Proceedings of the Thirty-Sixth International Conference on Automated Planning and Scheduling (ICAPS 2026)

Finding Optimal Cost-Bounded Plan Reductions

Martha Del Toro, Raquel Fuentetaja, Angel García-Olaya

Computer Science and Engineering Department, Universidad Carlos III de Madrid, Spain
mdeltoro@pa.uc3m.es, {rfuentet, agolaya}@inf.uc3m.es

Abstract

In some real applications a plan may later become unfeasible due to newly imposed budget constraints, yet, at the same time, using only the original actions of the plan and their order is mandatory. In this paper, we study the problem of extracting, from a precomputed plan, a valid subplan that maximizes utility while respecting a cost bound. Each goal is given a utility value and the plan is reduced by removing actions that support low-utility goals, while preserving both executability and the original action order. We show the decision variant is NP-complete and propose two exact methods to solve it: one via oversubscription planning (OSP) and another via Integer Linear Programming (ILP). Experiments show that ILP outperforms OSP, providing a practical solution in domains where maintaining the structure of the original plan is essential.

1 Introduction

Automated planning systems generate plans that are executed under resource constraints that can change dynamically. When a cost limit is imposed, the original plan may become infeasible, and some goals may need to be dropped to satisfy the new cost constraints. Keeping a subplan rather than replanning is essential in applications where the original plan has already been externally validated or relies on pre-committed resources. In such cases, the overhead of approving a new plan can be high, and full reorganization is often too costly or too slow. For instance, in space operations, command sequences must meet strict safety standards, and operators may prefer to skip specific tasks rather than upload an unverified sequence that could risk the mission. Similarly, in industrial workflows, operators follow established procedures and fixed schedules, making task omission simpler than reorganizing the entire workflow. With that in mind, we study the problem of extracting a valid subplan from a precomputed plan that maximizes utility while respecting a cost bound. This problem is closely related to *oversubscription planning* (OSP) (Smith 2004; Domshlak and Mirkis 2015; Katz et al. 2019; García-Olaya, de la Rosa, and Borrajo 2021; Speck and Katz 2021; Katz and Keyder 2022), with the difference that we operate on a precomputed plan and allow only action deletion, preserving the plan structure.

It also differs from traditional plan-repair techniques (Fox et al. 2006), which typically aim to restore all original goals after unexpected changes in the environment; in our case, some goals are deliberately dropped to satisfy a new budget.

We show that even the decision version of the problem is NP-complete. To solve it, we build on techniques for identifying and removing redundant actions from plans (Fink and Yang 1992; Nakhost and Müller 2010; Chrpa, McCluskey, and Osborne 2012a,b; Balyo, Chrpa, and Kilani 2014; Salerno, Fuentetaja, and Seipp 2025), specifically on the one introduced by Salerno, Fuentetaja, and Seipp (2025), and propose two new exact methods: one based on compiling the problem into an OSP instance, and another based on formulating it as an Integer Linear Program (ILP).

Our empirical evaluation shows that the ILP approach outperforms OSP in optimally solving cost-bounded plan reduction problems across diverse domains.

2 Background

We consider classical planning tasks with action costs in the SAS$^+$formalism (Bäckström and Nebel 1995).

Definition 1 (**Planning task**). *A planning task is a tuple* $\Pi = \langle \mathcal{V}, \mathcal{A}, \mathcal{I}, \mathcal{G}, c \rangle$*, where:*

