MIP Formulations for Delete-Free AI Planning

Salvagnin Domenico* Zanella Matteo†

Abstract

We investigate existing Mixed Integer Programming (MIP) formulations for cost-optimal delete-free STRIPS planning: these models are built by enforcing acyclicity in the underlying causal relation graph using time labeling and vertex elimination methods. We then propose two new approaches to modeling acyclicity, one based on subtour elimination constraints and the other based on disjunctive landmark constraints. In addition, we propose to warm-start the models with the landmarks computed by the LM-cut heuristic, and describe a simple greedy primal heuristic to provide a starting feasible solution to the MIP solver. Our results demonstrate that the proposed techniques outperform the current state of the art.

1 Introduction

AI planning is a field of artificial intelligence that deals with *planning tasks*, i.e., problems where we look for a cost-optimal sequence (a *plan*) of actions to reach a desired goal state from some initial state. In its classical version, the current state is described by a finite set of boolean variables, called *facts*, and each action is characterized by its cost, by its prerequisites (a subset of the facts that need to be true for the action to be applicable), and by its effects, i.e., which facts become true (*add* effects) and which become false (*delete* effects) as the result of applying the action. Solving a planning task is in general very challenging, as both its feasibility and optimality versions are known to be PSPACE-complete [7].

The *delete-free relaxation* of a planning task is a relaxation where for each action we drop its delete effects: in other words, in a delete-free planning task, once a fact becomes true it can never become false again. This property has far reaching consequences: each action can be applied at most once, as there is no benefit in using an action multiple times, feasibility can be checked in polynomial time, and the length of any optimal plan becomes polynomial in the number of actions and facts. At the same time, computing the cost h^+ of the optimal delete-free plan is NP-hard [7].

The *delete-free relaxation* has been the subject of extensive research, for several reasons. First of all, being a relaxation, h^+ provides an *admissible* heuristic for A^* search, which is one of the state of the art approaches for domain independent planning. Second, it provides a measure of informativeness for other common heuristic functions, such as h^{max} [5] and *LM-cut* [19], that are admissible for the delete-free relaxation of a task, and thus dominated by h^+. Another reason for the importance of delete-free planning is that optimal plans for general planning tasks can be computed by iteratively solving and reformulating delete-relaxed tasks [16]. Finally, some combinatorial problems, like the minimal seed-set problem [15] and determining join orders in relational database query plan generation [26], can be naturally modeled as delete-free problems, so delete-free planning is interesting in its own right.

*Department of Information Engineering, University of Padova, Email: salvagni@dei.unipd.it

†Department of Information Engineering, University of Padova, Email: matteo.zanella.3@phd.unipd.it

The current state of the art for computing h^+ in a domain independent setting consists in formulating the problem as a combinatorial optimization problem, i.e., a declarative set of variables and constraints, and using an off-the-shelf solver (SAT, CSP or MIP, depending on the chosen paradigm). In this paper, we consider MIP formulations of delete-free planning tasks.

In section 3, we review the two MIP formulations known from the literature, namely the time-stamp formulation by [21] and the formulation based on vertex elimination by [11], which is the current state of the art. Those two formulations share a common base structure, but differ in the way they model acyclicity, i.e., the fact that we cannot have a circular dependency between actions in any feasible plan. The former uses a more compact encoding based on time-stamps, that results in smaller problems but also weaker linear programming (LP) relaxations. The latter is based on a vertex elimination order of the *causal relation graph*: this requires a number of constraints that is the worst case cubic w.r.t. the number of facts in the task, and thus quite expensive as a MIP encoding, in particular on large-scale tasks, yet it provides provably stronger LP relaxations, and thus eventually proves to be the most effective method overall.

Both the formulations in [21] and [11] have polynomial size, and can thus be solved by just feeding them into a blackbox MIP solver. Inspired by the connection to the *traveling salesman problem* (TSP), where we also have to model acyclicity, but the current state of the art is *not* based on polynomial formulations, in section 4 we introduce two new models with an exponential number of constraints, thus requiring the dynamic separation of so-called violated *lazy constraints* on the fly during the solution process. The first model is based on *subtour elimination constraints* [10], again inspired by the TSP, while the second is based on the formulation of a delete-free planning task as a *hitting set* problem over a set of (exponentially many) *disjunctive landmark* constraints, an approach dating back to [4] and [17].

In section 5, we investigate how to improve the performance of the formulations described, both old and new. In detail, we noticed empirically that the MIP solver sometimes struggled in finding a good quality feasible solution early on in the search, which is unusual and quite surprising, given that it is always trivial to find a feasible solution in the delete-free case: we thus designed a simple greedy heuristic based on h^{add} [5], and used the solution obtained in this way as a MIP start for the solver. We also noticed that in some tasks the LP relaxation of the MIP model was significantly weaker than what could be obtained much faster by the well-known LM-cut heuristic [19], which is also based on disjunctive landmarks. We thus decided to warm-start each formulation with the set of landmarks computed by LM-cut itself.

Separating subtour elimination constraints (SEC) and/or disjunctive landmarks needs to be done in our dynamic generation approach each time an integer solution is encountered, in order to guarantee the correctness of the method. However, the current state of the art for many lazy constraint approaches, like TSP [2] and Benders [3], does not just separate constraints at integer solutions: to the contrary, it usually deploys a full branch-and-cut approach, where cutting planes are separated also at fractional nodes. In section 6, we thus investigate how to separate SEC and disjunctive landmarks on fractional solutions, showing that the former can still be done in polynomial time, while for the latter the complexity is still unknown. We provide nevertheless an exact approach based on a MIP formulation, and a polynomial separation heuristic.

In section 7 we provide an extensive computation evaluation of the proposed formulations, comparing them against the previous state of the art, and measuring the impact of the different enhancements, showing that some of the proposed methods significantly improve upon current state of the art. Conclusions and potential lines of future work are finally drawn in section 8.

2 Background and notation

A *STRIPS* [12] *planning task* is a 5-tuple $\Pi = (P, A, I, G, \text{cost})$: P is a finite set of boolean variables (or *facts*); A is a finite set of *actions*: each action $a \in A$ is a triple $(\text{pre}(a), \text{add}(a), \text{del}(a))$, where $\text{pre}(a), \text{add}(a), \text{del}(a) \subseteq P$ denote respectively the set of *preconditions*, *add effects* and *delete effects* of a; $I, G \subseteq P$ are, respectively, the *initial state* and *goal state*; $\text{cost} : A \to \mathbb{R}_0^+$ is a cost function that maps each action to a non-negative real number.

Executing an action a from a state $S \subseteq P$ transitions the system to a new state $S' = exec_a(S) = S \setminus \text{del}(a) \cup \text{add}(a)$. Let $\pi = (a_{\pi_1}, ..., a_{\pi_n})$ be a finite sequence of n actions and $S \subseteq P$ a state: we denote with $S(\pi) = exec_{a_{\pi_n}}(...(exec_{a_{\pi_1}}(S)))$, the state achieved through π, starting from S. A *solution* to a planning task is a sequence of actions that transitions the system from the initial state to a state $I(\pi) \supseteq G$. Moreover, a *feasible* solution, or *plan*, is a solution $\pi = (a_{\pi_1}, ..., a_{\pi_n})$ that satisfies $pre(a_{\pi_i}) \subseteq I((a_{\pi_1}, ..., a_{\pi_{i-1}}))$ for all $i \in \{1, \dots, n\}$, meaning that the preconditions of each action are met before it is executed. Finally, the cost of an action sequence π is $\text{cost}(\pi) = \sum_{a \in \pi} \text{cost}(a)$, and our goal is to find an *optimal plan* π^* with minimum cost c^*: $\pi^* = argmin_\pi\{\text{cost}(\pi) | \, \pi \text{ is a plan for } \Pi\}$.

In this paper, we always work with the *delete-relaxed task* $\Pi^+ = (P, A^+, I, G, cost)$, where $A^+ = \{(\text{pre}(a), \text{add}(a), \emptyset), \forall a \in A\}$ is obtained by replacing, from each action, the delete effects with the empty set. Without loss of generality, we can assume that $I = \emptyset$: in a delete-relaxed task, a fact that is true in the initial state remains true throughout the whole plan, rendering it irrelevant. If the delete-relaxed task does not have an empty initial state, we can obtain an equivalent task from any relevant set, without changing its optimal plan. Going forward, we assume the task Π to be delete-free and with an empty initial state, in order to simplify notation.

3 Static MIP formulations

We now define the basic MIP formulation for a delete-free task. At a bare minimum, the model needs to keep track of which actions are used in a plan: since in a delete-free task an action can be used at most once in a plan π, this naturally translates into a set of binary variables x_a, one for each $a \in A$, that take value 1 if and only if $a \in \pi$. In general, we also need to keep track of which facts become true, in order to enforce that the goal is reached and that the preconditions of actions are met, and this translates into another set of binary variables x_p, one for each $p \in P$, that take value 1 if and only if $p \in I(\pi)$. Finally, we need to keep track of which action is responsible for making a fact true for the first time, i.e., which action $a \in A$ is the first achiever of a fact $p \in P$, so we have variables $x_{a,p}$ that take value 1 if and only if a is the *first achiever* of p in π. Of course, $x_{a,p}$ is defined only for pairs (a, p) in which $p \in \text{add}(a)$.

