegsol-08(30)	22	74.55	23	16.60	25	16.24 ± 37.01	**30**	**0.01 ± 0.01**
pegsol-opt11(20)	9	131.78	12	36.09	13	16.65 ± 20.17	**20**	**0.01 ± 0.01**
pipesworld-notankage(50)	8	5.53	13	37.58	11	16.55 ± 52.18	**17**	**21.71 ± 35.68**
pipesworld-tankage(50)	5	31.71	9	21.57	9	14.65 ± 34.93	**10**	**4.31 ± 11.93**
psr-small(50)	**50**	0.03	**50**	0.02	**50**	**0.01 ± 0.04**	**50**	**0.01 ± 0.05**
rovers(40)	**40**	10.26	**40**	0.33	**40**	**0.13 ± 0.23**	31	12.55 ± 28.76
satellite(36)	31	29.87	30	28.81	**34**	**1.03 ± 1.79**	11	16.92 ± 40.99
scanalyzer-08(30)	7	57.26	7	48.99	9	41.39 ± 56.98	**16**	**21.91 ± 45.07**
scanalyzer-opt11(20)	4	34.20	4	15.28	6	52.22 ± 66.89	**9**	**23.20 ± 54.45**
sokoban-opt08(30)	28	27.27	29	0.58	29	0.25 ± 0.33	**30**	**0.07 ± 0.12**
sokoban-opt11(20)	19	22.22	**20**	0.46	**20**	0.23 ± 0.28	**20**	**0.07 ± 0.13**
tpp(30)	11	12.42	24	49.31	**30**	**4.60 ± 9.66**	24	46.47 ± 81.70
transport-opt08(30)	3	8.64	4	43.16	13	11.60 ± 24.25	**24**	**56.58 ± 87.08**
transport-opt11(20)	0	0	0	0	13	28.55 ± 42.33	**20**	**17.45 ± 32.84**
trucks(30)	**30**	1.60	**30**	0.67	**30**	**0.03 ± 0.02**	**30**	1.74 ± 3.16
visitall-opt11(20)	**20**	3.80	**20**	1.83	**20**	1.11 ± 3.08	**20**	**0.21 ± 0.38**
woodworking-opt08(30)	**30**	1.98	**30**	0.49	**30**	**0.02 ± 0.01**	**30**	0.14 ± 0.26
woodworking-opt11(20)	**20**	2.17	**20**	0.46	**20**	**0.02 ± 0.01**	**20**	0.08 ± 0.07
zenotravel(20)	14	4.02	14	1.04	**20**	**3.38 ± 9.70**	12	20.86 ± 65.51
Total coverage (1376)	790		994		1190		1134	
# Best Domains	13		17		33		31	

Table 6: Detailed comparison of $\text{IP}(T^+)$, $\text{IP}(T^+)$+LM, $\text{IP}^{\text{e}}(T^+)$, and HST/CPLEX on 1376 delete-free tasks (exact computation of h^+). 5-minute time limit, 2GB RAM. Coverage and mean ± standard deviation of runtimes (average of successful runs only, excludes unsuccessful runs).

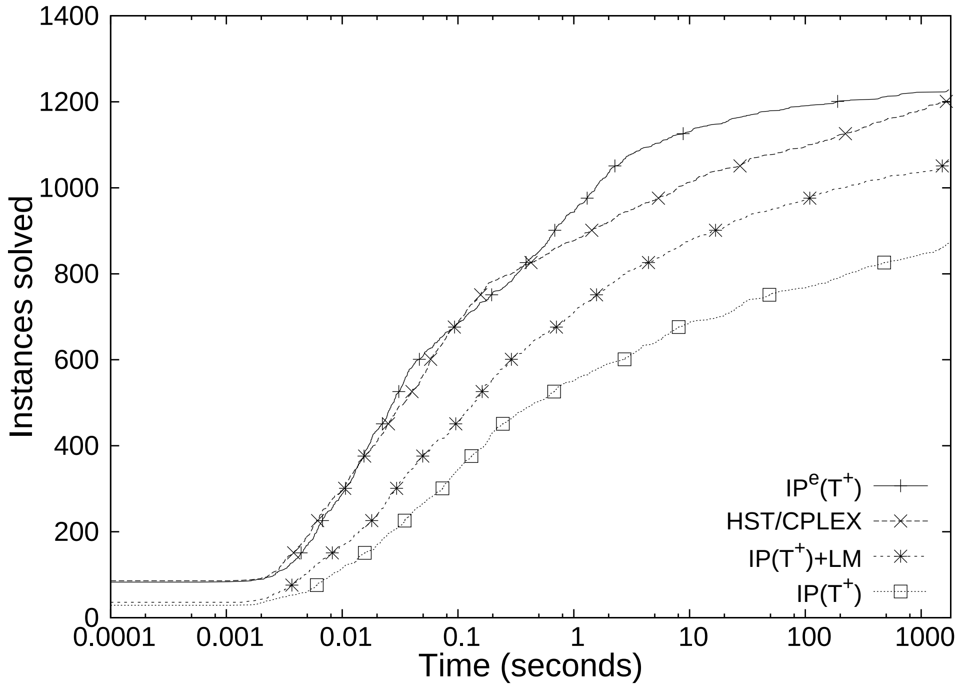


Figure 2: Comparison of IP(T^+), IP(T^+)+LM, IP$^{\mathrm{e}}(T^+)$, and HST/CPLEX on delete-free tasks (exact computation of h^+). 30-minute time limit, 2GB RAM. The cumulative number of instances (out of the same 1376 instances as Table 5) solved within $Time$ seconds is shown.

computing h^+ is NP-equivalent (Bylander, 1994). Haslum (2012) reported some previous, baseline results using a direct computation of h^+ using the hitting-set method proposed in his earlier work (Haslum et al., 2012) as a heuristic for A* , and reported poor results. Although we show in Section 7 that A* using IP$^{\mathrm{e}}(T^+)$ performs well on some domains, using h^+ directly as a heuristic for A* continues to pose a significant challenge. Thus, we turn next to *relaxations* of IP(T^+) and IP$^{\mathrm{e}}(T^+)$ that are lower bounds on h^+ and can be computed faster, making them more suitable as admissible heuristics for a forward-search planner than our IP models.

6.1 LP(T^+) and LP$^{\mathrm{e}}(T^+)$: LP Relaxations of the Delete-Relaxation (h^+) Models

The linear programming (LP) relaxations of the IP models are obvious candidates for tractable alternatives to computing h^+ using IP(T^+) and IP$^{\mathrm{e}}(T^+)$. The LP-relaxations are trivially derived from the IP models by eliminating the integer constraints on the variables, and the optimal cost of the LP-relaxation is a lower bound on the optimal cost of the IP. We denote the LP relaxation of IP(T^+) as LP(T^+) and the LP relaxation of IP$^{\mathrm{e}}(T^+)$ as LP$^{\mathrm{e}}(T^+)$ (see Table 1). In the case of problem domains with integer action costs, the ceiling of the LP costs are used.

Although LP$^{\mathrm{e}}(T^+)$ can be solved quickly, tight theoretical bounds on the gap between IP(T^+) and LP(T^+) or the gap between IP$^{\mathrm{e}}(T^+)$ and LP$^{\mathrm{e}}(T^+)$ are difficult to obtain – it has been proven by Betz and Helmert (2009) that there exists no constant $c > 0$ and no polynomial-time algorithm for computing a lower bound h such that for all states s, $h(s) \geq ch^+$, unless $P = NP$ (i.e., h^+ is not polynomial-time approximable for any constant factor c). Fortunately, the worst-case

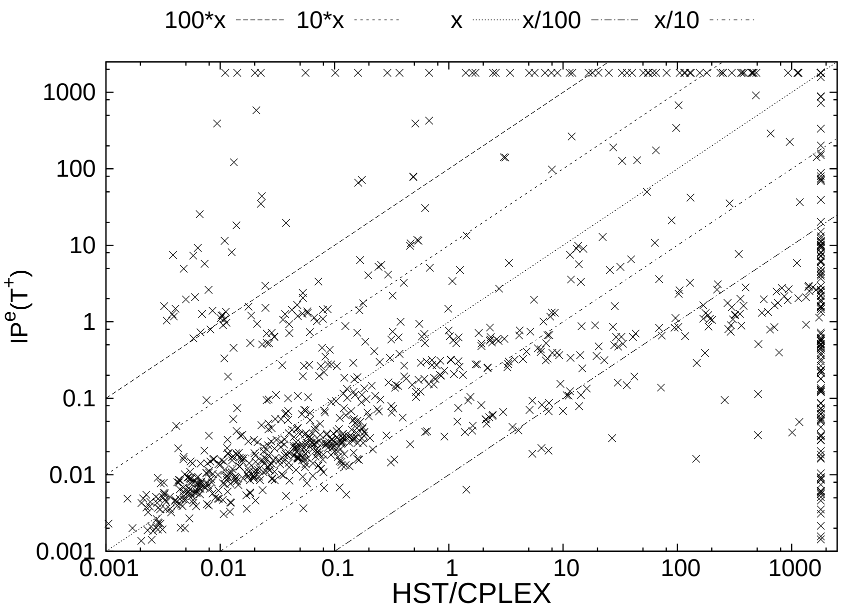


Figure 3: Comparison of runtimes of $\mathrm{IP}^{\mathrm{e}}(T^{+})$ and HST/CPLEX on 1376 delete-free instances (exact computation of h^{+}, same instances as Table 5). 30-minute time limit, 2GB RAM. Each point represents a problem instance. The x-axis represents the runtime of HST/CPLEX, while the y-axis represents the runtime of $\mathrm{IP}^{\mathrm{e}}(T^{+})$. For example, a point "below" the diagonal ($y = x$) indicates that $\mathrm{IP}^{\mathrm{e}}(T^{+})$ solved the problem represented by that point faster than HST/CPLEX, and a point below the $y = x/10$ line indicates that $\mathrm{IP}^{\mathrm{e}}(T^{+})$ solved the problem represented by that point at least 10 times faster than HST/CPLEX. If an algorithm failed to solve an instance within the 30-minute time limit, the runtime is shown as 1800 seconds.

theoretical approximation results do not necessarily apply to real-world problem instances. In fact, our experimental results below show that the LP-relaxations often provide fast, accurate, lower bounds on h^{+} for standard planning benchmark problems.

6.2 Time-Relaxation of h^{+} Models

If our motivation is to embed a computation for h^{+} (or an approximation thereof) as an admissible heuristic for A*, we are not necessarily interested in the actual optimal delete-free plan for T^{+}, but only the cost of that plan (or its approximation). In particular, if the exact order in which actions are executed in the delete-relaxed plan does not matter, the necessity of all time-related variables can be brought into question.

The *time-relaxation* of $\mathrm{IP}(T^{+})$, which is $\mathrm{IP}(T^{+})$ without constraints C5 and C6, is denoted $\mathrm{IP}_{\mathrm{tr}}(T^{+})$. The LP relaxation of $\mathrm{IP}_{\mathrm{tr}}(T^{+})$ is denoted $\mathrm{LP}_{\mathrm{tr}}(T^{+})$. Table 1 summarizes the relationships among these models.