- $\mathcal{V}$ *is a set of* state variables*, each with a finite* domain $\mathcal{D}(v)$*. A pair* $\langle v, d \rangle$ *such that* $v \in V$ *and* $d \in \mathcal{D}(v)$ *is a* fact*, denoted as* $v \mapsto d$*.* F *is the set of all facts. A* partial state s *maps a subset of variables* $vars(s) \subseteq \mathcal{V}$ *to values in their domains. We often interpret partial states as sets of facts. The value of variable* $v \in vars(s)$ *in partial state* s *is denoted as* $s[v] \in \mathcal{D}(v)$*. A partial state that assigns values to all variables* ($vars(s) = \mathcal{V}$) *is a* state.
- $\mathcal{A}$ *is a finite set of actions. Each* action $a \in \mathcal{A}$ *is a pair* $\langle pre(a), \mathit{eff}(a) \rangle$*, where* $pre(a)$ *and* $\mathit{eff}(a)$ *are both partial states defining the* precondition *and the* effect *of* a*, respectively. An action* $a \in \mathcal{A}$ *is applicable in state* s *iff* $pre(a) \subseteq s$*. Applying action* a *in* s *yields the successor state* $s[\![a]\!]$*, with* $s[\![a]\!][v] = \mathit{eff}(a)[v]$ *if* $v \in vars(\mathit{eff}(a))$ *and as* $s[\![a]\!][v] = s[v]$ *otherwise.*
- $\mathcal{I}$ *is the* initial state.
- $\mathcal{G}$ *is a partial state describing the* goal condition*.*
- c *is a non-negative* cost function *defining the cost of each action* $c : \mathcal{A} \to \mathbb{N}_0$.

A *solution* or *plan* for Π is an action sequence $\pi = \langle a_1, \ldots, a_n \rangle$ that, when applied in succession starting from the initial state, induces a state sequence $\mathcal{S}_\pi = \langle s_0, \ldots, s_n \rangle$ such that $s_0 = \mathcal{I}$, $\mathcal{G} \subseteq s_n$, and for each i with $1 \leq i \leq n$, a_i is applicable in s_{i-1}, and $s_i = s_{i-1}[\![a_i]\!]$. We denote the *length* of plan $\pi = \langle a_1, \ldots, a_n \rangle$ as $|\pi| = n$. The *cost* of a plan π is $c(\pi) = \sum_{i=1}^{n} c(a_i)$. A plan is *optimal* if there is no cheaper plan. We slightly abuse notation and let $a_i \in \pi$ denote that action a_i occurs at position i of π.

A *plan reduction* of a plan π for a planning task Π is a subsequence of π that is also a plan for Π (Salerno, Fuentetaja, and Seipp 2023). Techniques for filtering redundant actions from a plan aim to find *perfectly justified* plan reductions for it, i.e., subplans without redundant actions. A plan π for Π is *perfectly justified* if it admits no plan reduction ρ of π such that $|\rho| < |\pi|$. A *Minimal Reduction (MR)* of a plan is a perfectly justified plan reduction of it, with no other perfectly justified plan reduction having a lower cost (Nakhost and Müller 2010; Balyo, Chrpa, and Kilani 2014).

Salerno, Fuentetaja, and Seipp (2025) encode the problem of finding an MR of a given plan as a classical planning task. Their best-performing variant is the *skip one, apply one* compilation, where each action in the original plan is explicitly represented and, at each position, the corresponding action can either be applied or skipped. Since this technique is relevant to our setting, we summarize it next. They define $F_\pi = R_\pi \cup W_\pi$, where R_π is the set of facts that appear in a precondition of the actions in the plan π or in the goal, and W_π is the set of facts that appear in the effects of the actions or the initial state. Then, they define the function τ that, given a fact $v \mapsto d$, maps the fact to itself if it is in R_π and to a new fact named the *irrelevant fact* $(v \mapsto \theta)$ otherwise, where θ denotes that v takes a value that is not used afterward (it is not a precondition for any subsequent action and is not in the goal). The encoding for solving the minimal reduction problems is a planning task $\Pi' = \langle \mathcal{V}', \mathcal{A}', \mathcal{I}', \mathcal{G}', c' \rangle$, with facts $F'_\pi = \{\tau(v \mapsto d) \mid v \mapsto d \in F_\pi\}$, referred to as the *relevant facts* for π; where:[1]