With this initial set of variables, a first (incomplete) model, first introduced in [21] and then refined in [11], reads:

$$\min \sum_{a \in A} \text{cost}(a) x_a \tag{C0}$$

$$\sum_{a \in A_p} x_{a,p} = x_p \qquad \forall p \in P, A_p = \{a \in A \mid p \in \text{add}(a)\} \tag{C1}$$

$$\sum_{a \in A_{p,q}} x_{a,q} \leq x_p \qquad \forall p, q \in P,\ A_{p,q} = \{a \in A \mid\ p \in \text{pre}(a), q \in \text{add}(a)\} \tag{C2}$$

$$x_{a,p} \leq x_a \qquad \forall a \in A, p \in \text{add}(a) \tag{C3}$$

$$x_p = 1 \qquad \forall p \in G \tag{C4}$$

The objective function $(C0)$ just sums up the costs of the actions used. As for the constraints:

$(C1)$: a fact is achieved iff an action is its first achiever; moreover there can be at most one first achiever per fact;

$(C2)$: an action can be a first achiever only if its preconditions are achieved;

$(C3)$: an action can be a first achiever only if it is used;

$(C4)$: each fact in the goal state is achieved.

We denote with IP(Π) the solution obtained by the model described by $(C0) - (C4)$. As anticipated, unfortunately this model is incomplete, as it fails to forbid circular dependencies between actions: for example, we can select an action a with prerequisite p and add-effect q, together with an action b with prerequisite q and add-effect p, without any other selected action. In order to obtain a complete (and correct) model, we thus need to model causal *acyclicity* between the actions in the plan.

3.1 Modeling acyclicity

A general framework for reasoning about acyclicity is the so-called *causal relation graph* [11]. Given a delete-relaxed task Π, its causal relation graph is defined as the directed graph $G_\Pi = (P, E_\Pi)$, where an arc $(p, q) \in E_\Pi$ if and only if there is an action a with p as a precondition and q as an add-effect.

Any solution $\hat{x} = \text{IP}(\Pi)$ induces a subgraph $G_{\hat{x}} = (P, E_{\hat{x}})$ of G_Π, where we select only the edges (p, q) where q is first achieved by some action a, and thus $\hat{x}_{a,q} = 1$, and p are all the preconditions of such action. As shown in [11], for $\hat{x}$ to be associated to a feasible plan, $G_{\hat{x}}$ needs to be acyclic: a cycle in this graph constitutes a cyclical causal dependency between two or more facts, which means none of these facts can be applied before the others. The existing formulations in the literature differ in how cycles are forbidden in $G_{\hat{x}}$.

3.2 Time labeling

The first approach to deal with acyclicity is due to [21], and further refined in [11]. The basic idea is to simply assign a time label to each fact and ensuring that, whenever an action a is the first achiever of a fact q, the time stamp assigned to q must be greater by at least 1 of the timestamp of any precondition p of a. Timestamps are encoded with integer variables $t_p \in \{1, ..., |P|\}$ for any $p \in P$, and the model is completed with the constraints:

$$t_p - t_q + 1 \leq |P|(1 - x_{a,q}) \qquad \forall a \in A, p \in \text{pre}(a), q \in \text{add}(a) \tag{1}$$

We denote the model that uses Time Labeling as TL(Π). Note that in the constraints (1), the number of facts $|P|$ acts as a big-M coefficient: as such, we do not expect the model TL(Π) to have a strong LP relaxation. On the other hand, the model is quite compact, requiring only $|P|$ additional variables and $O(|A| \cdot |P|)$ additional constraints.

3.3 Vertex elimination

A completely different approach is based on vertex elimination graphs, a concept originally introduced by [27]. Given a directed graph $G = (P, E)$, let $O = p_1, ..., p_{|P|}$ be an arbitrary ordering of the vertices P. According to the ordering O, we construct a sequence of graphs $G_0 = G, ..., G_{|P|}$

by eliminating the vertices of G, where G_i is produced from G_{i-1} by eliminating p_i, and adding edges from all its in-neighbors to all its out-neighbors. Given an ordering O[1], we can apply the vertex elimination procedure to the causal relation graph G_Π, and obtain the graph $G^* = (P, E^*_\Pi)$; let Δ be the set of all ordered triplets (p_i, p_j, p_k) encoding the set of edges (p_i, p_k) added when eliminating p_j in the vertex elimination process. We can enforce $G_{\hat{x}}$ to be acyclic by adding the binary variables $e_{p,q} = \{0, 1\}$ for any edge $(p, q) \in E^*_\Pi$, together with the constraints:

$$\begin{aligned} x_{a,q} \leq e_{p,q} \quad & \forall a \in A, p \in \mathrm{pre}(a), q \in \mathrm{add}(a) \\ e_{p,q} + e_{q,p} \leq 1 \quad & \forall (p, q) \in E^*_\Pi \\ e_{p,q} + e_{q,r} - 1 \leq e_{p,r} \quad & \forall (p, q, r) \in \Delta \end{aligned}$$

We refer to [11] for the details and the proof of correctness of the resulting model, that we denote with VE(Π). The resulting model, while still polynomial, can be quite larger than TL(Π), as it requires $O(|P|^2)$ variables and $O(|P|^3)$ constraints. At the same time, its LP relaxation is provably stronger than the one of TL(Π).

4 Dynamic MIP formulations

Polynomial formulations, while appealing for their simplicity, are not necessarily the only approach for enforcing acyclicity in $G_{\hat{x}}$. Indeed, mathematical optimization has a decades-long experience in dealing with it in the context of the well-known TSP: numerous formulations have been proposed over the years, some of which polynomial, like those in [24, 14, 13, 8], but the most effective one, dating back to [10], yet still at the core of the state of the art TSP solver Concorde [9], is based on the so-called *subtour elimination constraint* (SEC) formulation and it has an exponential number of constraints. Of course, in this case we can no longer provide all the constraints upfront, but we need to be able to separate them dynamically during the solution process, in a branch-and-cut fashion [25]. All modern MIP solvers provide explicit support for this scenario, through the availability of *lazy constraint* callbacks: each time an integer solution is found, either by some primal heuristic or at an integer node, a user defined callback is invoked, to check whether the solution can be accepted as a new incumbent, or rejected. In the latter case, one of more inequalities (the nominal lazy constraints) violated by the integer point can be returned to the solver.

4.1 Subtour elimination constraints

Our first attempt at using a dynamic formulation is to exploit, with the due modifications, the class of subtour elimination constraints for delete-free planning as well. For any cycle C in the causal relation graph G_Π, we could impose the corresponding subtour elimination constraint $\sum_{(p,q)\in C} x_{p,q} \leq |C| - 1$, which forces the removal of (at least) one edge in the cycle. As anticipated, we cannot just add all those constraints upfront, as they are exponentially many. In addition, our basic model does *not* contain variables $x_{p,q}$, so we need to formulate the SEC in a different space. We now provide the details of the separation procedure: given a point $\hat{x}$, consider the corresponding causal graph $G_{\hat{x}}$, and denote with $\mathrm{act}_{\hat{x}}(p, q)$ the unique action $a \in A$ such that $\hat{x}_{a,q} = 1$. It is easy to check whether $G_{\hat{x}}$ is acyclic and, if not, for every cycle C detected, return the violated SEC inequality:

$$\sum_{(p,q)\in C} x_{\mathrm{act}_{\hat{x}}(p,q),q} \leq |C| - 1$$

[1] In practice, a minimum degree heuristic is used to compute O.

Note that in the implementation we do not compute all cycles in the graph, but we limit ourselves to disjoint cycles, i.e., cycles that do not share any edges: those cycles can be detected in linear time with a simple depth first search, keeping track of already visited edges. We denote the model that uses subtour elimination constraints as SEC(Π).

4.2 Disjunctive action landmarks

A completely different approach, that implicitly removes all cycles in $G_{\hat{x}}$, is based on the seemingly unrelated concept of *disjunctive action landmark* (landmark, for short). A *landmark* $L \subseteq A$ for Π is a set of actions such that every plan for Π must contain at least one action in L. A key result, due to [6], is that the computation of the optimal plan for a delete-relaxed task can be formulated as a minimum-cost hitting set problem over all its landmarks. In other words, given a complete set of landmarks F_L for a delete-free planning task, a valid formulation reads:

$$\min \sum_{a \in A} \text{cost}(a) x_a \tag{2}$$

$$\sum_{a \in L} x_a \geq 1 \qquad \forall L \in F_L \tag{3}$$

$$x_a \in \{0, 1\} \qquad \forall a \in A \tag{4}$$

Note that this formulation does not enforce acyclicity explicitly, and does not even require variables x_p and $x_{a,q}$. However, we can still use the landmark constraints (3) as cycle breaking constraints on top of our base model $(C0) - (C4)$, as those alone are enough to obtain a complete formulation. In addition, this is still a dynamic formulation, as the number of landmarks for a given planning task is in general exponential in the problem size.