If the propositions and actions of the task satisfy some conditions, eliminating the time-related variables does not affect the cost of the optimal solution to $\mathrm{IP}(T^{+})$. For example, if the relaxed causal AND/OR graph (Gefen & Brafman, 2012) of the task does not have a cycle, then we can decide the values of $\mathcal{T}(p)$ and $\mathcal{T}(a)$ such that constraints C5 and C6 of $\mathrm{IP}(T^{+})$ are satisfied in-

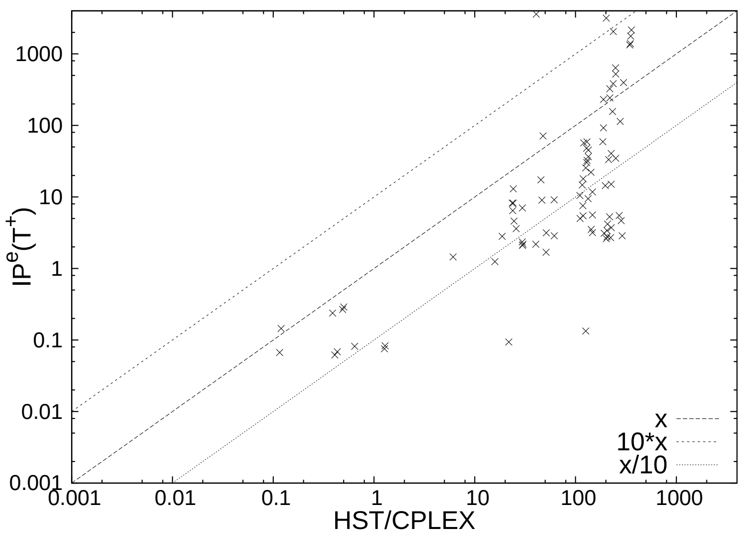


Figure 4: Runtime comparisons of $\mathrm{IP}^{\mathrm{e}}(T^+)$ and HST/CPLEX on minimal seed set problem (88 natural, delete-free instances from Gefen & Brafman, 2011). 60-minute time limit, 2GB RAM. Each point represents a problem instance. If an algorithm failed to solve an instance within the 60-minute time limit, the runtime is shown as 3600 seconds. The coverage of $\mathrm{IP}^{\mathrm{e}}(T^+)$ was 87 instances, while the coverage of HST/CPLEX was 88 instances.

dependently of the values of the other variables, in which case 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.

Indeed, we shall show experimentally in Section 6.3 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 $\mathrm{IP}(T^+)$. Similarly, $\mathrm{LP}_{\mathrm{tr}}(T^+), \mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$, and $\mathrm{IP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$, the time-relaxations of $\mathrm{LP}(T^+), \mathrm{LP}^{\mathrm{e}}(T^+)$, and $\mathrm{IP}^{\mathrm{e}}(T^+)$, can be computed much faster than their non-time-relaxed counterparts.

6.3 Experimental Evaluation of LP and Time Relaxation Gaps

We evaluated the quality of the $\mathrm{LP}(T^+), \mathrm{LP}^{\mathrm{e}}(T^+)$, and $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$ linear programming bounds described above by comparing optimal costs computed for these bounds to exact h^+ values (computed using $\mathrm{IP}^{\mathrm{e}}(T^+)$). We used the same set of 1376 instances as in Table 5. Table 7 shows the mean ratio of the optimal cost of each LP model to h^+, on all instances where h^+ could be computed using $\mathrm{IP}^{\mathrm{e}}(T^+)$. The “perfect” columns indicate the fraction of instances where the optimal cost of the LP model was equal to h^+. Note that we used the ceiling of the LP cost, since the IPC benchmark instances have integer costs. A stacked histogram representation of the same data (aggregated over all domains) which classifies the ratios of the optimal costs of the LP relaxations to the value of h^+ is shown in Figure 5.

We should expect that the variable-fixing constraints in our enhanced $\mathrm{LP}^{\mathrm{e}}(T^+)$ model would tend to increase the value of the optimal solution to $\mathrm{LP}^{\mathrm{e}}(T^+)$ compared to the optimal value of the base LP relaxation, $\mathrm{LP}(T^+)$. In addition, we would also expect that the optimal value for $\mathrm{LP}^{\mathrm{e}}(T^+)$ would tend to be greater than the optimal value of its time relaxation, $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$. Table 7 shows that in general, $\text{LP}^{\text{e}}(T^+) \geq \text{LP}^{\text{e}}_{\text{tr}}(T^+) \geq \text{LP}(T^+)$. For 10/43 domains, $\text{LP}^{\text{e}}(T^+)$ matches h^+ perfectly, i.e., $\text{LP}^{\text{e}}(T^+)/h^+ = 1$. For 20/43 domains, $\text{LP}^{\text{e}}(T^+)/h^+ \geq 0.95$. On almost every single domain, the optimal LP value of the enhanced model $\text{LP}^{\text{e}}(T^+)$ is significantly better (higher) than the basic formulation $\text{LP}(T^+)$, confirming that variable elimination and the additional constraints serve to tighten the LP bound. Thus, the enhancements to the basic model described in Section 4 provide a significant benefit beyond the speedups that were demonstrated in Section 5. The time-relaxation $\text{LP}^{\text{e}}_{\text{tr}}(T^+)$ is usually very close to $\text{LP}^{\text{e}}(T^+)$, indicating that the time relaxation can potentially achieve a good tradeoff between computation cost and accuracy (and in fact, as we see later in Section 7, $\text{LP}^{\text{e}}_{\text{tr}}(T^+)$ performs quite well when used as a heuristic for A^*).

For comparison, we also evaluated the ratio of the value of the LM-cut heuristic (Helmert & Domshlak, 2009) to h^+. Comparing the average ratios of each lower bound to h^+, we see that:

- $\text{LP}(T^+)$ is less informative than LM-cut on 31 domains, more informative than LM-cut on 5 domains, and equivalent on 6 domains.
- $\text{LP}^{\text{e}}(T^+)$ is less informative than LM-cut on 16 domains, more informative than LM-cut on 19 domains, and equivalent on 8 domains.
- $\text{LP}^{\text{e}}_{\text{tr}}(T^+)$ is less informative than LM-cut on 17 domains, more informative than LM-cut on 17 domains, and equivalent on 9 domains.

Thus, while LM-cut is a better approximation to h^+ than the basic LP-relaxation, $\text{LP}(T^+)$, $\text{LP}^{\text{e}}(T^+)$ and $\text{LP}^{\text{e}}_{\text{tr}}(T^+)$ are roughly equivalent to LM-cut. Interestingly, the LP-relaxation approach appears to be highly complementary to the cost-partitioning approach of LM-cut, in that the LP-relaxation and LM-cut are each more informative than the other on roughly half of the cases compared to each other.

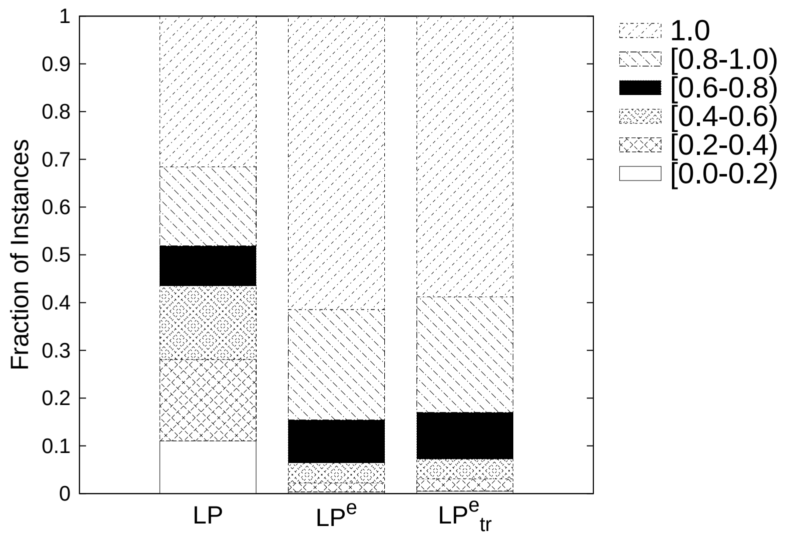
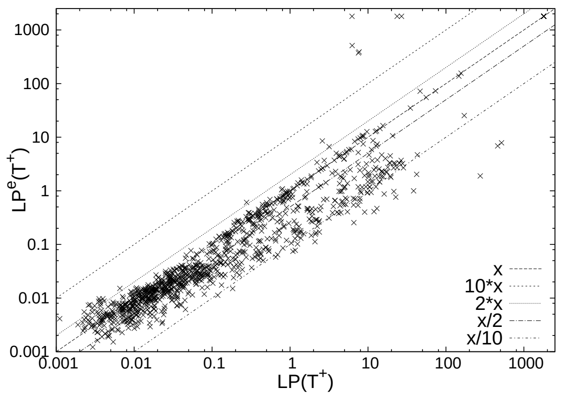


Figure 5: Ratio between optimal LP costs and h^+, categorized into buckets. [x:y) = "fraction of instances where the ratio LP/h^+ is in the range [x:y)". For example, the fraction of instances where the ratio between the optimal value of $\text{LP}^{\text{e}}_{\text{tr}}(T^+)$ and h^+ was in the range [0.8,1,0) is approximately 0.24 (this stacked histogram is based on the same data as Table 6.3).

	LM-cut		$\mathrm{LP}(T^+)$		$\mathrm{LP}^{\mathrm{e}}(T^+)$		$\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$	
	LM-cut/h^+	perfect	$\mathrm{LP}(T^+)/h^+$	perfect	$\mathrm{LP}^{\mathrm{e}}(T^+)/h^+$	perfect	$\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)/h^+$	perfect
airport	**1.00**	.74	.46	.02	.98	.94	.98	.70
barman-opt11	**.74**	0	.17	0	.38	0	.38	0
blocks	.99	.97	.92	.20	**1.00**	1.00	**1.00**	1.00
depot	.64	0	.50	0	**.92**	.22	.91	.18
driverlog	**.89**	.20	.85	.10	.87	.21	.83	.05
elevators-opt08	**.77**	.06	.21	0	.65	0	.64	0
elevators-opt11	**.80**	.10	.20	0	.64	0	.62	0
floortile-opt11	.94	.05	**.95**	.10	**.95**	.10	**.95**	.10
freecell	.29	0	.12	0	**.94**	.35	.92	.23
grid	.67	.40	.31	.20	**.81**	.20	.79	.20
gripper	**1.00**	1.00	**1.00**	1.00	**1.00**	1.00	**1.00**	1.00
logistics98	**.98**	.40	.39	.02	.89	.11	.88	.05
logistics00	**.99**	.92	.46	.03	**.99**	.85	**.99**	.78
miconic	**1.00**	1.00	**1.00**	1.00	**1.00**	1.00	**1.00**	1.00
no-mprime	**.76**	.20	.42	0	.71	.33	.63	.17
no-mystery	**.79**	.28	.39	0	.77	.33	.72	.30
nomystery-opt11	.93	.50	.96	.60	**1.00**	.95	**1.00**	.95
openstacks	.61	0	.23	.03	**1.00**	.96	.88	1.00
openstacks-opt08	**1.00**	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	**1.00**	1.00
parcprinter-08	**.99**	.70	**.99**	.66	**.99**	.66	**.99**	.66
parcprinter-opt11	**.99**	.65	**.99**	.70	**.99**	.70	**.99**	.70
parking-opt11	.87	0	.88	0	**.92**	.10	.87	0
pathways-noneg	.87	.13	.90	.13	**.98**	.60	**.98**	.60
pegsol-08	.60	.26	.26	.03	**.64**	.03	**.64**	.03
pegsol-opt11	.55	.05	.20	0	**.65**	0	**.65**	0
pipesworld-notankage	.68	.02	.52	0	**.83**	.38	.79	.08
pipesworld-tankage	.75	0	.58	0	**.93**	.55	.91	.36
psr-small	**1.00**	1.00	.87	.82	**1.00**	1.00	**1.00**	1.00
rovers	**.87**	.12	.48	0	.65	.35	.65	.30
satellite	**.95**	.23	.82	.13	.82	.21	.82	.20
scanalyzer-08	**.95**	.32	.94	.30	.94	.75	.94	.34
scanalyzer-opt11	**.97**	.26	.96	.25	.96	.71	.96	.29
sokoban-opt08	.94	.53	.33	.13	.95	.73	.94	.66
sokoban-opt11	.96	.60	.28	.15	**.97**	.80	**.97**	.75
tpp	**.98**	.55	.28	.13	.85	.26	.85	.26
transport-opt08	**.87**	.03	.08	0	.35	.08	.35	.03
transport-opt11	**.84**	.05	.09	0	.41	0	.41	0
trucks	.92	0	.40	0	**1.00**	1.00	**1.00**	1.00
visitall-opt11	.69	.10	**.98**	.65	**.98**	.65	.97	.65
woodworking-opt08	.89	.13	.81	0	**1.00**	1.00	**1.00**	1.00
woodworking-opt11	.88	.10	.80	0	**1.00**	1.00	**1.00**	1.00
zenotravel	**.95**	.50	.91	.25	.92	.31	.89	.30

Table 7: Gaps between LP models and h^+: The mean ratio of each LP model to h^+ (on the 1228 instances solved using $\mathrm{IP}^{\mathrm{e}}(T^+)$ is shown. The "perfect" columns indicate the fraction of instances where the optimal cost of the LP model was equal to h^+.