- $\mathcal{V}'$ retains only those variables in $\mathcal{V}$ with multiple values in F'_π and contains a new variable pos to track action positions in the original plan, $\mathcal{D}(pos) = \{0, \ldots, n\}$.
- $\mathcal{A}'$ contains, for each plan action, an *apply* action a_i and a *skip* action $skip_i$. Apply actions have the same preconditions as the original action plus an additional precondition enforcing that the current position is the preceding one. Their effects are those of the original action, mapped by τ to keep only relevant values, plus an additional effect that updates pos. Skip actions simply increment pos.
- $\mathcal{I}'$ contains the facts relevant for the plan $(\mathcal{I} \cap R_\pi)$, plus facts setting variables in $\mathcal{V}'$ with an irrelevant initial value to θ, and one fact setting pos to 0.
- $\mathcal{G}'$ contains the original goals and requires the *pos* variable to be at the end of the original plan.
- c' assigns the cost of the corresponding original action to the *apply* actions and a cost of zero to the *skip* actions.[2]

[1] Π' is denoted as Π^{sl}_{al} in Salerno, Fuentetaja, and Seipp (2025).

[2] The authors apply a cost scaling to deal with zero-cost actions that we do not consider in this paper.

Definition 2 (OSP task). *An* oversubscription planning task *is a tuple* $\Pi_{osp} = \langle \mathcal{V}, \mathcal{A}, \mathcal{I}, c, u, b \rangle$*, where* $\mathcal{V}$*,* $\mathcal{A}$*,* $\mathcal{I}$ *and* c *are as in Definition 1, with* F *as the set of all facts;* $u : F \to \mathbb{N}_0$ *is a fact utility function and* $b \in \mathbb{N}_0$ *is a cost bound.*

A plan π for an OSP task Π_{osp} is a sequence of operators applicable in $\mathcal{I}$ that yields states $s_1, \ldots, s_n$, such that its cumulative cost is less than or equal to the bound, $c(\pi) \leq b$. The utility of a plan $u(\pi)$ is the utility of the end state induced by π, $u(\pi) = u(s_n)$, and the utility of a state s is the sum of the utilities of its facts, $u(s) = \sum_{p \in s} u(p)$. A plan π is optimal if there is no other plan π' such that $u(\pi') > u(\pi)$. Finally, a *soft goal* is any fact $p \in F$ with $u(p) > 0$.

3 Cost-Bounded Plan Reductions

The task to solve is to extract a subsequence of a given plan that is executable and respects the cost limit:

Definition 3 (Cost-Bounded Plan Reduction). *Let* π *be a plan for a classical planning task* $\Pi = \langle \mathcal{V}, \mathcal{A}, \mathcal{I}, \mathcal{G}, c \rangle$ *with facts* F*, a cost bound* $b \in \mathbb{N}_0$*, and a fact utility function* $u : F \to \mathbb{N}_0$*, such that* $u(p) = 0$ *for each fact* $p \in F$ *if* $p \notin \mathcal{G}$*. A* Cost-Bounded Plan Reduction *of* π *is a subsequence of* π *that is a plan for the OSP task* $\Pi_{osp} = \langle \mathcal{V}, \mathcal{A}, \mathcal{I}, c, u, b \rangle$*.*

We denote by $\Delta = \langle \Pi, \pi, b, u \rangle$ the problem of finding a cost-bounded plan reduction of a given plan π for Π subject to cost bound b and fact utility function u. A solution for Δ is optimal if it is an optimal plan for Π_{osp}.[3]

Theorem 1. *Given a cost-bounded plan reduction problem* $\Delta = \langle \Pi, \pi, b, u \rangle$*, determining whether there exists a solution with a utility of at least* M *is NP-complete.*

Proof. Checking whether a subsequence of a given plan π is a plan for Π_{osp} is trivially polynomial, so the problem is in NP. For NP-hardness, we reduce from the classical *Subset Sum* problem. Let $S = \{x_1, \ldots, x_n\}$ be a set of integers and T a target sum. We construct an instance of the cost-bounded plan-reduction problem as follows. The planning task has a binary variable v_i for each integer in S, actions $select_i$ that set v_i to true (i.e., include x_i in the sum), an initial state where all v_i are false, and a goal requiring all v_i to be true. The initial plan is $\pi = \langle select_1, \ldots, select_n \rangle$, with cost function $c(select_i) = x_i$ and fact utility function $u(v_i) = x_i$. The cost bound b and minimal utility M are both T. Any cost-bounded plan reduction of π whose cost does not exceed b and whose utility is at least M corresponds exactly to selecting a subset of integers whose sum is T. □

We propose two methods to solve cost-bounded plan reduction problems: an OSP compilation and an Integer Linear Programming (ILP) formulation.