We now turn to the questions on how to separate violated landmark constraints given an integer point $\hat{x}$. In the following, and without loss of generality, we will assume that:

- all actions have a non-empty set of preconditions;
- each action has a *single* add effect;
- there are two facts s and t such that $I = \{s\}$ and $G = \{t\}$.

If the original task does not satisfy these assumption, we can always construct an equivalent planning task that meet them via linear transformation. Note that those are different assumptions from the one done so far, i.e., that $I = \emptyset$; however the treatment of landmark constraints is greatly simplified in this new setting. We remark that this is only a conceptual simplification for ease of exposition: the actual implementation does not depend on it.

Following the treatment in [19, 6, 4], let us introduce *justification graphs*. A justification graph is a directed graph $G_D = (P, E_D)$, that has again the set of facts as nodes, and whose edges are defined by a so-called *precondition choice function* (pcf) D, i.e., a function $D : A \to P$ that associates to each action $a \in A$ one of its preconditions $p \in \text{pre}(a)$, so that we have an edge $(p, q) \in E_D$ labelled a if and only if there is an action a with $D(a) = p$ and $q = \text{add}(a)$. For a given justification graph G_D, an s-t-cut is a partition $(S, P \setminus S)$ such that $s \in S$ and $t \notin S$, and its cut-set is the set of edges (p, q) crossing the cut in the forward direction, i.e., with $p \in S$ and $q \notin S$. It is easy to show that, since any feasible plan defines a s-t-path in G_D, the labels of the edges in any cut-set define a valid landmark for Π. It was shown in [6] that the set of landmarks that can be obtained in this way, i.e., as cut-sets of justification graphs, is complete, meaning that they are sufficient to compute the value of h^+ via the hitting set formulation $(2) - (4)$.

This, by itself, does not immediately give a polynomial separation procedure, as both the number of pcfs D and the number of cut-sets of G_D for a given pcf are exponential in the problem size. However, as shown in [4], the completeness proof in [6] readily gives a linear separation procedure, even though it was not framed as a separation problem. Let H be the set of actions used in a solution $\hat{x}$, and $R(H) \subseteq P$ be the set of facts reachable from I using only actions in H. Then, if $G \not\subseteq R(H)$, a violated landmark can be obtained simply as:

$$L_H = \{a \in A \mid \text{pre}(a) \in R(H), \text{add}(a) \notin R(H)\} \tag{5}$$

This corresponds to constructing a pcf D by picking any $p \in \text{pre}(a)$ for the actions a such that $\text{pre}(a) \subseteq R(H)$, and $p \in \text{pre}(a) \setminus R(H)$ for the remaining ones. It is easy to show that H does not hit the s-t cut $(R(H), P \setminus R(H))$, and thus (5) gives a (maximally violated) landmark.

As observed in [17], the landmark obtained in this way is not necessarily minimal, so the corresponding cut could be dominated by other, equally violated, landmark constraints. In order to produce a minimal landmark, and thus a stronger cut, the same paper proposes a greedy heuristic that tries to obtain from H a maximal set of actions $H' \supseteq H$ such that the goal is still not reachable, i.e., $G \not\subseteq R(H')$: if H' is such a maximal set, then $A \setminus H'$ is a minimal landmark, still violated by $\hat{x}$. As noted in [17], with the appropriate data structures, the repeated reachability check implied by the greedy procedure outlined above can be done incrementally, and is quite efficient in practice. We denote the model that uses subtour elimination constraints as LMC(Π).

5 Warm-starting the MIP formulations

While experimenting with the MIP formulations described in the previous section, both old and new, we noticed a couple of common shortcomings. First, sometimes the MIP solver would struggle in finding a good feasible solution, or even just any feasible solution, early on in the solution process: this was common to all formulations, but even more severe in the dynamic formulations, where the incomplete nature of the model hinders the effectiveness of the primal heuristics implemented in the solver, a well-known side effect of dynamic constraint generation. Then, we noticed that in some cases the LP relaxation at the root node was significantly worse than the value that could be computed, much more efficiently, by the admissible heuristic LM-cut [19]. This is particularly surprising for formulation LMC(Π): since the LM-cut heuristic eventually just computes a clever set of landmark constraints, formulation LMC(Π) has always at least the potential, depending on how often landmark constraints are separated and how they are chosen, to dominate the bound from LM-cut. In this section we propose two techniques to fix those issues: a dedicated primal heuristic and the warmstarting of the model with LM-cut.

5.1 MIP start

Computing a plan for a delete-relaxed task is trivial, since a simple dive, choosing at each step any applicable action, either finds a feasible plan or proves the planning task to be infeasible. Based on this observation, we designed a simple greedy heuristic that incrementally constructs a feasible plan, to be provided as a *MIP start* to the MIP solver. The method starts with the initial state, and chooses, at each iteration, an applicable action maximizing a given score. After some preliminary experiments, we eventually settled on a lookahead strategy based on the h^{add} heuristic function [5]. For a given state S, we remind that the h^{add} value of a given fact p can be computed as the solution of the recursive relation:

$$h_S^{add}(p) = \begin{cases} 0 & \text{if } p \in S \\ \min_{a \in A_p}\{\text{cost}(a) + \sum_{q \in \text{pre}(a)}\{h_S^{add}(q)\}\} & \text{otherwise} \end{cases} \quad (6)$$

where $A_p = \{a \in A \mid p \in \text{add}(a)\}$. Those values can be computed in polynomial time by a generalized Dijkstra algorithm. Once $h_S^{add}(p)$ is computed for every $p \in P$, the h^{add} value of the state S itself is defined as $h_S^{add} = \sum_{p \in G} h_S^{add}(p)$. Thus, in the greedy construction heuristic, we choose at each step an action a, among the applicable ones, minimizing the quantity $h_{S \cup \text{add}(a)}^{add}$, i.e., minimizing the h^{add} of the state that we would reach if we applied that action.

5.2 LM-cut initialization

LM-cut [19] is a state of the art admissible heuristic for classical AI planning, based on the iterative computation of disjunctive landmarks: at each step of the process, a precondition choice function based on the $h^{\max}$ [5] values of the facts is chosen, an s-t cut-set is identified yielding a disjunctive landmark L, and the cost of actions in L is reduced by the minimum between their costs. The process is repeated until no cut-set of positive (reduced) cost can be found. We refer to [19, 22] for details about the procedure.

The value h^{LM} computed by LM-cut is valid for the delete-free relaxation of the a planning task, and thus dominated by h^+. Empirical studies have shown that h^{LM} is a very good approximation of h^+, often within a few percent. In other words, solving a hitting set problem over the set of landmarks computed by LM-cut is guaranteed to give at least h^{LM}, and thus any of the MIP formulations in this study can potentially benefit from those landmarks in terms of initial LP bound.

It is worth noting that the LM-cut procedure is not completely specified, as in general there are degrees of freedom in the choice of the pcf based on $h^{\max}$: for a given action a, we might have multiple preconditions achieving the maximum value. In addition, different pcfs can give, sometimes significantly, different lower bounds. In our implementation, we follow the approach described in [22], where LM-cut is run multiple times with different tie-breaking strategies, and collect all the produced landmarks; in particular, we execute LM-cut three times, using the ARB, INV and VDM policies in [22]. The runtime of LM-cut has always proved negligible w.r.t. the overall solution process.

6 Separating cuts on fractional solutions

In a modern branch-and-cut framework, it is very uncommon to separate lazy constraints only at integer solutions: it is, in general, more efficient to separate the same constraints also at fractional solutions, in particular at the root node, in order to improve the LP bound faster. This is a routine approach for both the TSP [9] and Benders decomposition [23], for example. Of course, given a family of inequalities, the separation procedure must be extended to deal with fractional solutions, and this sometimes becomes more expensive. For example, in the classical TSP case, separating SECs on integer solutions can be done in linear time with a simple graph search, while on fractional solutions this requires solving a sequence of max flow problems: still polynomial, but more expensive both in theory and in practice. In this section we describe separation procedures for separating SECs and landmarks over fractional solutions. We will see that the former can still be separated in polynomial time, while for the latter the complexity is still unknown: in this case we provide both an exact separation procedure based on a MIP formulation, and a polynomial heuristic separation procedure.

6.1 SEC on fractional solutions

Given a fractional solution x^* coming from the LP relaxation of any of our models, we can construct a directed weighted graph $G_{x^*} = (P, E_{x^*}, w)$, where we still have the facts as nodes, and an arc $(p, q) \in E_{x^*}$ if and only if there is an action a with p as a precondition, q as an add-effect, and $x_{a,p} > 0$; its label is $\text{act}_{x^*}(p, q) = \text{argmin}_a\{1 - x^*_{a,q} \mid a \in A, p \in \text{pre}(a), q = \text{add}(a)\}$ and its weight is $w_{p,q} = 1 - x_{\text{act}_{x^*}(p,q)}, q$.