Figure 6 compares the runtimes the CPLEX LP solver on the relaxed h^+ models. $\mathrm{LP}^{\mathrm{e}}(T^+)$ is significantly faster than $\mathrm{LP}(T^+)$, solving many instances 2-10 times faster (and solving some instances more than 10 times faster), demonstrating the benefits of our enhanced model. The comparison of $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$ and $\mathrm{LP}^{\mathrm{e}}(T^+)$ shows that using the time relaxation results in an addition speedup of up to a factor of 2. While this additional speedup may not seem very significant when solving a single LP instance that takes a fraction of a second, the cumulative effects when using the LP models as a heuristic for forward-search based planning is significant, and as we show in Section 7, this results in increased coverage when using $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$ as a heuristic for A*, compared to $\mathrm{LP}^{\mathrm{e}}(T^+)$.



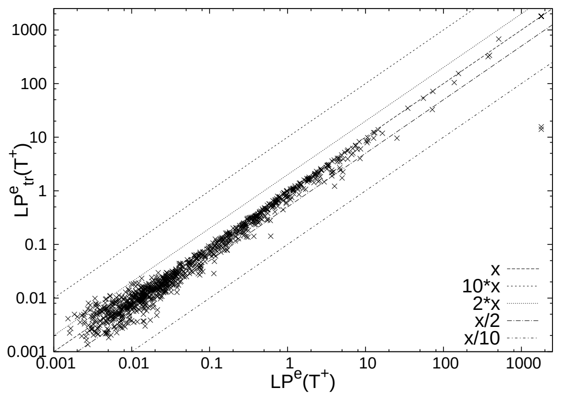


Figure 6: Runtime comparisons for relaxed h^+ models. on 1376 delete-free instances (exact computation of h^+, same instances as Table 5). 30-minute time limit, 2GB RAM. Each point represents a problem instance. The left subfigure compare $\mathrm{LP}(T^+)$ vs $\mathrm{LP}^{\mathrm{e}}(T^+)$, showing the impact of our enhancements to the basic LP model, and the right subfigure compares $\mathrm{LP}^{\mathrm{e}}(T^+)$ vs $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$, showing the impact of the time relaxation. If an algorithm failed to solve an instance within the 30-minute time limit, the runtime is shown as 1800 seconds.

7. Cost-Optimal Planners Using Our h^+-Based Heuristics

We embedded the IP and LP models that have been introduced so far into an A*-based, cost-optimal forward search planner (our own planner implementation, which uses a propositional representation internally) and evaluated their performance. Note that this particular experiment is limited to admissible heuristics whose value is bounded above by h^+. The later results in Section 8 and 9 include heuristics that are not necessarily bounded above by h^+. Specifically, we evaluated the following solver configurations:

- A*/$\mathrm{IP}(T^+)$: A* with the basic delete-free IP model $\mathrm{IP}(T^+)$ as a heuristic.
- A*/$\mathrm{IP}^{\mathrm{e}}(T^+)$: A* with the enhanced delete-free IP model $\mathrm{IP}^{\mathrm{e}}(T^+)$ as a heuristic.
- A*/$\mathrm{LP}^{\mathrm{e}}(T^+)$: A* with the LP relaxation of the enhanced delete-free IP model $\mathrm{IP}^{\mathrm{e}}(T^+)$ as a heuristic.
- A*/$\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$: A* with the LP relaxation of the time-relaxed, enhanced delete-free IP model $\mathrm{IP}^{\mathrm{e}}(T^+)$ as a heuristic.

- hsp/HST/CPLEX : A* where the heuristic is the hitting-set based h^+ solver HST/CPLEX (Haslum et al., 2012) using CPLEX to solve hitting set instances (hsp planner provided by Patrik Haslum).
- FD/h^{max} : Fast Downward using the hmax heuristic (Bonet & Geffner, 2001).
- FD/LM-cut : Fast Downward using the landmark cut heuristic (Helmert & Domshlak, 2009) (the standard `seq-opt-lmcut` configuration)

As per standard IPC sequential optimal track settings, all solver configurations were run with a 30 minute time limit per problem and a 2GB RAM limit. A set of 1376 instances from IPC1998-IPC-2011 were used. Our planner currently handles the STRIPS subset of PDDL with action costs.

Table 8 compares the coverage of these heuristics. Figure 7a shows the cumulative coverage (out of 1376) solved as a function of time for the solver configurations compared in Table 8, and Figure 7b shows cumulative coverage as a function of the number of node evaluations (calls to the heuristic function by A*).

While we compare our IP/LP-based A*-heuristics with other planners, note that there are significant implementation-level differences other than the heuristic function that can affect execution speed. For example, Fast Downward uses a multi-valued SAS+ representation (Bäckström & Nebel, 1995) internally to represent states, while our planner uses a STRIPS propositional representation, so there are significant differences in internal data structures and implementation details. Thus, these results should only be used for qualitative comparisons.

Table 8 shows that A*/IP(T^+), which uses the basic IP(T^+) model, had the worst coverage among our IP models (403), comparable to that of A*/HST/CPLEX(398). As noted by Haslum (2012), straightforward use of h^+ as a heuristic can be unsuccessful (even worse than FD using $h^{\max}$, which has a coverage of 540) if the cost of computing h^+ at each search node is too high.

However, as shown in Section 5, solving the IP$^{\mathrm{e}}(T^+)$ IP model is significantly faster than IP(T^+) and A*/HST/CPLEX. This makes it much more viable as a heuristic function for A*, and as a result, A*/IP$^{\mathrm{e}}(T^+)$ has a coverage of 635, significantly outperforming both A*/HST/CPLEX as well as FD/hmax.

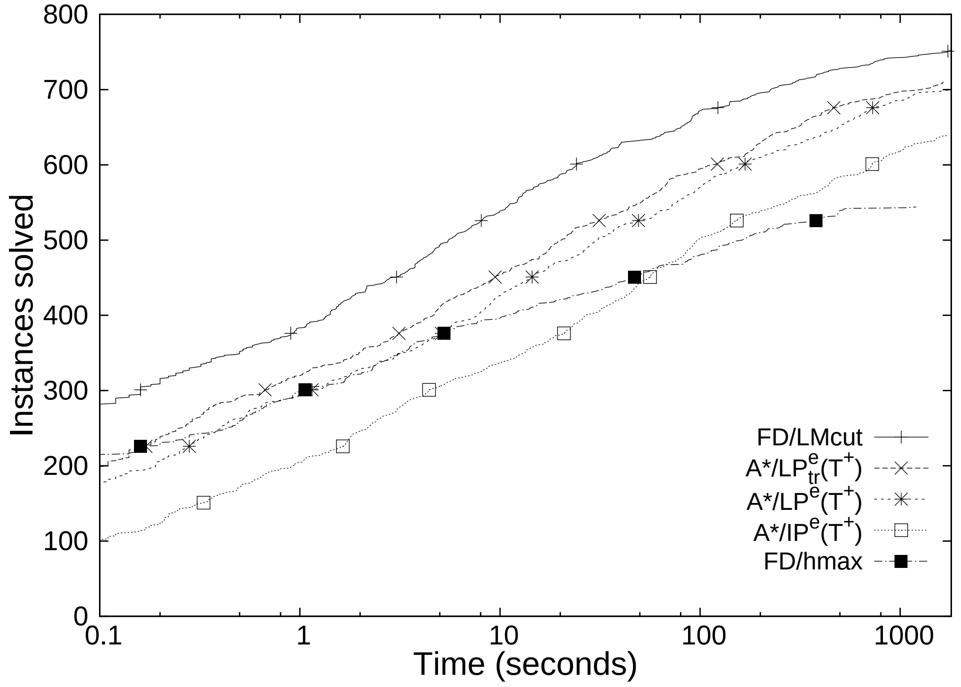
As shown in Section 6.3, the LP relaxations of our IP models provide relatively tight lower bounds for h^+. Since the LP models can be solved much faster than IP, they are quite effective when used as heuristics for A*. Thus, A*/LP$^{\mathrm{e}}(T^+)$, which uses the LP-relaxation of the enhanced IP$^{\mathrm{e}}(T^+)$ model, has a coverage of 696, and A*/LP$^{\mathrm{e}}_{\mathrm{tr}}(T^+)$, which uses the LP-relaxation of the time-relaxed, enhanced IP model, has a coverage of 705.

In Section 6.3, we showed that the LP$^{\mathrm{e}}(T^+)$ and LP$^{\mathrm{e}}_{\mathrm{tr}}(T^+)$ models are complementary to LM-cut with respect to informativeness, which suggests that at least with respect to search efficiency, our LP models should be competitive with LM-cut. Figure 7b shows that in fact, A*/LP$^{\mathrm{e}}(T^+)$ and A*/LP$^{\mathrm{e}}_{\mathrm{tr}}(T^+)$ tend to search quite efficiently, and it can be seen that both of these lines are above the LM-cut line (i.e., more problems were solved using a given number of evaluations) until between $10^5 \sim 10^6$ node evaluations, at which point they are overtaken by the LM-cut line. While the informativeness comparison in Section 6.3 showed that the LP models are comparable and complementary to LM-cut with respect to informativeness, FD/LM-cut outperforms A*/LP$^{\mathrm{e}}_{\mathrm{tr}}(T^+)$ and A*/LP$^{\mathrm{e}}_{\mathrm{tr}}(T^+)$ on most domains. This is because the LM-cut implementation in Fast Downward is often significantly faster than the current implementation of our LP-based heuristics. Nevertheless, there are several domains (freecell, parcprinter-08, parcprinter-opt11, satellite, trucks, visitall),