Compilation to Oversubscription Planning

Our approach to finding a cost-bounded plan reduction of π is to perform a search in the space of π's subsequences using

[3] Note that if $b \geq c(\pi)$, then π itself is a cost-bounded plan reduction of π with maximum utility.

Π: $\mathcal{V} = \{v_1, v_2, v_3, v_4\}$ with Δ
$\mathcal{D}(v_1) = \mathcal{D}(v_2) = \{0, 1\}$, $\mathcal{D}(v_3) = \mathcal{D}(v_4) = \{0, 1, 2, 3\}$;
$\mathcal{A} = \{a_1 = \langle\{v_1 \mapsto 0, v_3 \mapsto 0\}, \{v_1 \mapsto 1, v_3 \mapsto 2\}\rangle$
$a_2 = \langle\{v_1 \mapsto 1, v_2 \mapsto 0\}, \{v_2 \mapsto 1\}\rangle, \ldots\}$
$\mathcal{I} = \{v_i \mapsto 0 \mid \forall i\}$; $\mathcal{G} = \{v_1 \mapsto 1, v_2 \mapsto 1\}$; $c(a_1) = 10, c(a_2) = 5$
$\pi = \langle a_1, a_2\rangle$; $b = 10$
$u(v_1 \mapsto 1) = 1$; $u(v_2 \mapsto 1) = 2$ and $u(v \mapsto d) = 0$ for all other facts

$\mathcal{V}' = \{v_1, v_2, v_3, pos\}$ with Π^r_{osp}
$\mathcal{D}(v_1) = \mathcal{D}(v_2) = \{0, 1\}$, $\mathcal{D}(v_3) = \{0, \theta\}$, $\mathcal{D}(pos) = \{0, 1, 2\}$;
$\mathcal{A}' = \{a'_1 = \langle\{v_1 \mapsto 0, v_3 \mapsto 0, pos \mapsto 0\}, \{v_1 \mapsto 1, v_3 \mapsto \theta, pos \mapsto 1\}\rangle$
$a'_2 = \langle\{v_1 \mapsto 1, v_2 \mapsto 0, pos \mapsto 1\}, \{v_2 \mapsto 1, pos \mapsto 2\}\rangle$
$skip_i = \{\langle\{pos \mapsto i\}, \{pos \mapsto i + 1\ \}\rangle \mid i \in \{1, 2\}\}\}$
$\mathcal{I}' = \{v_i \mapsto 0 \mid \forall i\} \cup \{pos \mapsto 0\}$; $c'(a'_i) = c(a_i)$; $c'(skip_i) = 0$; $b = 10$
$u'(v_1 \mapsto 1) = 1; u'(v_2 \mapsto 1) = 2$ and $u'(v' \mapsto d) = 0$ for all other facts

Figure 1: Example of the OSP task Π^r_{osp} (bottom) for the cost-bounded plan reduction problem Δ (top).

the compilation proposed by Salerno, Fuentetaja, and Seipp (2025), which enables that search by deciding, for each action in the plan, whether to apply it or skip it. But instead of encoding the problem as a classical planning task, we encode it as an OSP task. This is a simple way of introducing the cost bound and the fact utilities into the search process.

Given a cost-bounded plan reduction problem $\Delta = \langle\Pi, \pi, b, u\rangle$ with $\Pi = \langle\mathcal{V}, \mathcal{A}, \mathcal{I}, \mathcal{G}, c\rangle$ and $\pi = \langle a_1, \ldots, a_n\rangle$, we make soft goals always relevant, redefining R_π (originally the set of facts in a precondition of the actions in the plan or in the goal) as the set of facts that appear in a precondition of the actions in the plan π or that have positive utility:

$$R_\pi = \bigcup_{a_i \in \pi} pre(a_i) \cup \{p \in F \mid u(p) > 0\}$$

We generate the OSP task $\Pi^r_{osp} = \langle\mathcal{V}', \mathcal{A}', \mathcal{I}', c', u', b\rangle$, where $\mathcal{V}'$, $\mathcal{A}'$, $\mathcal{I}'$ and c' are the same as for the original encoding for Π^r but considering the new definition of R_π, $u'(v') = u(v)$ for all $v' \in \mathcal{V}' \setminus \{pos\}$, and $u'(pos) = 0$. Figure 1 illustrates the OSP compilation with a simple example.