We have a violated SEC if and only if there exists a cycle C in G_{x^*} in which the sum of the weights is strictly < 1. Indeed, a SEC constraint reads

$$\sum_{(p,q)\in C} x_{\text{act}_{x^*}(p,q)}, q \leq |C| - 1$$

but after complementing all the variables it can be rewritten into:

$$\sum_{(p,q)\in C} (1 - x_{\text{act}_{x^*}(p,q)}, q) = \sum_{(p,q)\in C} w_{p,q} \geq 1$$

In other words, the separation problem over fractional solutions can be casted as the problem of a minimum weighted cycle in a directed weighted graph. This can be computed in polynomial time by the following algorithm. For any arc (p, q) with $w_{p,q} < 1$, we thus compute the shortest path $S_{q,p}$ from q back to p in G_{x^*}, using the weights as costs in the shortest path computation: if $w_{p,q} + S_{q,p} < 1$, we have found a violated SEC. As in the integer case, in our implementation we only separate disjoint cycles.

6.2 Landmarks on fractional solutions

The separation of landmark constraints is conceptually made of two steps: the choice of a precondition choice function D, yielding a justification graph G_D, and the choice of an s-t cut. For a given justification graph G_D, the choice of the s-t cut giving the most violated landmark constraint can easily be encoded as a (polynomial) max-flow computation, using the values x^*_a as capacities on the edges with label a. This is because the assumption of each action having exactly one add effect implies that there are no duplicate action labels in any cut-set, and thus the capacity of the cut corresponds to 1 minus the violation of the landmark.

In the integer case, a simple reachability analysis is enough to define a pcf D giving a maximally violated landmark constraint. Our first step is thus to try to generalize the reachability computation to the fractional case. The most natural approach consists in casting the reachability analysis as the computation of recursive values, similar to h^{add} or h^{max}, based on the relations:

$$R_1(p) = \max_{a\in A|p=\text{add}(a)} R_1(a) \qquad \forall p \in P \tag{7}$$

$$R_1(a) = \min\{\min_{p\in\text{pre}(a)} R_1(p), x^*_a\} \qquad \forall a \in A \tag{8}$$

with the starting condition $R_1(s) = 1$. Those conditions can be interpreted as replacing the Boolean semiring defining reachability with a *fuzzy* counterpart, where max replaces OR and min replaces AND. For binary x^*_a, the resulting values of $R_1(p)$ encode exactly the reachability of each fact, and we can claim that there is no violated landmark if and only if $R_1(t) = 1$. Unfortunately, an analogous condition does not hold in the fractional case, as it is easy to construct planning tasks where there is no violated landmark even if $R_1(t) < 1$. A small example is given in fig. 1, where each edge corresponds to a different action, and for which the corresponding fractional value is 1/2.

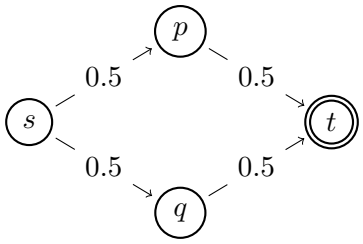

Figure 1: Example of $R_1(x^*) < 1$ but no violated landmark exists.

In the following, we will denote with $R_1(x^*)$ the value of $R_1(t)$ computed for a given fractional solution x^*.

We did not find a way to turn the fuzzy reachability values $R_1(p)$ into an exact separation procedure, akin to the one used in the integer case. However, we will show how to use them, together with some tie-breaking conditions, to devise a heuristic separation procedure.

6.2.1 Exact algorithm

In order to gain insights on the landmark separation process, and to empirically evaluate the effectiveness of potential heuristic procedures, we first formulated the exact separation of landmarks constraints as a MIP. This is unlikely to be an effective approach in practice, but it proved quite useful as a baseline. The MIP model is based on the definition of disjunctive landmarks from justification graphs. Let us introduce binary variables z_a encoding whether a given action a is part of the cut-set defining the landmark, and binary variables y_p encoding whether a fact p is on the s-side of the cut. The separation model reads:

$$\min \sum_{a \in A} x^*_a z_a \tag{9}$$

$$z_a \geq \sum_{p \in \text{pre}(a)} y_p - |\text{pre}(a)| + (1 - y_q) \qquad \forall a \in A, q = \text{add}(a) \tag{10}$$

$$y_s = 1 \tag{11}$$

$$y_t = 0 \tag{12}$$

The objective (9) minimizes the left-hand side of the landmark w.r.t. the solution to separate x^*, while constraints (10) impose that a can be part of the landmark only if all its preconditions are on the s side, but at least one of the effects is not. We are also imposing s to be on the s side (11), while forbidding t (12). In the following, we will denote with $\text{MC}(x^*)$ the optimal objective value of this separation MIP model.

6.2.2 Heuristic approach using max-flow

Even though the fuzzy $R_1(p)$ values seem not to be enough to devise an exact separation procedure, we devised a heuristic approach based on them. While computing R_1, we can at the same time also compute the relaxed values R_2, which are obtained with the similar relations:

$$R_2(p) = \min\{\sum_{a \in A | p = \text{add}(a)} R_2(a), 1\} \qquad \forall p \in P \tag{13}$$

$$R_2(a) = \min\{\min_{p \in \text{pre}(a)} R_2(p), x^*_a\} \qquad \forall a \in A \tag{14}$$

Clearly, we have $0 \leq R_1(p) \leq R_2(p) \leq 1$ for any fact $p \in P$, and analogously for actions. As usual, we denote with $R_2(x^*)$ the value of $R_2(t)$ computed for a given fractional solution x^*. Intuitively, we would expect the condition $R_1(x^*) \leq \text{MC}(x^*) \leq R_2(x^*)$ to always be satisfied, and, indeed, we have yet to find a counterexample for it. Even assuming it is true, however, note that we can still have $\text{MC}(x^*) < 1$ but $R_2(x^*) = 1$. A small example is given in fig. 2, where again we have a planning task where each edge corresponds to a different action, and for which the corresponding fractional value is 1/2.

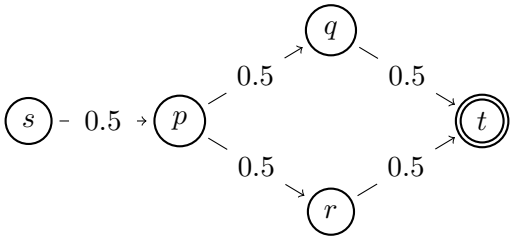

Figure 2: Example of $R_2(x^*) = 1$ but $\text{MC}(x^*) < 1$.

An important property of the values R_1 and R_2 is that, as with h^{max} values, we can always choose a pcf D that justifies them, i.e., such that the R_1 (resp. R_2) value of each fact/action is preserved in the simplified task: we just need to choose as precondition of a given action a a fact with minimum $R_1(p)$ (resp. $R_2(p)$). This property is at the basis of our proposed heuristic separation: after computing the values of R_1 and R_2, we choose a pcf D that justifies R_1, using R_2 as a tie-breaker, preferring lower values of R_2. We then identify a min-cut in the resulting justification graph G_D, via a max-flow computation: if the resulting cut has capacity < 1, then we have found a landmark constraint violated by x^*. Otherwise we fail to return a cut, even though one might exist. In our computation evaluation, this separation heuristic managed to find a violated inequality in $> 97\%$ of the cases in which one existed, as certified by the exact MIP separation, while being, on average, one order of magnitude faster.

7 Computational results

We implemented our approaches in C++, and evaluated them on the standard IPC benchmarks[2], for a total of 2799 planning tasks. The corresponding PDDL files were fed into the state of the art planner *FastDownward* [18], to obtain an equivalent SAS$^+$ description. All of our models start from this grounded SAS$^+$ representation. The resulting instances are then solved using a black box MIP solver, namely IBM ILOG CPLEX 22.1.0 [20], with default settings. The MIP solver was run on a cluster of 16 identical machines, each equipped with an Intel Xeon CPU E5-2623 V3 CPU running at 3.00 GHz, and 16 GB of RAM. Each method was run on each task with a time limit of 15 minutes. The source code is available as a single solver on GitHub[3]: the results are measured on version 2.4.2 of the solver.

Before constructing a MIP model, each planning task went through a preprocessing phase, consisting of landmark-based model reduction, relevance analysis and dominated actions elimination, as described in [21], as well as forward relevance analysis [17]. This preprocessing phase is common to all the formulations under test, and is well known to significantly speed up the solution of delete-free tasks when encoded as MIP formulations.

[2]https://github.com/AI-Planning/classical-domains
[3]https://github.com/Zanzibarr/MIPxHPLUS

In this section, the results are presented in tables comparing various models against a baseline. The baseline's absolute results appear on the left side of each table, while other methods are measured as ratios relative to the baseline (except for the number of instances solved, shown as deltas). Each row of the table corresponds to a subset of the instances, selected based on difficulty: in particular, the bracket $[l, u)$ collects all the instances where the smaller computing time among the methods under comparison is contained in the corresponding interval. Beyond the time brackets, three additional categories are shown: `all`, showing the results over all instances; `solvable`, selecting only the instances that could be solved to optimality by at least one method under comparison, and `all-solvable`, selecting only the instances that could be solved to optimality by all methods under comparison. Within each category, for time and nodes we report shifted geometric means [1], with a shift of 1.

For each comparison between the baseline and another method, we perform statistical tests on the results within each category to determine whether the difference is significant. We apply the Wilcoxon paired signed-rank test for nodes and times, using ratios of shifted values rather than differences to maintain consistency with the shifted geometric mean. For the number of instances solved, we use the McNemar's test on the binary success status. When a method shows statistically significant differences from the baseline in a given category, i.e., the test returns a p-value < 0.05, its results are displayed in bold.