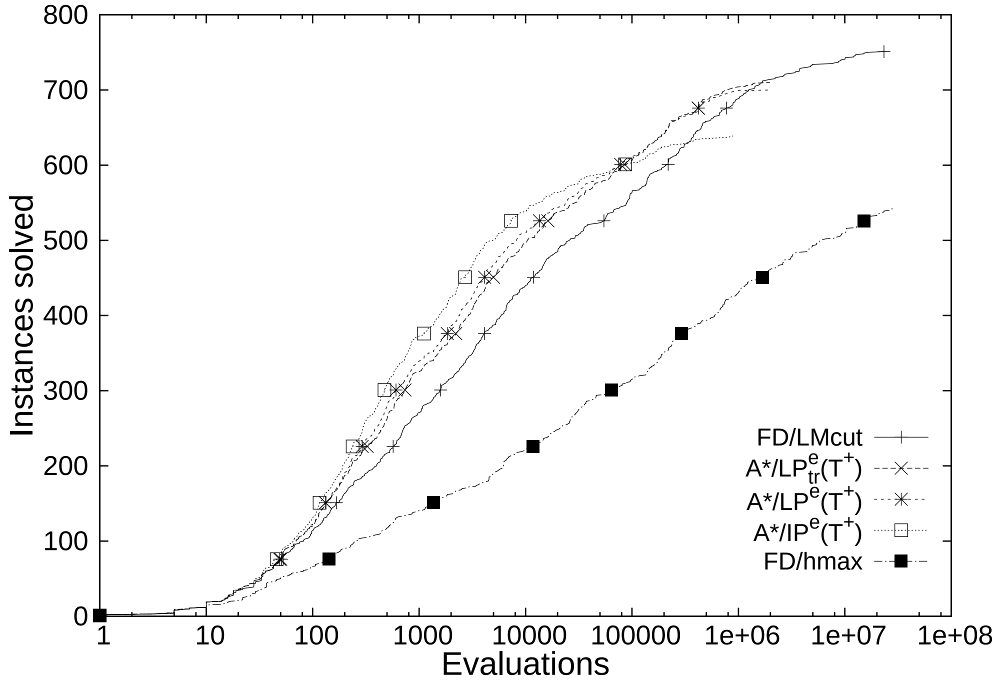
	FD/hmax	FD/LM-cut	hsp/HST/CPLEX	$\text{A}^*/\text{IP}(T^+)$	$\text{A}^*/\text{IP}^{\text{e}}(T^+)$	$\text{A}^*/\text{LP}^{\text{e}}(T^+)$	$\text{A}^*/\text{LP}^{\text{e}}_{\text{tr}}(T^+)$
Domain (# problems)	solved	solved	solved	solved	solved	solved	solved
airport(50)	21	**28**	24	14	24	25	25
barman-opt11(20)	**4**	**4**	0	0	0	0	0
blocks(35)	18	**28**	17	19	27	**28**	**28**
depot(22)	4	**7**	1	2	**7**	**7**	**7**
driverlog(20)	9	**14**	7	9	10	11	13
elevators-opt08(30)	15	**22**	3	0	9	13	13
elevators-opt11(20)	13	**18**	1	0	7	10	10
floortile-opt11(20)	4	**7**	1	2	4	6	**7**
freecell(80)	15	15	19	8	**54**	44	43
grid(5)	**2**	**2**	1	0	**2**	**2**	**2**
gripper(20)	**7**	**7**	2	4	5	6	6
logistics98(35)	2	**6**	3	3	5	**6**	**6**
logistics00(28)	10	**20**	10	16	19	**20**	**20**
miconic(150)	50	**141**	79	137	140	140	**141**
no-mprime(35)	**23**	**23**	15	10	20	18	17
no-mystery(30)	**17**	16	15	5	15	13	12
nomystery-opt11(20)	8	**14**	8	8	**14**	**14**	**14**
openstacks(30)	**7**	**7**	5	0	**7**	**7**	**7**
openstacks-opt08(30)	**19**	**19**	7	2	10	11	11
openstacks-opt11(20)	**14**	**14**	2	0	5	6	6
parcprinter-08(30)	14	19	19	19	**21**	20	20
parcprinter-opt11(20)	10	14	14	14	**16**	**16**	**16**
parking-opt11(20)	0	**3**	0	0	2	1	1
pathways-noneg(30)	4	**5**	4	**5**	**5**	**5**	**5**
pegsol-08(30)	**27**	**27**	17	1	10	26	26
pegsol-opt11(20)	**17**	**17**	4	0	2	16	16
pipesworld-notankage(50)	16	**17**	9	3	10	12	13
pipesworld-tankage(50)	7	**8**	6	2	**8**	7	7
psr-small(50)	**49**	**49**	19	43	48	48	48
rovers(40)	6	**7**	4	**7**	**7**	**7**	**7**
satellite(36)	6	7	5	8	**10**	**10**	**10**
scanalyzer-08(30)	9	**15**	5	5	5	8	8
scanalyzer-opt11(20)	6	**12**	2	2	2	5	5
sokoban-opt08(30)	27	**30**	6	3	17	23	25
sokoban-opt11(20)	**20**	**20**	3	1	13	19	19
tpp(30)	**6**	**6**	5	5	**6**	**6**	**6**
transport-opt08(30)	**11**	**11**	7	2	7	9	10
transport-opt11(20)	**6**	**6**	2	0	2	4	5
trucks(30)	7	10	3	7	13	**15**	**15**
visitall-opt11(20)	9	11	15	9	10	**16**	**16**
woodworking-opt08(30)	9	**17**	14	12	**17**	16	**17**
woodworking-opt11(20)	4	**12**	8	7	11	10	11
zenotravel(20)	8	**13**	7	9	9	10	11
Total coverage (1376)	540	748	398	403	635	696	705
# Best domains	15	36	0	0	13	14	17

Table 8: Comparison of forward search (A*) planners, part 1: Number of problems solved with 30 minute, 2GB RAM limit using A* and our IP/LP models which are bounded above by h^+ (Sections 3-7) as heuristic functions. Comparison with Fast Downward with h^{max}, Fast Downward with Landmark Cut, and the hsp planner using HST/CPLEX (Haslum et al., 2012) to compute h^+, as the heuristic function.

where $\text{A}^*/\text{LP}^{\text{e}}_{\text{tr}}(T^+)$ achieves higher coverage than FD/LM-cut. Thus, $\text{A}^*/\text{LP}^{\text{e}}_{\text{tr}}(T^+)$, our best model among those which are bounded above by h^+, can be considered a fairly powerful, admissible heuristic function for forward-state search based planning.



(a) Cumulative number of problems solved (out of 1376) vs time (30 minute time limit).



(b) Cumulative number of problems solved (out of 1376) vs number of search nodes evaluated (30 minute time limit).

Figure 7: Comparison of forward search (A*) planners, part 1 (heuristics that are bounded above by h^+).

8. Incorporating Counting Constraints

So far, we have concentrated on efficient computation of h^+ as well as relaxations of h^+, and all of our models so far have been bounded above by h^+. However, our IP model can be extended with constraints that consider 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 an auxiliary variable similar to $\mathcal{I}(p)$. Furthermore, in this extended model, the meaning of $\mathcal{U}(a) \in \{0, 1\}$ is slightly modified to mean that action a is used *at least once* in the optimal solution (in the basic model proposed in Section 3, which was a pure delete-free model, $\mathcal{U}(a)$ denoted whether a was used exactly once or not at all in the optimal solution).

New constraints are defined as follows:

(C7) $\forall a \in A, \mathcal{N}(a) \geq \mathcal{U}(a)$.

(C8) $\forall p \in P, \mathcal{G}(p) + \sum_{as.t.p \in \mathrm{predel}(a)} \mathcal{N}(a) \leq \mathcal{I}(p) + \sum_{as.t.p \in \mathrm{add}(a)} \mathcal{N}(a)$,

where $\mathrm{predel}(a) = \mathrm{pre}(a) \cap \mathrm{del}(a)$. Finally, the objective function is modified so as to minimize $\sum_{a \in A} c(a)\mathcal{N}(a)$. Given a planning task T, we use $\mathrm{IP_c}(T)$ to denote an IP problem which adds the and above new variables and constraints to $\mathrm{IP}(T^+)$

The idea for these types of constraints have been previously proposed several times (for a SAS$^+$ formulation), and correspond to the action order relaxation by van den Briel et al. (2007), the *state equation heuristic* by Bonet (2013), and the *net change* constraints by Pommerening et al. (2014).

Intuitively, the final constraint states that the number of uses of actions adding p must be greater than or equal to the number of uses of actions requiring and deleting p at the same time in a feasible plan for T. Any feasible plan for a STRIPS planning task always satisfies this condition. Hence, for any task T and any feasible plan π for T, we can clearly derive a feasible solution to $\mathrm{IP_c}(T)$ with the same cost as π. In addition to this, a stronger proposition can be proved for modifications of models by the enhancements in Section 4.

Proposition 7. *Given a task T, for any feasible plan $\pi = (a_0, \cdots, a_n)$ of T, there exists a feasible solution to $\mathrm{IP_c}(T)$ that has the same cost as the cost of π. In addition to this, there exists a feasible solution to $\mathrm{IP_c}(T)$ with any combination of landmark extraction and substitution, relevance analysis, and inverse action constraints that has the same cost as the cost of π.*

Proof. Let π^+ be the delete relaxation of the subsequence of the plan π extracted by Algorithm 3. First we show that the subsequence π^+ is a feasible delete-free plan for T^+, and then we show that the assignment derived from π^+ satisfies the constraints.

We use $(a_0^+, \cdots, a_m^+)$ to denote the elements of π^+. To show that π^+ is feasible in T^+, assume a_i^+ is the first infeasible action in π^+. Let p be a proposition such that $p \in \mathrm{pre}(a_i^+)$ and $p \notin I((a_0^+, \cdots, a_{i-1}^+))$. Since π is a valid feasible plan for T, the delete-relaxation of the entire sequence of π is a valid feasible plan for T^+. Hence, if a_i^+ is not feasible, then this is because Algorithm 3 skipped all the actions that add p before a_i^+ is applied. Since S on line 5 in Algorithm 3 is equal to $I((a_0^+, \cdots, a_{i-1}^+))$ for each i, all the skipped actions that add p satisfy $\mathrm{add}(a_i) \setminus S \neq \emptyset$, and thus

Algorithm 3 Extracting a subsequence of $\pi = (a_0, \cdots, a_n)$ (for the proof of Proposition 7)

```
1: $\pi^+ \leftarrow ()$; // empty
2: $S \leftarrow I$;
3: **for** $a = a_0, \cdots, a_n$ **do**
4:     Let $a'$ be the delete-relaxation of $a$.
5:     **if** $a'$ is relevant to $T^+$ and $\mathrm{add}(a') \setminus S \neq \emptyset$ **then**
6:         append $a'$ at the end of $\pi^+$;
7:         $S \leftarrow S \cup \mathrm{add}(a')$;
8:     **end if**
9: **end for**
10: return $\pi^+$;
```

they are irrelevant to T^+. However this contradicts the definition of the relevance analysis and the fact that a_i^+ is relevant. Similar to this argument, we have $G \subseteq I(\pi^+)$. Hence π^+ is a valid feasible plan for T^+.

Define an assignment F for $\mathrm{IP_c}(T)$ as:

- $\mathcal{V}_F := \mathcal{V}_{F^+}$ for each variable $\mathcal{V}$ that is defined for $\mathrm{IP}(T^+)$, where F^+ is the assignment derived from π^+ for $\mathrm{IP}(T^+)$, and

- $\mathcal{N}(a)_F :=$ (the number of occurrences of a in π) for each $a \in A$.

The assignment F clearly satisfies constraints C1 through C6. The assignment F also satisfies constraint C8 since π is a valid plan for T, and F satisfies constraint C7 since $\mathcal{U}(a)_F = 0$ if a is not included in π. Hence F is a feasible solution to $\mathrm{IP_c}(T)$ which has the same cost as π.

In addition, F is also a feasible solution to $\mathrm{IP_c}(T)$ with any combination of landmark extraction and substitution, relevance analysis, and inverse action constraints. We can see this by checking the feasibility of F with each type of modified constraints *independently*. If F satisfies each of the modified constraints, then it satisfies any combination of such constraints.

F satisfies the constraints added by the landmark extraction and substitution (i.e. substituting 1 for variables corresponding to landmarks) since π^+ is a valid feasible plan for T^+. F also satisfies constraints added by the relevance analysis (i.e. substituting 0 for irrelevant actions and propositions) since π^+ contains only relevant actions. Finally, we can show that F satisfies inverse action constraints similarly to the proof of Proposition 6. We have $\sum_{a' \in \mathrm{inv}(a,p)} \mathcal{E}(a', p)_F = 0$ when $\mathcal{U}(a)_F = 0$ and $\mathcal{U}(p)_F = 0$ hold, and we also have $\sum_{a' \in \mathrm{inv}(a,p)} \mathcal{E}(a', p)_F \leq 1$ when $\mathcal{U}(a)_F = 0$ and $\mathcal{U}(p)_F = 1$ hold. In addition, we can show that $\sum_{a' \in \mathrm{inv}(a,p)} \mathcal{E}(a', p)_F = 0$ for each $\mathcal{U}(a)_F = \mathcal{U}(p)_F = 1$. Assume that there exists $a' \in \mathrm{inv}(a, p)$ such that $\mathcal{E}(a', p)_F = 1$. Then, by constraint C3, $\mathcal{U}(a')_F = 1$, and this means a' is also a member of π^+. Without loss of generality, assume a is applied before a' is applied in π^+. Since $\mathrm{add}(a') \subseteq \mathrm{pre}(a)$ by the definition of inverse actions, nothing new is added to the state after applying a'. S on line 5 in Algorithm 3 is equal to $I((a_0^+, \cdots, a'))$, and this contradicts $\mathrm{add}(a_i) \setminus S \neq \emptyset$. □

Unfortunately, the counting constraints conflict with dominated action elimination (Section 4.3) and zero cost immediate action application (Section 4.4). When counting constraints are used, it is necessary to disable zero cost immediate action application and to modify the condition of dominated actions as follows:

Definition 2 (modified dominated action definition)**.** *Given a feasible task T, an action a is dominated by 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) $\mathrm{pre}(a') \cap \mathrm{del}(a') \subseteq \mathrm{pre}(a) \cap \mathrm{del}(a)$.*

We can no longer use the modified dominated actions to make a feasible plan for T, since fact landmarks are sometimes deleted after they are achieved. However the following fact can be proved.