Theorem 2. *Let Δ be a cost-bounded plan reduction problem and let Π^r_{osp} be the corresponding OSP task. Then, the set of solutions for Π^r_{osp} is exactly the set of solutions for Δ, and a solution for Π^r_{osp} is optimal iff it is optimal for Δ.*

Proof. By Definition 3, any solution of Δ is a plan π' for Π_{osp}. The compiled task Π^r_{osp} unfolds the original plan π position by position and, at each step, allows either *applying* or *skipping* the corresponding action, preserving its relevant preconditions and effects; cost and utility functions coincide with those of Π_{osp}. π' can be translated into a plan for Π^r_{osp} by selecting *apply* actions exactly at the positions of the actions in π' and *skip* actions elsewhere. Conversely, any plan for Π^r_{osp} induces a subsequence of π by removing all *skip* steps; the compilation guarantees that this subsequence is a plan for Π_{osp}. Hence, there is a one-to-one correspondence between solutions of Π^r_{osp} and solutions of Δ, including the optimal ones. □

Integer Linear Programming Formulation

To solve the cost-bounded plan reduction problem $\Delta = \langle\Pi, \pi, b, u\rangle$, we formulate the associated OSP task $\Pi^r_{osp} = \langle\mathcal{V}', \mathcal{A}', \mathcal{I}', c', u', b\rangle$, with facts F', as a 0-1 Integer Linear Program (ILP). As in standard linear programming models defined for classical and partial satisfaction planning (Vossen et al. 1999; van den Briel et al. 2004; van den Briel and Kambhampati 2005), we use binary variables to track the truth value of each fact at every layer of the horizon, binary action variables to encode action execution, and linear constraints that impose action preconditions, propagate action effects, and enforce fact persistence (frame axioms). However, our ILP formulation focuses on *plan reduction* rather than *plan synthesis*. Thus, the horizon is fixed to the length n of the given plan and the action set consists solely of the *apply* and the *skip* actions for each plan position. Our objective is to maximize the utility of the final state induced by the resulting subplan while respecting the cost bound.

The model is as follows: There are two families of binary decision variables: (1) fact variables $x_{p,i}$, defined for each fact $p \in F'$ and layer $i \in \{0, \ldots, n\}$, and indicating whether p holds in the state at layer i; and (2) action variables $y_{a,i}$, defined for each action $a \in \mathcal{A}'_i$, where $\mathcal{A}'_i = \{a_i, skip_i\}$, $\mathcal{A}'_i \subseteq \mathcal{A}'$, and for every plan position $i \in \{1, \ldots, n\}$; indicating whether action a is selected at layer i. Since the representation of state variables is propositional, we explicitly encode the add and delete effects of actions. For each action a_i, $add(a_i) = \mathit{eff}(a_i)$, and $del(a_i) = \{v \mapsto d \in pre(a_i) \mid v \mapsto d' \in \mathit{eff}(a_i),\ d' \neq d\} \cup \{v \mapsto d \in F' \mid v \notin vars(pre(a_i)), v \mapsto d' \in \mathit{eff}(a_i),\ d' \neq d\}$.