7.1 Analysis of the current state of the art

In our first experiment, we compare the two known static models, namely TL(Π) and VE(Π), without any of the techniques proposed in this paper, in order to define a baseline for future comparisons. Aggregated results are given in table 1. As expected from the literature, VE(Π) clearly outperforms TL(Π), by solving 165 more instances; moreover, given the stronger LP relaxation, VE(Π) requires only a fraction of the nodes and it is on average almost 40% faster. The improvement is also quite consistent across different brackets. Going forward, we use VE(Π) as the baseline for future comparisons.

		TL(Π)			VE(Π)		
Category	Count	Solved	Nodes	Time (s)	Δ Solved	Nodes (rel.)	Time (rel.)
all	2799	2268	13.990	5.660	**+165**	**0.152**	**0.623**
solvable	2450	2268	9.093	2.364	**+165**	**0.131**	**0.513**
all-solvable	2251	2251	4.235	1.092	0	**0.198**	**0.686**
$[0, 1)$	1822	1812	1.863	0.295	**+10**	**0.133**	**0.483**
$[1, 10)$	354	292	180.612	18.515	**+62**	**0.021**	**0.213**
$[10, 100)$	188	129	583.999	115.768	**+49**	**0.025**	**0.261**
$[100, 900)$	134	35	401.762	712.435	**+44**	**0.074**	**0.418**

Table 1: Comparison between state of the art models TL(Π) and VE(Π).

7.2 Effectiveness of dynamic models

Next, we evaluate the effectiveness of dynamic MIP formulations, based on subtour elimination (SEC(Π), described in section 4.1) and landmark constraints (LMC(Π), described in section 4.2). As far as landmarks is concerned, we tested their separation both with and without the greedy strengthening procedure that makes the landmark minimal, and the former proved more effective,

using on 70% less nodes and $50 - 80\%$ less time on average. We thus enable this strengthening step in LMC(Π). We also tested the combined separation of both type of cuts, and we refer to this combination as (LM + SE)C(Π).

		VE(Π)			SEC(Π)			LMC(Π)		
Category	Count	Solved	Nodes	Time (s)	Δ Solved	Nodes (rel.)	Time (rel.)	Δ Solved	Nodes (rel.)	Time (rel.)
all	2799	2433	2.121	3.521	**-141**	**10.461**	**1.452**	+8	**3.173**	**0.936**
solvable	2517	2433	1.410	1.582	**-141**	**10.078**	**1.606**	+8	**2.820**	**0.882**
all-solvable	2217	2217	0.691	0.747	0	**6.637**	1.107	0	**2.628**	**0.766**
$[0, 1)$	1934	1934	0.450	0.248	**-33**	**8.174**	**2.000**	**-14**	**2.600**	**0.836**
$[1, 10)$	350	338	5.618	9.160	**-84**	**85.599**	**3.124**	-10	**7.116**	**0.801**
$[10, 100)$	156	130	16.675	69.634	**-29**	**48.309**	**1.561**	+9	**6.898**	0.795
$[100, 900)$	114	31	30.315	423.745	+5	**21.047**	**1.528**	**+23**	**4.559**	**1.257**

Table 2: Comparisons between static and dynamic models.

table 2 shows that using subtour elimination constraints is not competitive with the current state of the art, resulting in significantly fewer models solved to optimality and increased runtimes across the board; on the other hand, using landmarks constraints proves to be an improvement, reducing the computation time, with the only exception of the harder bracket. We note that the higher number of nodes explored is to be expected, given the fact that SEC(Π) and LMC(Π) are dynamic models, and that constraints are added on the fly. We also tested the use of SEC on top of LMC(Π), but it did not provide a significant benefit (table 3).

		LMC(Π)			(LM + SE)C(Π)		
Category	Count	Solved	Nodes	Time (s)	Δ Solved	Nodes (rel.)	Time (rel.)
all	2799	2444	6.424	3.214	+7	**1.160**	1.005
solvable	2490	2444	3.458	1.236	+7	1.069	1.008
all-solvable	2405	2405	2.565	0.865	0	1.004	1.045
$[0, 1)$	1902	1902	0.898	0.131	0	**0.860**	1.017
$[1, 10)$	327	323	26.199	4.685	0	1.212	1.243
$[10, 100)$	176	157	119.592	49.852	+9	1.503	**0.761**
$[100, 900)$	89	62	1046.660	415.373	-2	**4.721**	0.938

Table 3: Adding SEC to LMC(Π).

7.3 Warm-starting the MIP formulations

In this section we evaluate the effectiveness of the warm start techniques discussed in section 5.1 and section 5.2, namely the addition of a greedy primal heuristic to provide a MIP start to the solver and the addition of the landmark constraints generated by LM-cut as constraints to the formulation. For each of the formulations TL(Π), VE(Π), SEC(Π), LMC(Π), (LM + SE)C(Π), we consider the versions with the added MIP start, that we denote with h subscript, and then finally the versions where we also add the landmark constraints, that we denote with the h, LM subscript.

Both TL(Π) and VE(Π) benefit from the proposed warm start techniques, with a significant improvement on every aspect (tables 4 and 5). These results show a significant improvement while still using a ready-to-use MIP formulation, without the need of a dynamic approach based on lazy constraints. We also note that both improvements contribute to the overall speedup, consistently across formulations and brackets.

		$\mathrm{TL}(\Pi)$			$\mathrm{TL}_h(\Pi)$			$\mathrm{TL}_{h,\mathrm{LM}}(\Pi)$		
Category	Count	Solved	Nodes	Time (s)	Δ Solved	Nodes (rel.)	Time (rel.)	Δ Solved	Nodes (rel.)	Time (rel.)
all	2799	2268	13.844	5.647	**+42**	**0.705**	**0.835**	**+122**	**0.266**	**0.636**
solvable	2396	2268	6.849	1.953	**+42**	**0.619**	**0.756**	**+122**	**0.182**	**0.471**
all-solvable	2257	2257	3.983	1.082	0	**0.701**	**0.787**	0	**0.242**	**0.565**
$[0, 1)$	1871	1863	2.033	0.381	-1	**0.675**	**0.747**	**+8**	**0.138**	**0.315**
$[1, 10)$	312	270	96.709	15.390	**+14**	**0.446**	**0.546**	**+42**	**0.063**	**0.232**
$[10, 100)$	156	104	374.789	128.544	**+28**	**0.180**	**0.509**	**+52**	**0.029**	**0.239**
$[100, 900)$	59	31	7812.194	550.706	+1	1.109	0.876	**+20**	**0.612**	**0.681**

Table 4: Improvements on $\mathrm{TL}(\Pi)$.

		$\mathrm{VE}(\Pi)$			$\mathrm{VE}_h(\Pi)$			$\mathrm{VE}_{h,\mathrm{LM}}(\Pi)$		
Category	Count	Solved	Nodes	Time (s)	Δ Solved	Nodes (rel.)	Time (rel.)	Δ Solved	Nodes (rel.)	Time (rel.)
all	2799	2435	2.126	3.526	+14	**0.817**	**0.923**	**+61**	**0.389**	**0.793**
solvable	2505	2435	1.506	1.531	+14	**0.764**	**0.890**	**+61**	**0.308**	**0.704**
all-solvable	2415	2415	1.119	1.049	0	**0.791**	**0.907**	0	**0.347**	**0.758**
$[0, 1)$	1904	1904	0.604	0.210	0	**0.833**	**0.891**	0	**0.313**	**0.573**
$[1, 10)$	324	315	3.722	6.825	0	**0.600**	**0.802**	**+9**	**0.146**	**0.519**
$[10, 100)$	217	187	16.138	67.519	**+13**	**0.514**	**0.749**	**+29**	**0.150**	**0.503**
$[100, 900)$	225	29	9.229	392.234	+1	0.971	**0.902**	**+23**	**0.550**	**0.832**

Table 5: Improvements on $\mathrm{VE}(\Pi)$.

		$\mathrm{TL}_{h,\mathrm{LM}}(\Pi)$			$\mathrm{VE}_{h,\mathrm{LM}}(\Pi)$		
Category	Count	Solved	Nodes	Time (s)	Δ Solved	Nodes (rel.)	Time (rel.)
all	2799	2394	3.695	3.599	**+104**	**0.224**	**0.778**
solvable	2517	2394	2.541	1.557	**+104**	**0.176**	**0.714**
all-solvable	2375	2375	1.215	0.844	0	**0.250**	0.940
$[0, 1)$	1939	1931	0.764	0.220	**+8**	**0.222**	**0.665**
$[1, 10)$	334	285	20.793	11.339	**+44**	**0.031**	**0.422**
$[10, 100)$	180	146	34.848	60.264	**+26**	**0.068**	**0.633**
$[100, 900)$	235	32	73.069	629.846	**+26**	**0.114**	**0.477**

Table 6: Comparison between state of the art models with warm start.

Comparing the static methods directly, we see that warmstarting did not change the relative ranking between $\mathrm{TL}(\Pi)$ and $\mathrm{VE}(\Pi)$, with the vertex elimination approach still clearly superior, see table 6. We will thus use $\mathrm{VE}_{h,\mathrm{LM}}(\Pi)$ as the baseline for future comparisons.