Proposition 8. *Given a task T, let $\pi = (a_0, \cdots, a_n)$ be a feasible solution to T. There exists a feasible solution to $\mathrm{IP}_{\mathrm{c}}(T)$ with any combination of landmark extraction and substitution, relevance analysis, inverse action constraints, and the modified dominated action elimination that has cost equal to or less than the cost of π.*

Proof. Recall that the dominated action elimination constraints substitute 0s for $\mathcal{U}(a)$ for each dominated action a. If π does not contain any modified dominated actions, then the proposition holds due to Proposition 7.

Otherwise, we can derive a feasible solution using the sequence of actions made by replacing modified dominated actions in π with their corresponding dominating actions. Let π' be such a sequence. Note that the sum of the costs of the actions in π' is clearly less than or equal to that of π.

Let π'^{+} be the relaxation of the subsequence of π' extracted by Algorithm 3. Since we can prove that the delete-relaxation of π' is a feasible plan for T^+ by an argument similar to the proof of Proposition 4, we can prove that π'^{+} is also a feasible plan for T^+ by an argument similar to the proof of Proposition 7.

If π'^{+} is a feasible plan, then we can derive a feasible solution for $\mathrm{IP}_{\mathrm{c}}(T)$ with the constraints from π' as in the proof of Proposition 7. The solution satisfies constraints C1 through C6 with any combination of landmark extraction and substitution, relevance analysis, and inverse action constraints. It satisfies constraint C7 because $\mathcal{U}(a) = 0$ if a is not included in π', and it satisfies constraint C8 because replacing dominated actions does not invalidate constraint C8 if π is a feasible plan for T. It also satisfies dominated action elimination constraints (i.e. $\mathcal{U}(a) = 0$ for each dominated action a) since π' does not contain any modified dominated action. $\square$

$\mathrm{IP}_{\mathrm{c}}^{\mathrm{e}'}(T)$ and $\mathrm{LP}_{\mathrm{c}}^{\mathrm{e}'}(T)$ denote the models constructed by applying all of the valid reductions to $\mathrm{IP}_{\mathrm{c}}(T)$ and $\mathrm{LP}_{\mathrm{c}}(T)$ respectively. The LP and time relaxations for $\mathrm{IP}(T^+)$ described in Section 6 can be applied to $\mathrm{IP}_{\mathrm{c}}(T)$ as well, and $\mathrm{LP}_{\mathrm{ctr}}^{\mathrm{e}'}(T)$ is the time-relaxed, LP-relaxation of the enhanced $\mathrm{IP}_{\mathrm{c}}^{\mathrm{e}'}(T)$ model. Table 1 summarizes the relationships among these models.

8.1 Experimental Results for Models Enhanced with Counting Constraints

To see the impact of adding counting constraints, we evaluated the informativeness of $\mathrm{LP}_{\mathrm{c}}^{\mathrm{e}'}(T)$, $\mathrm{LP}_{\mathrm{ctr}}^{\mathrm{e}'}(T)$, $\mathrm{LP}^{\mathrm{e}}(T^+)$, and $\mathrm{LP}_{\mathrm{tr}}^{\mathrm{e}}(T^+)$ by comparing their values with the LM-cut heuristic values (Helmert & Domshlak, 2009). Table 9 shows the values of $\mathrm{LP}_{\mathrm{c}}^{\mathrm{e}'}(T)$, $\mathrm{LP}_{\mathrm{ctr}}^{\mathrm{e}'}(T)$, $\mathrm{LP}^{\mathrm{e}}(T^+)$, and $\mathrm{LP}_{\mathrm{tr}}^{\mathrm{e}}(T^+)$ as a multiple of the LM-cut values (means for each domain are shown). Note that in contrast to Table 7, which was limited to the 1228 instances for which h^+ could be computed exactly, Table 9 includes all 1376 instances (because the LM-cut values could be computed for all 1376 instances).

On the majority of domains, the counting constraints result in a more informative heuristic, compared to the models without the counting constraints, so in most cases, $\mathrm{LP}^{\mathrm{e}}(T^+) \leq \mathrm{LP}_{\mathrm{c}}^{\mathrm{e}'}(T)$ and $\mathrm{LP}_{\mathrm{tr}}^{\mathrm{e}}(T^+) \leq \mathrm{LP}_{\mathrm{ctr}}^{\mathrm{e}'}(T)$. It is sometimes possible for the optimal value of $\mathrm{LP}^{\mathrm{e}}(T^+)$ to be larger than the optimal value of $\mathrm{LP}^{\mathrm{e}'}_{\mathrm{c}}(T)$ and for $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$ to have a larger optimal value than $\mathrm{LP}^{\mathrm{e}'}_{\mathrm{ctr}}(T)$ because as explained in Section 8, some of the additional constraints that are part of $\mathrm{IP}^{\mathrm{e}}(T^+)$ are incompatible with $\mathrm{IP}_{\mathrm{c}}(T)$ and are excluded from $\mathrm{IP}^{\mathrm{e}'}_{\mathrm{c}}(T)$, resulting in different LP polytopes for their LP-relaxations.

Next, to see the impact of adding counting constraints on forward-search planning using these delete-relaxation LP models, we compare A*/$\mathrm{LP}^{\mathrm{e}'}_{\mathrm{c}}(T)$ with A*/$\mathrm{LP}^{\mathrm{e}}(T^+)$, and A*/$\mathrm{LP}^{\mathrm{e}'}_{\mathrm{ctr}}(T)$ with A*/$\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$. Coverage on the same instances as our previous experiment are shown in Table 10. There is a tradeoff between the improved search efficiency due to the additional informativeness in the heuristic provided by the counting constraints, and the additional time required to solve the LPs (because the additional constraints make the LP more difficult to solve). Table 10 shows that the overall effects of enhancing our delete-relaxation model are mixed. A*/$\mathrm{LP}^{\mathrm{e}'}_{\mathrm{c}}(T)$ attains a coverage of 672 instances, which is lower than the coverage for A*/$\mathrm{LP}^{\mathrm{e}}(T^+)$, while A*/$\mathrm{LP}^{\mathrm{e}'}_{\mathrm{ctr}}(T)$ solves 716 problems compared to the 705 problems solved by A*/$\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$. There are some domains where adding the counting constraints significantly improved coverage, including parcprinter, pathways-noneg, rovers, woodworking. On the other hand, coverage dropped significantly in elevators, freecell, openstacks as a result of adding the counting constraints. The time relaxation seems to be advantageous overall, resulting in an increase from 672 instances for A*/$\mathrm{LP}^{\mathrm{e}}(T^+)$ to 716 problems for A*/$\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$.

Table 9 also shows the value for the LMC-SEQ LP value (Pommerening et al., 2014). This combination of the landmark constraints and net change constraints in their operator-counting framework is analogous to the combination of our delete-free model with counting constraints, so it is interesting to compare their optimal LP values. $\mathrm{LP}^{\mathrm{e}'}_{\mathrm{c}}(T)$ and $\mathrm{LP}^{\mathrm{e}'}_{\mathrm{ctr}}(T)$ have a higher average value than LMC-SEQ on 16 and 15 domains, respectively, while LMC-SEQ has a higher value than both $\mathrm{LP}^{\mathrm{e}'}_{\mathrm{c}}(T)$ and $\mathrm{LP}^{\mathrm{e}'}_{\mathrm{ctr}}(T)$ on 17 domains. Thus, as with our previous comparison of LM-cut with $\mathrm{LP}^{\mathrm{e}}(T^+)$ and $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$ in Section 6.2, our delete-relaxation approach seems to be complementary to the LMC-SEQ combination in the operator-counting framework. On the other hand, comparing the results of forward search based optimal planning using these LP models, we see that FD/LMC-SEQ has significantly higher coverage than A*/$\mathrm{LP}^{\mathrm{e}'}_{\mathrm{c}}(T)$ and A*/$\mathrm{LP}^{\mathrm{e}'}_{\mathrm{ctr}}(T)$, as well as A*/$\mathrm{LP}^{\mathrm{e}'}_{\mathrm{c}}(T)$ and A*/$\mathrm{LP}^{\mathrm{e}'}_{\mathrm{ctr}}(T)$.

9. Automatic LP Model Selection

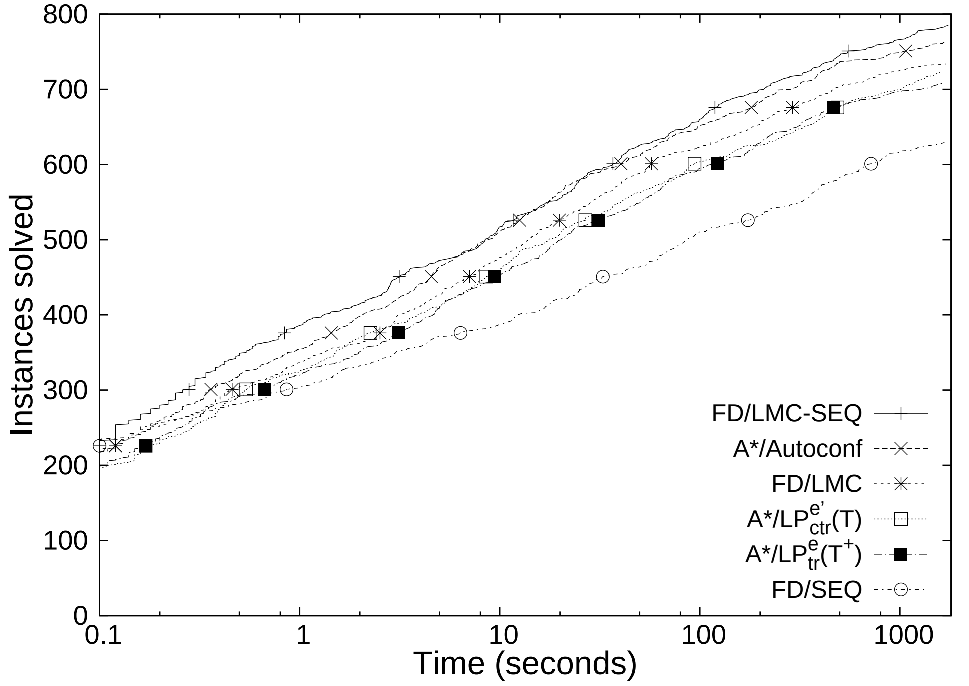
From the definitions of the models, we know that for any STRIPS planning task T with action costs, the relationships among the IP models are as follows: $\mathrm{IP}_{\mathrm{tr}}(T^+) \leq \mathrm{IP}^{\mathrm{e}}_{\mathrm{tr}}(T^+) \leq \mathrm{IP}(T^+) = \mathrm{IP}^{\mathrm{e}}(T^+) = h^+ \leq \mathrm{IP}_{\mathrm{c}}(T) = \mathrm{IP}^{\mathrm{e}'}_{\mathrm{c}}(T)$. As for the LP relaxations, we know that $\mathrm{LP}(T^+) \leq \mathrm{LP}^{\mathrm{e}}(T^+)$, $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+) \leq \mathrm{LP}^{\mathrm{e}}(T^+)$, $\mathrm{LP}^{\mathrm{e}'}_{\mathrm{ctr}}(T) \leq \mathrm{LP}^{\mathrm{e}'}_{\mathrm{c}}(T)$, and $\mathrm{LP}^{\mathrm{e}'}_{\mathrm{ctr}}(T) \leq \mathrm{LP}^{\mathrm{e}'}_{\mathrm{c}}(T)$. Note that $\mathrm{LP}^{\mathrm{e}'}_{\mathrm{c}}(T)$ does not always dominate $\mathrm{LP}^{\mathrm{e}}(T^+)$, because the dominated action elimination and immediate action application eliminate different sets of variables in these two LP models. Figure 1 illustrates the dominance relationships among the bounds.