Objective Maximize the utility of the final state:

$$\max \sum_{\substack{p \in F' \\ u'(p) > 0}} u'(p) \cdot x_{p,n}$$

Subject to

1. Cost bound: $\sum_{i=1}^{n} \sum_{a \in \mathcal{A}'_i} c'(a_i) \cdot y_{a,i} \leq b$
2. Initial state: $x_{p,0} = \begin{cases} 1, & \text{if } p \in \mathcal{I}', \\ 0, & \text{otherwise} \end{cases} \quad \forall p \in F'$
3. Exactly one action per layer:
$$y_{a_i,i} + y_{skip_i,i} = 1 \quad \forall i \in \{1, \ldots, n\}$$
4. Precond. of selected actions must be true in the previous layer:
$$y_{a,i} \leq x_{p,i-1} \quad \forall a \in \mathcal{A}'_i,\ \forall p \in pre(a),\ \forall i \in \{1, \ldots, n\}$$
5. Del effects of selected actions must be false at the action layer:
$$x_{p,i} \leq 1 - y_{a,i} \quad \forall a \in \mathcal{A}'_i,\ \forall p \in del(a),\ \forall i \in \{1, \ldots, n\}$$
6. Add effects of selected actions must be true at the action layer:
$$x_{p,i} \geq y_{a,i} \quad \forall a \in \mathcal{A}'_i,\ \forall p \in add(a),\ \forall i \in \{1, \ldots, n\}$$
7. Facts not modified by selected actions persist:
$$x_{p,i} \geq x_{p,i-1} - \sum_{\substack{a \in \mathcal{A}'_i \\ p \in del(a)}} y_{a,i} \quad \forall p \in F',\ \forall i \in \{1, \ldots, n\}$$
$$x_{p,i} \leq x_{p,i-1} + \sum_{\substack{a \in \mathcal{A}'_i \\ p \in add(a)}} y_{a,i} \quad \forall p \in F',\ \forall i \in \{1, \ldots, n\}$$

The ILP model represents Π^r_{osp}, and therefore any feasible solution for it is a solution for Δ, and any optimal solution is an optimal solution for Δ.

Domain	25%		50%		75%	
	OSP	ILP	OSP	ILP	OSP	ILP
barman (20)	17	20	14	20	13	20
elevators (20)	4	20	2	20	2	20
floortile (10)	10	10	10	10	10	10
nomystery (20)	20	20	20	20	20	20
openstacks (20)	2	15	0	15	0	15
parcprinter (20)	11	20	8	20	8	20
parking (20)	20	20	20	20	20	20
pegsol (20)	20	20	20	20	20	20
scanalyzer (20)	20	20	20	20	20	20
sokoban (20)	20	20	20	20	20	20
tidybot (19)	18	19	18	19	18	19
transport (17)	1	17	1	17	0	17
visitall (20)	2	5	1	5	1	5
woodworking (20)	4	20	2	20	2	20
Total (**266**)	169	**246**	156	**246**	154	**246**

Table 1: Coverage per domain, cost bound percentage, and approach (OSP vs. ILP) for heterogeneous utilities.

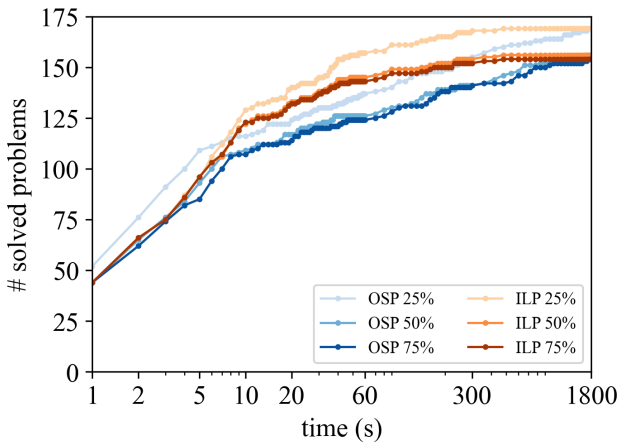

Figure 2: Coverage over time for heterogeneous utilities, restricted to instances solved by both methods (OSP and ILP).

4 Experiments

To evaluate the two approaches, we use the OSP versions of the 14 domains of the IPC 2011 created by García-Olaya, de la Rosa, and Borrajo (2021). We generate initial plans for every instance with the Fast Downward planner[4], using the *seq-sat-fdss-2018* portfolio with a time limit of 30 minutes per instance and requesting a single plan. For 14 problems, no plan was found within the time limit: 10 in *Floortile*, 1 in *Tidybot* and 3 in *Transport*. These problems were discarded from the experiments, since we require a valid initial plan as input. Then, we create problems with different degrees of oversubscription, with cost bounds of 25%, 50%, and 75% of their solution cost. As in García-Olaya, de la Rosa, and Borrajo (2021), we consider two different utility schemes: one where all goals are assigned the same value, and another where each goal is assigned an integer utility in $[1, 10]$.