		$\mathrm{SEC}(\Pi)$			$\mathrm{SEC}_h(\Pi)$			$\mathrm{SEC}_{h,\mathrm{LM}}(\Pi)$		
Category	Count	Solved	Nodes	Time (s)	Δ Solved	Nodes (rel.)	Time (rel.)	Δ Solved	Nodes (rel.)	Time (rel.)
all	2799	2290	22.211	5.154	**+21**	**0.742**	**0.917**	**+120**	**0.253**	**0.651**
solvable	2421	2290	9.659	1.854	**+21**	**0.698**	**0.877**	**+120**	**0.191**	**0.493**
all-solvable	2278	2278	5.318	0.997	0	**0.711**	**0.870**	0	**0.263**	**0.652**
$[0, 1)$	1911	1880	2.658	0.423	-1	**0.836**	**0.896**	**+31**	**0.146**	**0.253**
$[1, 10)$	306	271	169.606	13.576	**+7**	**0.448**	**0.757**	**+35**	**0.059**	**0.298**
$[10, 100)$	141	107	2290.830	97.085	**+14**	**0.139**	**0.652**	**+33**	**0.037**	**0.385**
$[100, 900)$	67	32	13482.374	575.280	+1	**0.821**	0.943	**+21**	**0.506**	**0.633**

Table 7: Improvements on $\mathrm{SEC}(\Pi)$.

		LMC(Π)			LMC$_h$(Π)			LMC$_{h,\mathrm{LM}}$(Π)		
Category	Count	Solved	Nodes	Time (s)	Δ Solved	Nodes (rel.)	Time (rel.)	Δ Solved	Nodes (rel.)	Time (rel.)
all	2799	2442	6.660	3.288	**+22**	**0.732**	**0.923**	**+73**	**0.328**	**0.697**
solvable	2531	2442	3.871	1.465	**+22**	**0.684**	**0.891**	**+73**	**0.245**	**0.575**
all-solvable	2417	2417	2.621	0.894	0	**0.666**	**0.899**	0	**0.273**	**0.677**
$[0, 1)$	2014	1992	1.475	0.308	-5	**0.699**	**0.936**	**+22**	**0.175**	**0.353**
$[1, 10)$	328	315	30.438	9.845	**+7**	**0.626**	**0.761**	**+13**	**0.135**	**0.386**
$[10, 100)$	124	96	143.679	93.496	**+13**	**0.190**	**0.591**	**+26**	**0.050**	**0.367**
$[100, 900)$	262	39	800.856	441.708	+7	1.288	1.097	+12	0.925	0.796

Table 8: Improvements on LMC(Π).

In tables 7 and 8 we show the improvements on the dynamic models as well. Again, both techniques give consistent improvements. Lastly, we reevaluate our dynamic models against the baseline taking warm start techniques into account: while SEC$_{h,\mathrm{LM}}$(Π) is still not competitive with our baseline, LMC$_{h,\mathrm{LM}}$(Π) further widens the gap with VE$_{h,\mathrm{LM}}$(Π) by solving 22 more instances and reducing the computation time below 70% (table 9).

		VE$_{h,\mathrm{LM}}$(Π)			SEC$_{h,\mathrm{LM}}$(Π)			LMC$_{h,\mathrm{LM}}$(Π)		
Category	Count	Solved	Nodes	Time (s)	Δ Solved	Nodes (rel.)	Time (rel.)	Δ Solved	Nodes (rel.)	Time (rel.)
all	2799	2498	0.826	2.804	**-84**	**6.871**	1.194	**+22**	**2.687**	**0.824**
solvable	2563	2498	0.502	1.344	**-84**	**7.528**	**1.239**	**+22**	**2.388**	**0.729**
all-solvable	2357	2357	0.270	0.792	0	**5.266**	**0.893**	0	**1.953**	**0.690**
$[0, 1)$	2017	2017	0.238	0.211	**-29**	**6.217**	1.648	**-11**	**1.635**	**0.722**
$[1, 10)$	339	326	0.502	9.257	**-48**	**56.004**	1.520	+9	**8.280**	**0.421**
$[10, 100)$	138	120	3.951	70.187	-4	**11.928**	**0.805**	**+11**	**4.113**	**0.583**
$[100, 900)$	200	35	10.397	363.968	-3	**21.489**	**1.725**	+13	**8.445**	**1.399**

Table 9: Comparison between static and dynamic models with warmstarting.

We also compare LMC$_{h,\mathrm{LM}}$(Π) with (LM + SE)C$_{h,\mathrm{LM}}$(Π), to see whether the improvements made by the warm start techniques help the hybrid method, but as shown in table 10, the use of SEC is still not justified by any significant improvement. Going forward, we will thus use LMC$_{h,\mathrm{LM}}$(Π) as the baseline for future comparisons.

		LMC$_{h,\mathrm{LM}}$(Π)			(LM + SE)C$_{h,\mathrm{LM}}$(Π)		
Category	Count	Solved	Nodes	Time (s)	Δ Solved	Nodes (rel.)	Time (rel.)
all	2799	2520	2.220	2.310	+7	**1.195**	**1.029**
solvable	2546	2520	1.073	0.900	+7	**1.255**	**1.046**
all-solvable	2501	2501	0.779	0.725	0	**1.280**	**1.090**
$[0, 1)$	2004	2002	0.271	0.111	+2	**1.299**	**1.044**
$[1, 10)$	333	330	4.264	3.626	+1	**1.742**	**1.430**
$[10, 100)$	147	137	21.465	42.407	+7	0.977	**0.689**
$[100, 900)$	62	51	354.147	325.437	-3	**1.871**	1.052

Table 10: Adding SEC to LMC$_{h,\mathrm{LM}}$(Π).

7.4 Separating cuts on fractional solutions

In this section we enable the separation of violated constraints on fractional solutions, to see whether those can provide further improvements. In preliminary experiments, we found that, if implemented naïvely, the root cutloop with the separation of cuts at fractional solution could become way too expensive, with the vast majority of the solution time spent in adding cuts with a very small effect

on the LP bound. Due to this fact, and to have full control on the cutloop strategy, we implemented our own root cutloop, to be executed before solving the problem as a MIP with CPLEX. Our cutloop terminates when no violated cut has been found in the last iteration, if the relative MIP gap drops below 10% (w.r.t. the primal bound obtained by the solution constructed by our greedy heuristic) or if the improvement on the lower bound, with respect with the last 10 iterations, is below 0.5%. Other than at the root node, we separate cuts on fractional nodes as well, but only once per node. We evaluate the following models:

- fSEC(Π): $\text{LMC}_{h,\text{LM}}(\Pi)$ with violated subtour elimination constraints, as described in section 6.1;
- $\text{fLM}_{\text{MIP}}(\Pi)$: $\text{LMC}_{h,\text{LM}}(\Pi)$ with violated landmark constraints, computed with the exact approach described in section 6.2.1;
- fLM(Π): $\text{LMC}_{h,\text{LM}}(\Pi)$ with violated landmark constraints, computed with the heuristic approach described in section 6.2.2;
- fLS(Π): combination of fLM(Π) and fSEC(Π).

		$\text{LMC}_{h,\text{LM}}(\Pi)$			$\text{fLM}_{\text{MIP}}(\Pi)$		
Category	Count	Solved	Nodes	Time (s)	Δ Solved	Nodes (rel.)	Time (rel.)
all	2799	2519	2.213	2.306	-2	**0.661**	**1.069**
solvable	2541	2519	1.026	0.879	-2	**0.716**	**1.113**
all-solvable	2495	2495	0.750	0.701	0	**0.726**	**1.142**
$[0, 1)$	1998	1998	0.242	0.102	0	**0.735**	**1.309**
$[1, 10)$	342	339	4.801	3.680	+3	**0.612**	**1.178**
$[10, 100)$	132	124	13.166	38.143	+3	**0.511**	1.047
$[100, 900)$	69	58	378.944	312.727	-8	**0.327**	**1.410**

Table 11: Use of the MIP LM-based fractional separator.

As shown in table 11, separating violated landmark constraints on fractional solutions manages to reduce the number of nodes explored; however, the total computation time is not yet competitive with our best model.

		$\text{fLM}_{\text{MIP}}(\Pi)$			fLM(Π)		
Category	Count	Solved	Nodes	Time (s)	Δ Solved	Nodes (rel.)	Time (rel.)
all	2799	2516	1.459	2.464	+6	**1.201**	**0.961**
solvable	2537	2516	0.722	0.963	+6	**1.229**	**0.938**
all-solvable	2501	2501	0.583	0.820	0	**1.219**	**0.933**
$[0, 1)$	1969	1969	0.155	0.117	0	**1.264**	**0.837**
$[1, 10)$	349	349	2.941	3.701	0	**1.273**	**0.933**
$[10, 100)$	156	151	7.369	37.884	-3	**1.688**	0.929
$[100, 900)$	63	47	89.542	434.676	+9	1.497	**0.796**

Table 12: MIP vs polynomial LM-based fractional separator.

The heuristic separator manages to reduce the computation time (table 12), at the expense of an increased number of nodes, due to the fact that it can fail to find a violated cut when one exists.