While the time-relaxed $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$ and $\mathrm{LP}^{\mathrm{e}'}_{\mathrm{ctr}}(T)$ are dominated by the non-time-relaxed models $\mathrm{LP}^{\mathrm{e}}(T^+)$ and $\mathrm{LP}^{\mathrm{e}'}_{\mathrm{c}}(T)$, respectively, the time-relaxed LPs are significantly cheaper to compute than their non-relaxed counterparts.

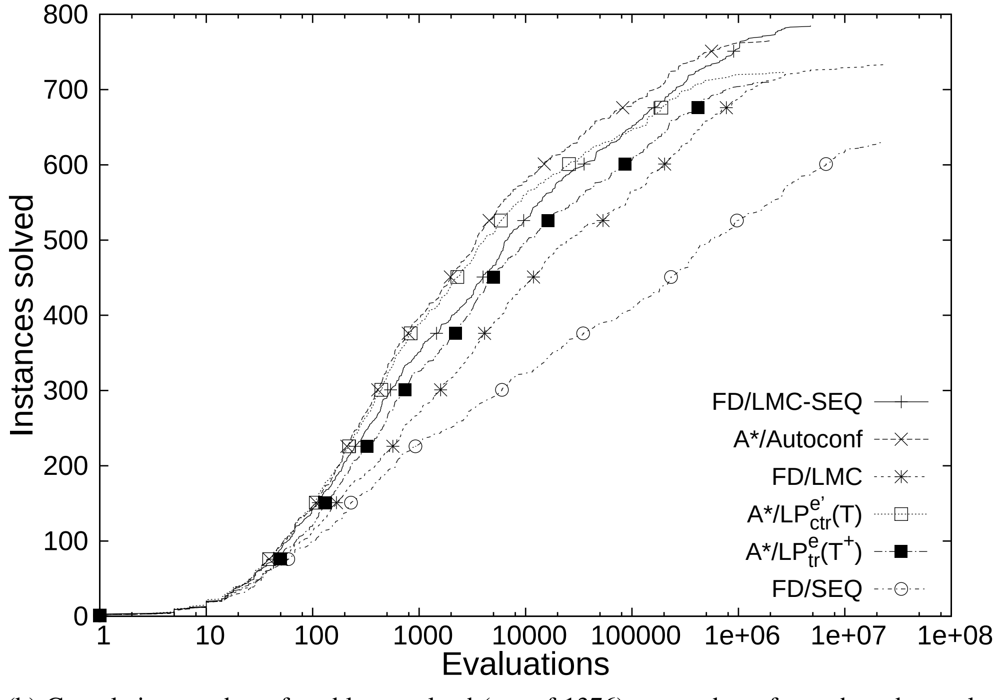
Similarly, although $\mathrm{IP}^{\mathrm{e}'}_{\mathrm{c}}(T)$ dominates $\mathrm{IP}^{\mathrm{e}}(T^+)$, it is possible for $\mathrm{LP}^{\mathrm{e}}(T^+)$ to be larger than $\mathrm{LP}^{\mathrm{e}'}_{\mathrm{c}}(T)$. Furthermore, if two LPs have the same optimal value, the one that can be solved faster is clearly preferable because the LPs must be solved at each node in the A* search. Thus, we have a set

	LMC-SEQ	$\mathrm{LP}^{\mathrm{e}}(T^+)$	$\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$	$\mathrm{LP}^{\mathrm{e}'}_{\mathrm{c}}(T)$	$\mathrm{LP}^{\mathrm{e}'}_{\mathrm{ctr}}(T)$
airport	**1.00**	.85	.85	.98	.98
barman-opt11	2.23	.51	.51	**3.59**	**3.59**
blocks	**1.07**	1.00	1.00	**1.07**	**1.07**
depot	1.10	1.43	1.42	**1.54**	**1.54**
driverlog	1.04	1.01	.99	**1.12**	**1.12**
elevators-opt08	**1.02**	.84	.82	.71	.71
elevators-opt11	**1.01**	.80	.77	.67	.67
floortile-opt11	1.05	1.01	1.01	**1.08**	**1.08**
freecell	2.64	**3.14**	3.07	3.08	3.07
grid	1.09	1.20	1.19	**1.55**	**1.55**
gripper	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
logistics98	1.00	.91	.90	**1.01**	**1.01**
logistics00	**1.00**	.99	.99	**1.00**	**1.00**
miconic	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
no-mprime	**1.00**	.89	.78	.82	.82
no-mystery	**1.01**	.98	.90	.84	.81
nomystery-opt11	1.03	1.07	1.07	**1.10**	**1.10**
openstacks	1.36	**1.61**	1.43	**1.61**	1.43
openstacks-opt08	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
openstacks-opt11	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
parcprinter-08	**1.08**	1.00	1.00	**1.08**	**1.08**
parcprinter-opt11	**1.05**	1.00	1.00	**1.05**	**1.05**
parking-opt11	1.00	1.04	.99	**1.06**	1.00
pathways-noneg	1.53	1.13	1.13	**1.72**	**1.72**
pegsol-08	**1.34**	1.09	1.05	1.25	1.23
pegsol-opt11	**1.33**	1.10	1.10	1.22	1.22
pipesworld-notankage	1.45	1.18	1.16	**1.73**	1.70
pipesworld-tankage	1.32	1.27	1.26	**1.35**	1.20
psr-small	2.60	1.00	1.00	**2.61**	**2.61**
rovers	**1.23**	.72	.72	.81	.81
satellite	**1.00**	.83	.75	.85	.75
scanalyzer-08	**1.00**	.98	.94	.97	.93
scanalyzer-opt11	**1.01**	.98	.98	.97	.97
sokoban-opt08	1.15	1.01	1.00	**1.13**	**1.13**
sokoban-opt11	1.11	1.01	1.01	**1.12**	**1.12**
tpp	**1.43**	.89	.89	1.42	1.42
transport-opt08	**1.11**	.49	.49	.18	.18
transport-opt11	**1.08**	.49	.49	.18	.18
trucks	1.00	**1.08**	**1.08**	**1.08**	**1.08**
visitall-opt11	**1.50**	1.42	1.41	1.48	1.47
woodworking-opt08	1.04	1.12	1.12	**1.18**	**1.18**
woodworking-opt11	1.05	1.13	1.13	**1.19**	**1.19**
zenotravel	**1.00**	.96	.94	.94	.93

Table 9: Optimal values of LP models relative to LM-cut value for 1376 IPC instances. Means for each domain are shown. E.g., for barman-opt11, the mean LMC-SEQ value was 2.23 times the LM-cut value, the $\mathrm{LP}^{\mathrm{e}}(T^+)$ and $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$ values were 0.51 times the LM-cut value, and the $\mathrm{LP}^{\mathrm{e}'}_{\mathrm{c}}(T)$ and $\mathrm{LP}^{\mathrm{e}'}_{\mathrm{ctr}}(T)$ values were 3.59 times the LM-cut value.



(a) Cumulative number of problems solved (out of 1376) vs time (30 minute time limit).



(b) Cumulative number of problems solved (out of 1376) vs number of search nodes evaluated (30 minute time limit).

Figure 8: Comparison of forward search (A*) planners, part 2.

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. It is difficult to choose the best heuristic *a priori* because in general, we do not know (1) whether it is worthwhile to use the counting constraints or not, and (2) whether the time-relaxation is tight or not for a particular problem instance.

Thus, we implemented a simple mechanism for automatically selecting the LP to be used for each problem which works as follows: First, we compute $\mathrm{LP}^{\mathrm{e}}(T^{+})$, $\mathrm{LP}^{\mathrm{e}'}_{\mathrm{c}}(T)$, $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^{+})$, and $\mathrm{LP}^{\mathrm{e}'}_{\mathrm{ctr}}(T)$ 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, and break ties by choosing the heuristic that is cheapest to compute. Although the "cheapest" heuristic could be identified according to the CPU time required to compute each heuristic, for many problems, the computations are too fast for robust timing measurements, so we simply break ties in order of $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^{+})$, $\mathrm{LP}^{\mathrm{e}'}_{\mathrm{ctr}}(T)$, $\mathrm{LP}^{\mathrm{e}}(T^{+})$, $\mathrm{LP}^{\mathrm{e}'}_{\mathrm{c}}(T)$, because this ordering usually accurately reflects the timing order. This mechanism makes the simplistic assumption that ranking and behavior of the LP bounds at the root node will be similar to the ranking of the LP bounds throughout the search graph. A more sophisticated method for heuristic selection may result in better performance (c.f. Domshlak, Karpas, & Markovitch, 2012), and is an avenue for future work.

9.1 Experimental Results for Automated Model Selection and Comparison with the State-of-the-Art

We compared A* using our LP-based heuristics, including A*/autoconf, with state-of-the-art heuristics. Specifically, we compared:

- FD/LM-cut : Fast Downward using the landmark cut heuristic (Helmert & Domshlak, 2009) (the standard `seq-opt-lmcut` configuration)
- FD/LMC : Fast Downward using an LP-model for the optimal cost partitioning for landmark cut constraints (Pommerening et al., 2014)
- FD/SEQ : Fast Downward using the lower-bound net change constraints (Pommerening et al., 2014), corresponding to the state-equation heuristic by Bonet (2013).
- FD/OPT-SYS1, FD/PHO-SYS1, FD/PHO-SYS2 : Fast Downward using optimal cost partitioning constraints for projections on goal variables (OPT-SYS1), and post-hoc optimization constraints (PHO-SYS1, PHO-SYS2) (Pommerening et al., 2014).
- FD/LMC-SEQ : Fast Downward using both the landmark cut and net change constraints.
- A*/$\mathrm{LP}^{\mathrm{e}}(T^{+})$: A* with the LP relaxation of the enhanced delete-free IP model $\mathrm{IP}^{\mathrm{e}}(T^{+})$ (Section 4) as a heuristic.
- A*/$\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^{+})$: A* with the LP relaxation of the time-relaxed, enhanced delete-free IP model $\mathrm{IP}^{\mathrm{e}}(T^{+})$ as a heuristic.
- A*/$\mathrm{LP}^{\mathrm{e}'}_{\mathrm{c}}(T)$: A* with the LP relaxation of the enhanced delete-free IP model with counting constraints $\mathrm{IP}^{\mathrm{e}'}_{\mathrm{c}}(T)$ as a heuristic.
- A*/$\mathrm{LP}^{\mathrm{e}'}_{\mathrm{ctr}}(T)$: A* with the LP relaxation of the time-relaxed, enhanced delete-free IP model with counting constraints $\mathrm{IP}^{\mathrm{e}'}_{\mathrm{c}}(T)$ as a heuristic.

9.1.1 COVERAGE RESULTS

The coverage results (number of problems solved) are shown in Tables 10. The time spent at the root node by A*/autoconf for LP model selection is included in the runtimes, and also counts against the 30-minute runtime limit. Figures 8a-8b show the cumulative number of instances solved as a function of the number time and number of node evaluations, respectively (for legibility, only a subset of the algorithms are included in Figures 8a-8b). Table 11 shows a summary of total coverage results for all forward-search configurations that are included in Tables 8 and 10.

Our results indicate that automatic LP model selection significantly boosts the performance of an A*-based planner compared to relying on a single LP model. A*/autoconf achieved a coverage of 761 out of 1376 instances, which is significantly better than its 4 individual components. Furthermore, A*/autoconf attained higher coverage than all other solver configurations in Table 10 except for FD/LMC-SEQ (Pommerening et al., 2014), which solved 781 instances. Note that A*/autoconf has higher coverage than FD/LMC-SEQ on 11/43 domains (floortile-opt11, freecell, grid, logistics98, nomystery-opt11, pathways-noneg, rovers, satellite, trucks, woodworking-opt08, woodworking-opt11).