All experiments were run on an Intel Xeon X3470 (2.93 GHz) under Ubuntu 20.04.6 with 30 GB of RAM and a wall-clock limit of 30 minutes per run. All benchmarks, code, and data are available online (Del Toro, Fuentetaja, and García-Olaya 2026). For the OSP approach, we use *Sym-Osp* (Speck and Katz 2021), a state-of-the-art optimal oversubscription symbolic planner. For the ILP formulation, we use CPLEX 12.10.[5]

Table 1 reports the coverage results for heterogeneous utilities. ILP achieves higher coverage for all the oversubscription degrees, but under the most restrictive budget (25%) the gap between both approaches is less pronounced, indicating that OSP behaves comparatively better under severe budget reductions. For brevity, we omit the homogeneous case, which shows a similar pattern: OSP solves only a few additional instances (177 at 25%, 161 at 50%, and 158 at 75%), since equal goal utilities make the problem easier.

[4]Available at https://www.fast-downward.org.

[5]Available at the IBM CPLEX Optimization Studio website: https://www.ibm.com/products/ilog-cplex-optimization-studio.

Figure 2 shows the coverage over time of both approaches for the common solved problems. ILP is faster in general, except for the simplest 25% problems.

ILP solves the same number of instances across all budget levels, whereas OSP exhibits significant variation. In ILP, the model structure (variables and constraints) is fixed for a given instance, and changing the budget only modifies the cost bound constraint. By contrast, in OSP, the budget directly affects the set of possible plans: higher bounds allow longer subplans and many more feasible combinations, which can make search harder. Similar effects have been reported before: ILP model size scales with plan length (Kautz and Walser 2000; van den Briel and Kambhampati 2005), and symbolic OSP can require substantially more state expansions as the cost bound increases (Speck and Katz 2021).

OSP never solves instances with plans much longer than 450 actions, whereas ILP still finds solutions around 700 actions. Such long plans appear only in a few domains, notably *OpenStacks* and *Visitall*, whose plans can exceed 1000 actions and where OSP coverage drops sharply.

Beyond plan length, key factors limiting OSP coverage are goal interaction and the number of goals. Domains such as *Elevators*, *Transport*, *ParcPrinter*, and *Woodworking* exhibit tightly coupled subgoals. *OpenStacks* combines strong goal interactions with a large number of goals, while *Visitall* is challenging primarily due to its high goal count.

5 Conclusion

We have formalized and optimally solved the cost-bounded plan-reduction problem by introducing two methods: an OSP compilation and an ILP formulation. Experiments show that ILP achieves higher coverage across budget levels. As future work, we plan to explore alternative ILP formulations, to integrate the method with satisficing OSP approaches based on heuristic goal selection to bring plans that exceed the cost bound back within it (García-Olaya, de la Rosa, and Borrajo 2021) and to incorporate explainability features.

Acknowledgements

This work was partially supported by MICIU/AEI/10.13039 /501100011033, by "ERDF A way of making Europe", and by "ESF+", under grants PID2021-127647NB-C21 and PRE2022-104392.

References

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

Balyo, T.; Chrpa, L.; and Kilani, A. 2014. On Different Strategies for Eliminating Redundant Actions from Plans. In Edelkamp, S.; and Barták, R., eds., *Proceedings of the Seventh Annual Symposium on Combinatorial Search (SoCS 2014)*, 10–18. AAAI Press.

Chrpa, L.; McCluskey, T. L.; and Osborne, H. 2012a. Determining Redundant Actions in Sequential Plans. In *Proceedings of the 24th International Conference on Tools with Artificial Intelligence (ICTAI 2012)*, 484–491. IEEE.