Unfortunately, when comparing it with the baseline, it is still not competitive enough time-wise, see table 13.

		$\text{LMC}_{h,\text{LM}}(\Pi)$			fLM(Π)		
Category	Count	Solved	Nodes	Time (s)	Δ Solved	Nodes (rel.)	Time (rel.)
all	2799	2518	2.209	2.306	+4	**0.793**	**1.027**
solvable	2540	2518	1.005	0.879	+4	**0.868**	**1.045**
all-solvable	2500	2500	0.787	0.717	0	**0.896**	**1.071**
$[0, 1)$	2005	2005	0.257	0.105	0	0.866	**1.094**
$[1, 10)$	331	328	4.372	3.725	+3	0.883	**1.092**
$[10, 100)$	135	131	12.114	35.398	+1	**0.748**	1.009
$[100, 900)$	256	54	349.565	341.793	0	**0.489**	1.163

Table 13: Use of the polynomial LM-based fractional separator.

Finally, we evaluate the use of the SEC-based fractional-separator on our baseline but, once again, SEC do not seem to help significantly (table 14).

		$\text{LMC}_{h,\text{LM}}(\Pi)$			fSEC(Π)			fLS(Π)		
Category	Count	Solved	Nodes	Time (s)	Δ Solved	Nodes (rel.)	Time (rel.)	Δ Solved	Nodes (rel.)	Time (rel.)
all	2799	2519	2.210	2.308	**+16**	0.992	**1.049**	+12	**0.692**	**1.071**
solvable	2561	2519	1.170	0.973	**+16**	1.013	**1.076**	+12	**0.755**	**1.110**
all-solvable	2485	2485	0.708	0.695	0	1.009	**1.104**	0	**0.758**	**1.142**
$[0, 1)$	2009	2008	0.282	0.110	+1	1.101	**1.107**	+1	**0.695**	**1.096**
$[1, 10)$	338	333	5.111	4.050	-3	0.968	**1.461**	-2	0.807	**1.628**
$[10, 100)$	145	129	19.944	50.938	**+11**	0.940	0.793	**+12**	**0.523**	0.802
$[100, 900)$	69	49	530.190	379.259	+7	0.878	0.994	+1	**0.329**	1.155

Table 14: Use of the SEC-based fractional separator.

7.5 Overall improvement

In this section, we measure the impact of all the methods proposed, comparing our best method $\text{LMC}_{h,\text{LM}}(\Pi)$ with the previous state of the art VE(Π). Aggregated results are given in table 15: overall, with the proposed model and warm-starting techniques, we manage to reduce to 70% the average number of nodes used to find the optimal solution, and to reduce the overall computing time by approximately by 35%, and, more importantly, we can solve more instances to optimality within the time limit.

In table 16, we show the comparison divided by problem domain: according to the table, our model outperforms the state of the art for most domains, but there are still some domains, like *barman, elevators, rovers*, in which the new method is not competitive with VE(Π). There are also domains, like *petri* and *thoughtful*, in which, even though we manage to beat the state of the art time-wise, we are exploring a much larger number of nodes, which might suggest that either the LP relaxation is still too weak or that our MIP start is not a good primal heuristic in those domains.

Finally, in fig. 4, we provide a survival plot comparing the state of the art VE(Π) against its warm-started counterpart $\text{VE}_{h,\text{LM}}(\Pi)$ and our dynamic methods $\text{SEC}_{h,\text{LM}}(\Pi)$ and $\text{LMC}_{h,\text{LM}}(\Pi)$. The plot confirms that even just warm-starting VE(Π) has a positive effect, consistently increasing the number of instances that can be solved to optimality within each time limit, and the same applies in turn to our method $\text{LMC}_{h,\text{LM}}(\Pi)$. $\text{SEC}_{h,\text{LM}}(\Pi)$ can be considered somewhat competitive on smaller time limits, but it becomes less and less competitive as the instances become more difficult to solve.

		VE(Π)			$\mathrm{LMC}_{h,\mathrm{LM}}(\Pi)$		
Category	Count	Solved	Nodes	Time (s)	Δ Solved	Nodes (rel.)	Time (rel.)
all	2799	2435	2.126	3.526	**+81**	1.045	**0.653**
solvable	2549	2435	1.510	1.773	**+81**	**0.747**	**0.523**
all-solvable	2402	2402	1.057	1.076	0	**0.736**	**0.501**
$[0, 1)$	2017	2013	0.704	0.342	-8	**0.548**	**0.459**
$[1, 10)$	334	306	4.460	17.966	**+26**	0.813	**0.205**
$[10, 100)$	143	96	30.462	111.187	**+37**	0.748	**0.396**
$[100, 900)$	208	20	8.981	401.510	**+26**	**4.724**	1.212

Table 15: Comparison between state of the art and proposed model.

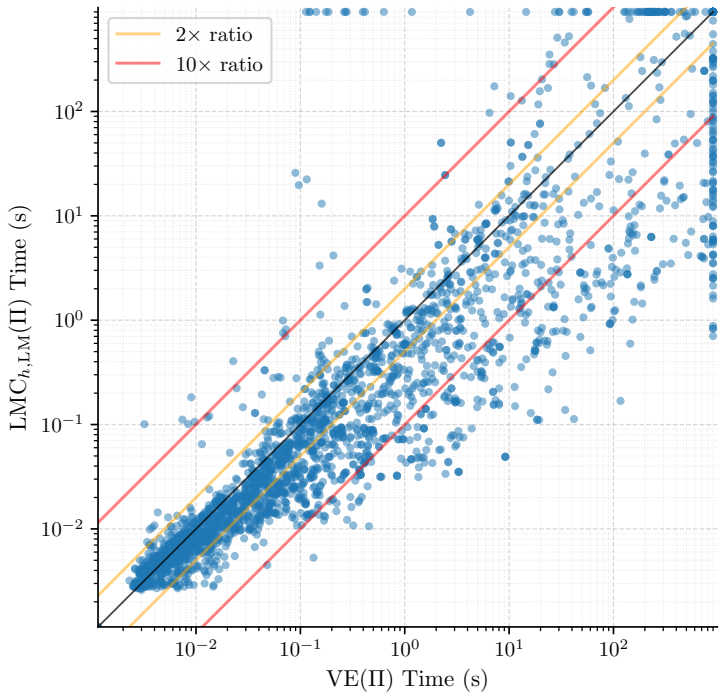

Figure 3: Scatter plot comparing the state of the art model VE(Π) against the new model $\mathrm{LMC}_{h,\mathrm{LM}}(\Pi)$.

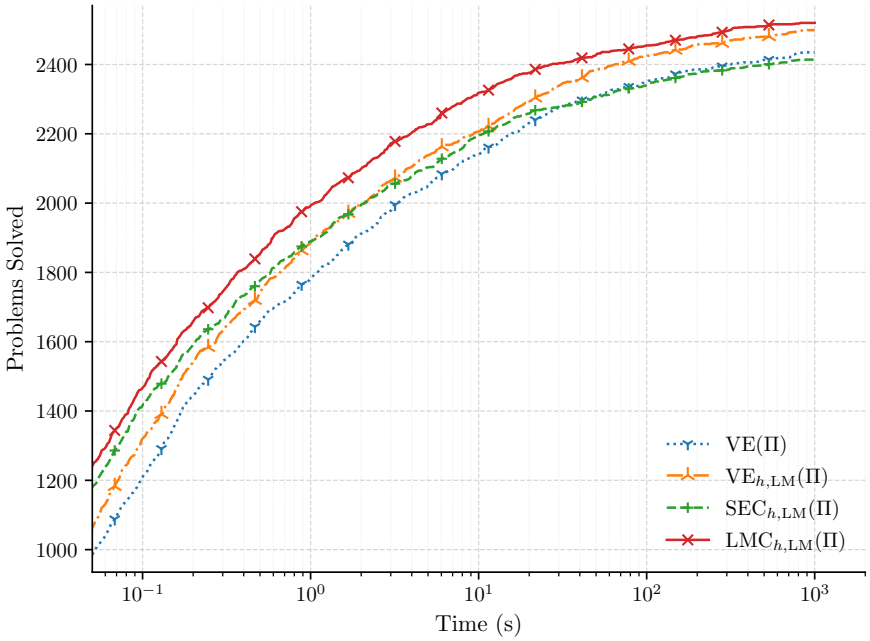

Figure 4: Cumulative number of instances solved by every model.