9.1.2 ACCURACY OF A*/AUTOCONF MODEL SELECTION

We analyzed the accuracy of the model selection by evaluating the performance of A*/autoconf on each problem instance vs the performance of each of its four component models. If only coverage is considered, then in 96.4% of the instances, A*/autoconf made the correct decision with respect to coverage, where the model selection by A*/autoconf was deemed to be correct if either A*/autoconf solved the problem instance, or none of the 4 components solved the problem instance. On the other hand, when runtimes are considered as well as coverage, then in 83.0% of the instances, A*/autoconf made the correct decision, where the selection was deemed to be correct if A*/autoconf selected the model that had the best runtime (including ties), or none of the 4 components solved the problem instance. As a baseline, $\mathrm{LP}^{\mathrm{e}'}_{\mathrm{ctr}}(T)$, which had the best coverage among all of the component models, is the "correct" choice according to this criterion 49.9% of the time. Mistakes in the selections made by A*/autoconf can be seen in Table 10 coverage results – for example, in the woodworking-opt11 domain, A*/autoconf solved 18 instances compared 20 instances solved by $\mathrm{LP}^{\mathrm{e}'}_{\mathrm{ctr}}(T)$. Thus, there is significant room for improvement when runtimes are considered in addition to coverage, and improving the model selection using machine learning techniques is a direction for future work.

10. Discussion and Conclusion

This paper proposed a new, integer-linear programming formulation of the delete relaxation h^+ for cost-optimal, domain-independent planning. We started with a basic IP model $\mathrm{IP}(T^+)$, and showed that an enhanced model $\mathrm{IP}^{\mathrm{e}}(T^+)$, which incorporates landmark-based variable reduction, relevance analysis, and action elimination, is competitive with previous methods for solving delete-free versions of the standard IPC planning benchmarks tasks (i.e., exact computation of h^+).

The results of embedding our IP model as the heuristic function in a A*-based forward search planner confirmed that the plain $\mathrm{IP}(T^+)$ model is not practical (coverage of 403/1367 instances vs. 540 for Fast Downward using h^{max}). However, we showed that the $\mathrm{IP}^{\mathrm{e}}(T^+)$ model, which uses variable reduction methods to reduce the size of the IP models and exactly computes h^+, performed much better, with a coverage of 635 instances. According to the summary results in

Domain	FD/LM-cut	FD/LMC	FD/LMC-SEQ	FD/OPT-SYS1	FD/PHO-SYS1	FD/PHO-SYS2	FD/SEQ	A*/LP$^{\mathrm{e}}(T^+)$	A*/LP$^{\mathrm{e}}_{\mathrm{tr}}(T^+)$	A*/LP$^{\mathrm{e}'}_{\mathrm{c}}(T)$	A*/LP$^{\mathrm{e}'}_{\mathrm{ctr}}(T)$	A*/autoconf
airport(50)	28	28	**30**	20	22	28	22	25	25	25	25	25
barman-opt11(20)	**4**	**4**	**4**	**4**	**4**	**4**	**4**	0	0	0	3	2
blocks(35)	28	28	**29**	26	28	27	28	28	28	**29**	**29**	**29**
depot(22)	**7**	**7**	**7**	4	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**
driverlog(20)	**14**	13	13	10	12	13	12	11	13	12	13	13
elevators-opt08(30)	**22**	20	19	8	11	19	10	13	13	6	8	13
elevators-opt11(20)	**18**	16	16	6	9	16	8	10	10	4	6	10
floortile-opt11(20)	**7**	6	6	2	2	2	4	6	**7**	6	**7**	**7**
freecell(80)	15	15	33	8	15	15	39	**44**	43	17	21	**44**
grid(5)	2	2	2	1	1	2	1	2	2	2	**3**	**3**
gripper(20)	**7**	6	6	6	**7**	**7**	**7**	6	6	6	6	6
logistics98(35)	6	6	6	2	4	5	4	6	6	**7**	**7**	**7**
logistics00(28)	20	20	20	14	16	**21**	16	20	20	20	20	20
miconic(150)	**141**	**141**	**141**	45	50	54	52	140	**141**	139	140	**141**
no-mprime(35)	**23**	**23**	22	19	21	21	20	18	17	15	16	18
no-mystery(30)	**16**	**16**	**16**	13	15	15	15	13	12	11	11	12
nomystery-opt11(20)	14	14	12	8	12	**16**	10	14	14	8	11	14
openstacks(30)	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**
openstacks-opt08(30)	**19**	**19**	16	11	**19**	**19**	17	11	11	6	10	11
openstacks-opt11(20)	**14**	**14**	11	6	**14**	**14**	12	6	6	2	5	6
parcprinter-08(30)	19	18	**29**	11	15	17	28	20	20	**29**	**29**	**29**
parcprinter-opt11(20)	14	13	**20**	7	11	13	**20**	16	16	**20**	**20**	**20**
parking-opt11(20)	3	2	2	1	**5**	1	4	1	1	1	1	1
pathways-noneg(30)	5	5	5	4	4	4	4	5	5	**14**	**14**	**14**
pegsol-08(30)	27	27	**28**	22	27	27	**28**	26	26	22	26	26
pegsol-opt11(20)	17	17	**18**	12	17	17	**18**	16	16	12	16	16
pipesworld-notankage(50)	**17**	**17**	14	13	14	16	15	12	13	12	13	13
pipesworld-tankage(50)	**8**	**8**	**8**	7	**8**	**8**	**8**	7	7	7	7	7
psr-small(50)	49	49	**50**	48	49	49	**50**	48	48	**50**	**50**	**50**
rovers(40)	7	7	7	6	6	6	6	7	7	**11**	**11**	**11**
satellite(36)	7	7	7	5	6	6	6	**10**	**10**	9	9	**10**
scanalyzer-08(30)	**15**	14	14	10	12	7	14	8	8	7	8	8
scanalyzer-opt11(20)	**12**	11	11	7	9	4	11	5	5	4	5	5
sokoban-opt08(30)	**30**	28	29	18	24	29	20	23	25	22	26	25
sokoban-opt11(20)	**20**	**20**	**20**	15	19	**20**	17	19	19	19	19	19
tpp(30)	6	6	**8**	6	6	6	**8**	6	6	**8**	**8**	**8**
transport-opt08(30)	**11**	**11**	**11**	9	**11**	**11**	**11**	9	10	6	6	10
transport-opt11(20)	**6**	**6**	**6**	4	**6**	**6**	**6**	4	5	1	1	5
trucks(30)	10	10	10	3	6	7	9	**15**	**15**	12	**15**	**15**
visitall-opt11(20)	11	10	**19**	15	16	16	17	16	16	17	17	17
woodworking-opt08(30)	17	16	21	8	10	16	14	16	17	**30**	**30**	28
woodworking-opt11(20)	12	11	16	3	5	11	9	10	11	**20**	**20**	18
zenotravel(20)	**13**	12	12	8	9	11	9	10	11	10	10	11
Total coverage (1376)	748	730	781	462	571	620	627	696	705	672	716	761
# Best domains	22	13	18	2	10	12	12	5	6	12	15	16

Table 10: Comparison of forward search (A*) planners, part 2: Number of problems solved with 30 minute, 2GB RAM limit using A* and our IP/LP models as heuristic functions. Includes LP models that incorporate counting constraints (LP$^{\mathrm{e}'}_{\mathrm{c}}(T)$, LP$^{\mathrm{e}'}_{\mathrm{ctr}}(T)$, Section 8), as well as A*/autoconf (Section 9). Comparison with Fast Downward using operator-counting LP models (Pommerening et al., 2014).

Configuration	# solved	Description
FD/LM-cut	748	Fast Downward (FD) using standard Landmark Cut heuristic (`seq-opt-lmcut`)
FD/$h^{\max}$	540	FD using $h^{\max}$ heuristic
FD/SEQ	627	FD using SEQ LP heuristic (Pommerening et al., 2014)
FD/PHO-SYS1	571	FD using PHO-SYS1 LP heuristic (Pommerening et al., 2014)
FD/PHO-SYS2	620	FD using PHO-SYS2 LP heuristic (Pommerening et al., 2014)
FD/LMC	730	FD using LP model of optimal cost partitioning on landmark constraints (Pommerening et al., 2014)
FD/OPT-SYS1	462	FD using OPT-SYS1 LP heuristic (Pommerening et al., 2014)
FD/LMC-SEQ	781	FD using LMC+SEQ LP heuristic (Pommerening et al., 2014)
A*/HST/CPLEX	398	hsp planner using A* and h^+ heuristic (Haslum et al., 2012; Haslum, 2012)
A*/IP(T^+)	403	basic IP formulation for h^+
A*/IP$^{\mathrm{e}}(T^+)$	635	IP(T^+) with all enhancements in Sections 4.1-4.6
A*/LP$^{\mathrm{e}}(T^+)$	696	LP relaxation of IP$^{\mathrm{e}}(T^+)$
A*/LP$^{\mathrm{e}}_{\mathrm{tr}}(T^+)$	705	LP relaxation of the time-relaxed model IP$^{\mathrm{e}}_{\mathrm{tr}}(T^+)$
A*/LP$^{\mathrm{e}'}_{\mathrm{c}}(T)$	672	LP relaxation of IP$^{\mathrm{e}'}_{\mathrm{c}}(T)$
A*/LP$^{\mathrm{e}'}_{\mathrm{ctr}}(T)$	716	LP relaxation of the time-relaxed model IP$^{\mathrm{e}'}_{\mathrm{ctr}}(T)$
A*/autoconf	761	Automated selection of LP at root node(Section 9)

Table 11: Summary of coverage (# solved) on 1376 IPC benchmark problems instances with 30 minute time limit and 2GB RAM (see Tables 8-10 for detailed results)

Table 11, the aggregate coverage of IP$^{\mathrm{e}}(T^+)$ is comparable to the coverage obtained by the LP-based SEQ, OPT-SYS1, PHO-SYS1, and PHO-SYS2 heuristics recently implemented using the operator-counting framework by Pommerening et al. (2014). However, the aggregate coverage on the IPC benchmarks is skewed by the miconic domain, where SEQ, OPT-SYS1, PHO-SYS1, and PHO-SYS2 perform particularly poorly compared to other heuristics. If the miconic domain is not included, then IP$^{\mathrm{e}}(T^+)$ is not competitive with these LP-based models. Note that on the freecell domain, A* with the IP$^{\mathrm{e}}(T^+)$ heuristic solved 54/80 instances, which is significantly higher than all other methods, so there is at least 1 domain where exact h^+ computation using the IP$^{\mathrm{e}}(T^+)$ model performs extremely well compared to other state-of-the-art heuristics.

We then showed that the gap between the optimal value of the LP relaxations of our IP models and h^+ tended to be quite small (the gap was often zero), suggesting that the LP relaxations, which can be computed much faster than the IP models, could be used as a heuristic for A*-based planning. A time-relaxation that eliminates all time-related constraints was also proposed as another way to reduce the model in order to be solvable faster. A comparison of our LP-relaxed delete relaxation models with the LM-cut (Helmert & Domshlak, 2009) heuristic values showed that these approaches are complementary with respect to how closely they approximate h^+. Thus, the LP-relaxation of our delete-free models provides a novel, practical alternative to approximating h^+. We showed that A* search using LP$^{\mathrm{e}}(T^+)$ (LP-relaxation of delete-free task) and LP$^{\mathrm{e}}_{\mathrm{tr}}(T^+)$ (time relaxed, LP-relaxation of delete-free task) significantly improves upon the IP models, solving 696 and 705 instances, respectively, making them usable as practical heuristics.

A major advantage of LP-based heuristics is the relative ease with which additional constraints can be added in order to obtain improved heuristics. We showed that the counting constraints, corresponding to the net change constraints proposed in previous work (van den Briel et al., 2007; Pommerening et al., 2014), could be added to our LP model. The resulting heuristic, LP$^{\mathrm{e}'}_{\mathrm{ctr}}(T)$ had mixed results, improving performance on some domains, but degrading performance on other domains, i.e., LP$^{\mathrm{e}}_{\mathrm{tr}}(T^+)$ and LP$^{\mathrm{e}'}_{\mathrm{ctr}}(T)$ are complementary heuristics.