Chrpa, L.; McCluskey, T. L.; and Osborne, H. 2012b. Optimizing Plans through Analysis of Action Dependencies and Independencies. In McCluskey, L.; Williams, B.; Silva, J. R.; and Bonet, B., eds., *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling (ICAPS 2012)*, 338–342. AAAI Press.

Del Toro, M.; Fuentetaja, R.; and García-Olaya, A. 2026. Code and Data for the ICAPS 2026 Paper "Finding Optimal Cost-Bounded Plan Reductions". Available at https://doi.org/10.5281/zenodo.18957117.

Domshlak, C.; and Mirkis, V. 2015. Deterministic Oversubscription Planning as Heuristic Search: Abstractions and Reformulations. *Journal of Artificial Intelligence Research*, 52: 97–169.

Fink, E.; and Yang, Q. 1992. Formalizing Plan Justifications. In *Proceedings of the Ninth Conference of the Society for Computational Studies of Intelligence*, 9–14.

Fox, M.; Gerevini, A.; Long, D.; and Serina, I. 2006. Plan Stability: Replanning versus Plan Repair. In Long, D.; Smith, S. F.; Borrajo, D.; and McCluskey, L., eds., *Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling (ICAPS 2006)*, 212–221. AAAI Press.

García-Olaya, A.; de la Rosa, T.; and Borrajo, D. 2021. Selecting goals in oversubscription planning using relaxed plans. *Artificial Intelligence*, 291: 103414.

Katz, M.; and Keyder, E. 2022. A* Search and Bound-Sensitive Heuristics for Oversubscription Planning. In Honavar, V.; and Spaan, M., eds., *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI 2022)*, 9813–9820. AAAI Press.

Katz, M.; Keyder, E.; Pommerening, F.; and Winterer, D. 2019. Oversubscription Planning as Classical Planning with Multiple Cost Functions. In Lipovetzky, N.; Onaindia, E.; and Smith, D. E., eds., *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling (ICAPS 2019)*, 237–245. AAAI Press.

Kautz, H. A.; and Walser, J. P. 2000. Integer optimization models of AI planning problems. *The Knowledge Engineering Review*, 15(1): 101–117.

Nakhost, H.; and Müller, M. 2010. Action Elimination and Plan Neighborhood Graph Search: Two Algorithms for Plan Improvement. In Brafman, R.; Geffner, H.; Hoffmann, J.; and Kautz, H., eds., *Proceedings of the Twentieth International Conference on Automated Planning and Scheduling (ICAPS 2010)*, 121–128. AAAI Press.

Salerno, M.; Fuentetaja, R.; and Seipp, J. 2023. Eliminating Redundant Actions from Plans using Classical Planning. In Marquis, P.; Son, T. C.; and Kern-Isberner, G., eds., *Proceedings of the Twentieth International Conference on Principles of Knowledge Representation and Reasoning (KR 2023)*, 774–778. IJCAI Organization.

Salerno, M.; Fuentetaja, R.; and Seipp, J. 2025. Finding Minimal Plan Reductions Using Classical Planning. *Journal of Artificial Intelligence Research*, 84: 1–35.

Smith, D. E. 2004. Choosing Objectives in Over-Subscription Planning. In Zilberstein, S.; Koehler, J.; and Koenig, S., eds., *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2004)*, 393–401. AAAI Press.

Speck, D.; and Katz, M. 2021. Symbolic Search for Oversubscription Planning. In Leyton-Brown, K.; and Mausam, eds., *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI 2021)*, 11972–11980. AAAI Press.

van den Briel, M.; and Kambhampati, S. 2005. Optiplan: A Planner Based on Integer Programming. *Journal of Artificial Intelligence Research*, 24: 919–931.

van den Briel, M.; Sanchez, R.; Do, M. B.; and Kambhampati, S. 2004. Effective Approaches for Partial Satisfaction (Over-Subscription) Planning. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI 2004)*, 562–569. AAAI Press.

Vossen, T.; Ball, M.; Lotem, A.; and Nau, D. 1999. On the Use of Integer Programming Models in AI Planning. In Dean, T., ed., *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI 1999)*, 304–309. Morgan Kaufmann.