		VE(Π)			$\text{LMC}_{h,\text{LM}}(\Pi)$		
Domain	Count	Solved	Nodes	Time (s)	Δ Solved	Nodes (rel.)	Time (rel.)
agricola	40	40	41.286	10.763	0	**0.033**	**0.213**
airport	50	50	3.429	0.395	0	**0.000**	**0.652**
barman	94	94	7.073	0.450	**-13**	1.361	4.744
blocks	35	35	0.000	0.006	0	1.000	0.900
childsnack	40	40	0.000	0.078	0	1.000	0.735
data	40	29	177.039	44.519	**+7**	**0.170**	**0.215**
depot	22	22	0.468	0.654	0	0.495	**0.337**
driverlog	20	19	6.497	2.842	0	0.738	0.874
elevators	100	100	0.452	0.855	-1	**128.460**	**2.430**
floortile	80	80	3.141	0.300	0	**0.014**	**0.324**
freecell	80	80	0.047	3.142	0	**65.547**	**0.390**
ged	40	40	0.000	3.293	0	1.000	1.077
grid	5	5	40.400	0.374	0	0.035	0.349
gripper	20	20	0.000	0.010	0	1.000	0.970
hiking	60	60	0.000	0.540	0	1.000	**0.875**
logistics00	28	28	0.000	0.009	0	1.000	0.672
logistics98	35	31	4.769	4.274	+4	**0.000**	**0.161**
miconic	150	150	0.000	0.012	0	1.000	0.699
movie	30	30	0.000	0.004	0	1.000	0.680
mprime	35	33	6.369	6.204	-1	**0.228**	**0.395**
mystery	30	30	2.198	1.763	-1	0.170	**0.511**
nomystery	40	40	0.000	2.638	0	1.000	**0.158**
openstacks	170	169	0.000	0.733	0	1.000	**0.889**
organic	52	27	0.619	45.339	**+7**	1.075	0.953
parcprinter	70	70	0.000	0.022	0	1.000	0.741
parking	80	80	0.064	1.587	0	**49.099**	**1.213**
pathways	30	30	0.000	0.130	0	1.000	1.155
pegsol	70	70	38.590	0.598	0	**0.011**	**0.052**
petri	20	20	0.149	9.982	0	**576.100**	**0.128**
pipesworld	100	47	6.565	79.699	+4	**18.597**	0.901
psr	50	50	0.000	0.005	0	1.000	0.815
rovers	40	40	0.000	0.170	0	1.000	**1.637**
satellite	36	36	0.000	1.770	0	1.000	**1.097**
scanalyzer	70	57	0.265	28.427	+5	**12.724**	**0.209**
schedule	150	149	0.000	0.008	0	1.000	0.798
snake	40	8	1.970	432.741	**+8**	**5.585**	0.375
sokoban	100	100	0.365	0.241	0	0.000	**0.125**
spider	40	18	5.529	143.843	**+20**	0.876	**0.107**
storage	30	30	0.000	0.361	0	1.000	**0.716**
termes	40	40	0.149	0.058	0	2.057	1.068
tetris	37	27	0.000	45.673	**+10**	1.000	**0.052**
thoughtful	40	5	0.653	553.137	0	**548.092**	0.793
tidybot	60	13	187.651	429.444	**+21**	**0.210**	**0.146**
tpp	30	30	3.053	1.585	0	**0.000**	**0.187**
transport	140	54	406.793	129.364	**-10**	**0.617**	0.971
trucks	30	30	0.000	0.100	0	1.000	0.833
visitall	80	59	35.402	18.907	**+21**	**0.008**	**0.144**
woodworking	100	100	0.000	0.280	0	1.000	0.904
zenotravel	20	20	0.177	0.538	0	0.000	**0.327**

Table 16: Comparison between state of the art and proposed model, divided by domain.

8 Conclusions and future work

We analyzed existing MIP formulations for computing h^+, which consists of a base model plus a set of constraints required to model acyclicity in the causal relation graph associated to the task Π, using timestamps and vertex elimination graphs. We then introduced different MIP formulations, which require an exponential number of constraints, and thus require the dynamic separation of violated constraints during the solution process: one formulation is based on SECs, while the other is based on disjunctive action landmarks. While the former did not prove competitive with the other methods, the latter managed to outperform the current state of the art. In addition, we proposed two warm-starting techniques, applicable to all MIP formulations, one providing a primal heuristic solution as a MIP start and the other initializing the set of constraints with the set of landmarks computed using the LM-cut heuristic. The warm-starting techniques proved effective in improving both the primal and dual behaviour of the MIP solution process, consistently across formulations. Finally, we tried separating cuts on fractional solutions, deploying a full branch-and-cut approach: unfortunately, with the current separation methods, we did not manage to obtain improvements in computing time, despite an encouraging reduction in the number of enumeration nodes. Combining the warm-starting techniques and the dynamic model using landmarks, we managed to significantly improve over the current state of the art, reducing both the number of nodes and the time needed to compute h^+ with a MIP approach. (tables 15 and 16 and figs. 3 and 4).

The present work leaves some interesting lines for future research: first of all, it would be interesting to assess the theoretical complexity of disjunctive landmark separation over fractional solutions. Finally, we found quite surprising that cutting planes separated from fractional solutions did not prove effective in speeding up the solution process, despite being relatively inexpensive to separate: even though preliminary experiments with more sophisticated separation strategies based on in-out also proved ineffective, we think that other approaches, in particular improvements in the selection of violated cuts and a better tuning of the separation frequency, will eventually lead to an improved computation of h^+.

9 Acknowledgments

We would like to express our sincere gratitude to Florian Pommerening, Malte Helmert, and Gabriele Röger for the many discussions (and patient explanations) on AI planning.

References

[1] Tobias Achterberg. *Constraint Integer Programming.* PhD thesis, Technische Universität Berlin, 2007.

[2] David L. Applegate, Robert E. Bixby, Vasek Chvátal, and William Cook. *The Traveling Salesman Problem: A Computational Study.* Princeton University Press, 2007.

[3] Pierre Bonami, Domenico Salvagnin, and Andrea Tramontani. Implementing automatic benders decomposition in a modern mip solver. In Springer, editor, *IPCO 2020 Proceedings*, pages 78–90, 2020.

[4] Blai Bonet and Julio Castillo. A complete algorithm for generating landmarks. In *Proceedings of the International Conference on Automated Planning and Scheduling.* AAAI Press, 2011.

[5] Blai Bonet and Héctor Geffner. Planning as heuristic search. *Artificial Intelligence*, 129, 2001.

[6] Blai Bonet and Malte Helmert. Strengthening landmark heuristics via hitting sets. In Helder Coelho, Rudi Studer, and Michael J. Wooldridge, editors, *ECAI*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 329–334. IOS Press, 2010.

[7] Tom Bylander. The computational complexity of propositional strips planning. *Artificial Intelligence*, 69, 1994.

[8] A Claus. A new formulation for the travelling salesman problem. *SIAM Journal on Algebraic Discrete Methods*, 5(1):21–25, 1984.

[9] William Cook. Concorde TSP solver, 2024. `https://www.math.uwaterloo.ca/tsp/concorde.html`, Last accessed on June, 2024.

[10] G. Dantzig, R. Fulkerson, and S. Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America*, 2(4):393–410, 1954.

[11] Masood Feyzbakhsh Rankooh and Jussi Rintanen. Efficient computation and informative estimation of h+ by integer and linear programming. *Proceedings of the International Conference on Automated Planning and Scheduling*, 32, 2022.

[12] Richard E. Fikes and Nils J. Nilsson. Strips: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[13] Kenneth R Fox, Bezalel Gavish, and Stephen C Graves. An n-constraint formulation of the (time-dependent) traveling salesman problem. *Operations Research*, 28(4):1018–1021, 1980.

[14] Bezalel Gavish and Stephen C Graves. The travelling salesman problem and related problems. *Massachusetts Institute of Technology, Operations Research Center Working Papers*, 1978.

[15] Avitan Gefen and Ronen Brafman. The minimal seed set problem. *Proceedings of the International Conference on Automated Planning and Scheduling*, 21, 2011.

[16] Patrik Haslum. Incremental lower bounds for additive cost planning problems. *Proceedings of the International Conference on Automated Planning and Scheduling*, 22, 2012.

[17] Patrik Haslum, John Slaney, and Sylvie Thiebaux. Minimal landmarks for optimal delete-free planning. *ICAPS 2012 - Proceedings of the 22nd International Conference on Automated Planning and Scheduling*, 22, 2012.

[18] Malte Helmert. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

[19] Malte Helmert and Carmel Domshlak. Landmarks, critical paths and abstractions: what's the difference anyway? In *Proceedings of the International Conference on Automated Planning and Scheduling*, 2009.

[20] IBM ILOG. CPLEX optimization studio 22.1. `https://www.ibm.com/products/ilog-cplex-optimization-studio`, 2022.

[21] Tatsuya Imai and Alex Fukunaga. On a practical, integer-linear programming model for delete-free tasks and its use as a heuristic for cost-optimal planning. *Journal of Artificial Intelligence Research*, 54, 2015.

[22] Pascal Lauer and Maximiliam Fickert. Beating LM-cut with LM-cut: Quick cutting and practical tie breaking for the precondition choice function. In *ICAPS 2020 Workshop on Heuristics and Search for Domain-independent Planning*, 2020.

[23] D. McDaniel and M. Devine. A modified Benders' partitioning algorithm for Mixed Integer Programming. *Management Science*, 4:312–319, 1977.

[24] Clair E Miller, Albert W Tucker, and Richard A Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7(4):326–329, 1960.

[25] Manfred Padberg and Giovanni Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33(1):60–100, 1991.

[26] Nathan Robinson, Sheila McIlraith, and David Toman. Cost-based query optimization via ai planning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 28, 2014.

[27] Donald J. Rose, R. Endre Tarjan, and George S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on Computing*, 5(2):266–283, 1976.