Since there is no dominance relationship among $A^*/\mathrm{LP}^{\mathrm{e}}(T^+)$, $A^*/\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$, $A^*/\mathrm{LP}^{\mathrm{e}'}_{\mathrm{c}}(T)$ and $A^*/\mathrm{LP}^{\mathrm{e}'}_{\mathrm{ctr}}(T)$, we proposed A*/autoconf , a simple method which automatically selects among these 4 heuristics by computing all 4 heuristic values at the root node and using the most accurate heuristic (breaking ties according to speed). We showed that overall, A*/autoconf significantly improves upon its 4 components, and is competitive with the landmark-cut heuristic, solving 761/1367 instances and achieving state-of-the-art performance on several domains.

While A*/autoconf has lower total coverage compared to Fast Downward using the LMC-SEQ LP-based heuristic (Pommerening et al., 2014), the $\mathrm{LP}(T^+)$-based approach outperforms LMC-SEQ on several domains including freecell, pathways-noneg, rovers, satellite, trucks, and woodworking. Although A*/autoconf includes LP models with counting constraints that consider some delete effects, note that A*/$\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$, which uses a "pure" delete-free LP, performs quite well, obtaining higher coverage than all of the operator-count based heuristics of Pommerening et al. (2014) in the floortile, freecell, nomystery-opt11, satellite, and trucks domains, so the counting constraints are *not* required in order for A* using the delete-relaxation based LPs to achieve state-of-the-art performance on some domains.

A comparison of the optimal values of our counting-constraint enhanced delete-relaxation LP models $\mathrm{LP}^{\mathrm{e}'}_{\mathrm{c}}(T)$ and $\mathrm{LP}^{\mathrm{e}'}_{\mathrm{ctr}}(T)$ with the optimal LP values of the LMC-SEQ model showed that they are complementary, with each class of models outperforming the other on roughly the same number of domains (Section 8.1). Thus, integrating these two approaches in a single LP model is a promising direction for future work. In a recent survey of LP-based heuristics for planning, Röger and Pommerening (2015) noted that our delete-relaxation model can be incorporated into the operator counting framework of Pommerening et al. (2014) by adding operator-counting variables for each operator in the delete-relaxed problem – this is a promising direction for future work. Note that while both Pommerening et al. (2014) and our approach use landmarks, they are used for very different purposes. The landmark constraints used by Pommerening et al. (2014) are used directly as operator counting constraints. In contrast, our approach uses landmarks order to decrease the size of the IP/LP models for the delete-free task and is used for the purpose of speeding up the computation of the IP/LP models, i.e., landmark based reduction does not change the optimal value of $\mathrm{IP}(T^+)$.

We showed that adding counting constraints that consider some delete effects (i.e., $\mathrm{LP}^{\mathrm{e}'}_{\mathrm{c}}(T)$ and $\mathrm{LP}^{\mathrm{e}'}_{\mathrm{ctr}}(T)$) can improve performance on some domains, but in some domains, coverage dropped significantly. This is because the additional constraints make the LP more difficult to solve, so the increased search efficiency due to the tighter bound is not enough to overcome the increased cost of solving the LP at each search node. A*/autoconf attempts to address this by selecting the models with counting constraints only when they return a higher value than the model without counting constraints at the root node, and otherwise uses a model that does not include the counting constraints (i.e., $\mathrm{LP}^{\mathrm{e}}(T^+)$ or $\mathrm{LP}^{\mathrm{e}}_{\mathrm{tr}}(T^+)$). On the other hand, strengthening the delete-relaxation by considering some of the delete effects has been an active area of research, and recently, two frameworks that allow flexible interpolation between the delete relaxation and the original model have been proposed. Keyder, Hoffmann, and Haslum (2014) propose an approach which adds new fluents that represent conjunctions of fluents in the original planning task. Red-black planning (Domshlak, Hoffmann, & Katz, 2015) is a framework which separates state variables into two groups – red variables which are relaxed, and black variables that are not relaxed. Combining these flexible relaxation frameworks with our IP approach and developing a more principled approach to deciding when to use counting constraints is an avenue for future work.

Our current implementation uses the CPLEX solver naively, relying entirely on default control parameters. Systematically tuning and improving the implementation of our IP/LP models in order to make better use of incremental IP/LP solving capabilities is a promising direction for future work.

Although we have shown that our LP models often compute h^+ exactly, there are some domains where there are significant gaps between h^+ and the optimal cost of the LP models. Improved modeling techniques may allow tighter LP bounds. For example, Constraint C6 uses straightforward a big-M encoding, and it may be possible to obtain tighter bounds using other methods.

Furthermore, although solving and IP at each node in a forward-search based planner has previously been considered impractical, we have shown that our IP$^{\mathrm{e}}(T^+)$ model, which computes h^+ exactly, is almost useful as a practical heuristic, and improving the techniques used to solve the IP for the IP$^{\mathrm{e}}(T^+)$ may result in a balance of accuracy and speed necessary for a practical general purpose heuristic. For example, significant performance improvements might be obtainable by improving the use of the IP solver. For example, in contrast to LP solvers, where parallel speedups are often difficult to obtain, IP solvers can often be sped up significantly by parallelization, and current IP solvers already provide parallel search algorithms (which we did not use in this paper because we limited our experiments to single threads). As the number of cores per processor continues to increase, it is possible that in some cases, IP-based heuristics may become more useful than LP-based heuristics.

Acknowledgments

Thanks to Patrik Haslum for assistance with his code for computing h^+ and his hsp_f planner. Thanks to Florian Pommerening for assistance with the code for the LP heuristic-based Fast Downward (Pommerening et al., 2014). Thanks to the anonymous reviewers for numerous helpful suggestions which significantly improved the paper. This research was supported by a JSPS Grant-in-Aid for JSPS Fellows and a JSPS KAKENHI grant.

References

Bäckström, C., & Nebel, B. (1995). Complexity Results for SAS$^+$ Planning. *Computational Intelligence*, *11*(4), 625–655.

Betz, C., & Helmert, M. (2009). Planning with h^+ in theory and practice. In *KI 2009*, pp. 9–16. Springer.

Blum, A., & Furst, M. (1997). Fast Planning Through Planning Graph Analysis. *Artificial Intelligence*, *90*(1-2), 281–300.

Bonet, B. (2013). An admissible heuristic for SAS+ planning obtained from the state equation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2268–2274.

Bonet, B., & Castillo, J. (2011). A complete algorithm for generating landmarks. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*.

Bonet, B., & Geffner, H. (2001). Planning as heuristic search. *Artificial Intelligence*, *129*(1-2), 5–33.

Bonet, B., & Helmert, M. (2010). Strengthening landmark heuristics via hitting sets. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pp. 329–334.

Bonet, B., & van den Briel, M. (2014). Flow-based heuristics for optimal planning: Landmarks and merges. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*.

Bylander, T. (1994). The Computational Complexity of Propositional STRIPS Planning. *Artificial Intelligence*, *69*(1–2), 165–204.

Bylander, T. (1997). A linear programming heuristic for optimal planning. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 694–699.

Cooper, M. C., de Roquemaurel, M., & Regnier, P. (2011). Transformation of optimal planning problems. *Journal of Experimental & Theoretical Artificial Intelligence*, *23*(2), 181–199.

Dimopoulos, Y. (2001). Improved integer programming models and heuristic search for ai planning. In *Proceedings of 6th European Conference on Planning (ECP)*, pp. 50–57.

Domshlak, C., Karpas, E., & Markovitch, S. (2012). Online speedup learning for optimal planning. *Journal of Artificial Intelligence Research*, *44*, 709–755.

Domshlak, C., Hoffmann, J., & Katz, M. (2015). Red-black planning: A new systematic approach to partial delete relaxation. *Artificial Intelligence*, *221*, 73–114.

Gefen, A., & Brafman, R. (2011). The minimal seed set problem. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 319–322.

Gefen, A., & Brafman, R. (2012). Pruning methods for optimal delete-free planning. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 56–64.

Haslum, P. (2012). Incremental lower bounds for additive cost planning problems. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 74–82.

Haslum, P. (2014a) Personal communication.

Haslum, P. (2014b). Hsp* code and documentatoin http://users.cecs.anu.edu.au/patrik/un-hsps.html..

Haslum, P., Slaney, J., & Thiébaux, S. (2012). Minimal landmarks for optimal delete-free planning. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 353–357.

Helmert, M., & Domshlak, C. (2009). Landmarks, critical paths and abstractions: What's the difference anyway?. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 162–169.

Hoffmann, J., & Nebel, B. (2001). The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research*, *14*, 253–302.

Hoffmann, J., Porteous, J., & Sebastia, L. (2004). Ordered landmarks in planning. *Journal of Artificial Intelligence Research*, *22*, 215–278.

Imai, T., & Fukunaga, A. (2014). A practical, integer-linear programming model for the delete-relaxation in cost-optimal planning. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*.

Karpas, E., & Domshlak, C. (2009). Cost-optimal planning with landmarks. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1728–1733.

Katz, M., & Domshlak, C. (2010). Optimal admissible composition of abstraction heuristics. *Artificial Intelligence*, *174*(12-13), 767–798.

Kautz, H., & Selman, B. (1992). Planning as Satisfiability. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pp. 359–363.

Kautz, H. A., & Selman, B. (1996). Pushing the envelope: Planning, propositional logic and stochastic search. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 1194–1201.

Kautz, H. A., & Selman, B. (1999). Unifying sat-based and graph-based planning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 318–325.

Keyder, E., Richter, S., & Helmert, M. (2010). Sound and complete landmarks for and/or graphs. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pp. 335–340.

Keyder, E., & Geffner, H. (2008). Heuristics for planning with action costs revisited. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pp. 588–592.

Keyder, E. R., Hoffmann, J., & Haslum, P. (2014). Improving delete relaxation heuristics through explicitly represented conjunctions. *Journal of Artificial Intelligence Research*, *50*, 487–533.

Mirkis, V., & Domshlak, C. (2007). Cost-sharing approximations for h+. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 240–247.

Pommerening, F., & Helmert, M. (2012). Optimal planning for delete-free tasks with incremental LM-cut. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 363–367.

Pommerening, F., Röger, G., Helmert, M., & Bonet, B. (2014). LP-based heuristics for cost-optimal planning. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*.

Pommerening, F., Röger, G., & Helmert, M. (2013). Getting the most out of pattern databases for classical planning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Rintanen, J. (2012). Planning as satisfiability: Heuristics. *Artificial Intelligence*, *193*, 45–86.

Rintanen, J., Heljanko, K., & Niemelä, I. (2006). Planning as satisfiability: parallel plans and algorithms for plan search. *Artificial Intelligence*, *170*(12-13), 1031–1080.

Robinson, N. (2012). *Advancing Planning-as-Satisfiability*. Ph.D. thesis, Griffith University.

Robinson, N., McIlraith, S. A., & Toman, D. (2014). Cost-based query optimization via AI planning. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, pp. 2344–2351.

Röger, G., & Pommerening, F. (2015). Linear programming for heuristics in optimal planning. In *AAAI2015 Workshop on Planning, Search, and Optimization*.

van den Briel, M. (2015) Personal communication.

van den Briel, M., Benton, J., Kambhampati, S., & Vossen, T. (2007). An LP-based heuristic for optimal planning. In *Proceedings of International Conference on Principles and Practice of Constraint Programming (CP)*.

van den Briel, M., & Kambhampati, S. (2005). Optiplan: A planner based on integer programming. *Journal of Artificial Intelligence Research*, *24*, 919–931.

van den Briel, M., Vossen, T., & Kambhampati, S. (2008). Loosely coupled formulation for automated planning: An integer programming perspective. *Journal of Artificial Intelligence Research*, *31*, 217–257.

Vossen, T., Ball, M. O., Lotem, A., & Nau, D. S. (1999). On the use of integer programming models in AI planning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 304–309.

Zhu, L., & Givan, R. (2003). Landmark extraction via planning graph propagation. In *Proceedings of ICAPS Doctoral Consortium*, pp. 156–160.