Proceedings of the 22nd International Conference on Principles of Knowledge Representation and Reasoning
KR in Planning and Scheduling

Cost-Optimal Delete-Free Classical Planning via Maximum Satisfiability

Masood Feyzbakhsh Rankooh , Andreas Niskanen , Matti Järvisalo
HIIT, Department of Computer Science, University of Helsinki, Finland
{masood.feyzbakhshrankooh,andreas.niskanen,matti.jarvisalo}@helsinki.fi

Abstract

We propose a maximum satisfiability (MaxSAT) based approach to cost-optimal delete-free planning, also known as optimal relaxed planning. Relaxed planning is a central subclass of classical planning, consisting of computing the h^+ heuristic for classical planning. As an alternative to the existing approaches to exactly computing h^+, we propose a maximum satisfiability (MaxSAT) based approach, motivated by the success of SAT-based planners and significant recent advances in MaxSAT solvers. Concretely, we both adapt a recent answer set optimization approach to computing h^+ for MaxSAT, propose further MaxSAT encoding variants for both representing cost-optimal plans and plan acyclicity, and combine them for further runtime improvements. Overall, our MaxSAT approach compares favourably to the current state-of-the-art answer set optimization approach.

1 Introduction

Delete-free problems (Hoffmann 2005) constitute an important class in classical planning (Ghallab, Nau, and Traverso 2016). Any classical planning problem with no negative preconditions can be relaxed into a delete-free one by removing negative effects of actions. Cost-optimal delete-free planning, i.e., computing the optimal cost h^+ of a given delete-free planning problem and an optimal plan witnessing the value, has various motivations.

Firstly, various settings give naturally rise to inherently delete-free planning problems, with e.g. join-order optimization (Robinson, McIlraith, and Toman 2014) and the minimal seed-set problem (Gefen and Brafman 2011) as two examples. Secondly, iteratively solving and strengthening the delete-free relaxation of a planning problem (essentially in the style of counterexample-guided abstraction refinement) allows for finding cost-optimal plans to the original problem (Haslum 2012). Thirdly, as h^+ is a lower bound for the optimal cost of the non-relaxed problem, it is a quite strong admissible planning heuristic. However, although easier than cost-optimal classical planning in general, computing h^+ remains presumably hard to compute (Bylander 1994) and to approximate (Betz and Helmert 2009). In fact, various polynomial-time computable admissible heuristics, including $h^{\max}$ (Bonet and Geffner 1999), its cost-sharing approximations (Mirkis and Domshlak 2007), and LM-cut (Helmert and Domshlak 2009) providing lower bounds for h^+ have been introduced. Approaches to exactly determining h^+ fast enough for practical applications could hence subsume such heuristics as more informative.

Various approaches to exactly computing h^+ have been proposed. These include early work on maximum satisfiability (MaxSAT) (Zhang and Bacchus 2012) as well as increasingly effective integer linear programming formulations approaches (Imai and Fukunaga 2015), the use of Boolean satisfiability (SAT) (Rankooh and Rintanen 2022b) solvers, and most recently answer set programming (ASP) (Rankooh and Janhunen 2023). The use of symbolic techniques such as (Max)SAT and ASP is in particularly motivated by the success of SAT-based planners (Kautz and Selman 1999; Rintanen, Heljanko, and Niemelä 2006; Suda 2014), as classical planning tasks allow for natural propositional representations.

Two key considerations for declarative encodings of delete-free planning and thereby computing h^+ are (i) how cost-optimality is represented and (ii) how acyclicity of plans as a central underlying constraint in the problem is enforced. The recent ASP-based approach (Rankooh and Janhunen 2023), and in particular the so-called diagnostic encoding of cost-optimal relaxed plans as answer set optimization using a vertex elimination encoding of plan acyclicity (Rankooh and Rintanen 2022c), is the current state-of-the-art approach to computing h^+: this ASP-based approach was shown to outperform the earlier state-of-the-art approach implementing similar ideas via integer linear programming (Rankooh and Rintanen 2022a) which was earlier shown to outperform an approach based on employing a SAT solver iteratively in a way that allows for reasoning about cost-optimality.

In this work, we harness recent advances in MaxSAT (Bacchus, Järvisalo, and Martins 2021) for cost-optimal delete-free planning. While MaxSAT has been employed for delete-free planning (Zhang and Bacchus 2012), predating the recent advances in ILP and ASP-based computation of h^+ as well as many more recent advances in MaxSAT solving, we go beyond (Zhang and Bacchus 2012) where a simpler encoding and a more straightforward cycle prevention approach was proposed by developing more compact encodings and by employing modern MaxSAT solvers. Indeed, with significant recent advances in various types of increasingly efficient MaxSAT solvers (Bacchus,

Järvisalo, and Martins 2021), employing MaxSAT allows for studying the suitability of a range of search techniques for computing h^+, whereas the choice of ASP solvers remains today more limited.

In terms of problem encodings, we contribute to both the choice of encodings of cost-optimality and the choice of how plan acyclicity is enforced. In terms of the former, we firstly consider a MaxSAT formulation of the diagnosis ASP encoding, and argue that the bijective relation between cost-optimal plans and supported models of the diagnostic ASP encoding transfers also to classical models under an analogous MaxSAT encoding because the diagnostic ASP encoding essentially does not lean in any particular way on answer set semantics. Building on the earlier SAT-based approach (Rankooh and Rintanen 2022b), we also consider a so-called proposition-based variant of the diagnostic MaxSAT encoding, which allows for achieving a more compact encoding of the objective function and thereby smaller and worst-case fewer unsatisfiable cores in the MaxSAT instances arising from the encoding. This impact of unsatisfiable cores is in particular interesting as the dominant algorithmic approaches implemented in modern MaxSAT solvers are based on iteratively extracting unsatisfiable cores which are iteratively relaxed away towards optimal solutions—specifically, more compact unsatisfiable cores are intuitively beneficial for faster termination. In terms of enforcing plan acyclicity, we propose a novel *cycle elimination* encoding as an alternative to the so-far state-of-the-art vertex elimination encoding. We evaluate state-of-the-art MaxSAT solvers on he MaxSAT encodings both individually as well as their combinations on standard classical planning problems against the current state-of-the-art ASP-based approach. Our MaxSAT approach compares favourably to the ASP-based approach, and we show that further performance improvements are obtained using combinations for the MaxSAT encodings and by tuning unsatisfiability core extraction.

2 Preliminaries

We briefly recall necessary background.

2.1 Classical Planning and Relaxed Plans

A typical STRIPS planning problem is represented as a five-tuple $\Pi = \langle X, I, A, G, \mathrm{cost}\rangle$, where X is a finite set of Boolean state variables (atomic propositions), $I \subseteq X$ represents the initial state, $G \subseteq X$ denotes the set of goal conditions, and A is a finite set of actions. Each action $a \in A$ is described by the triple $a = \langle \mathrm{pre}(a), \mathrm{add}(a), \mathrm{del}(a)\rangle$, where $\mathrm{pre}(a)$, $\mathrm{add}(a)$, and $\mathrm{del}(a)$ are subsets of X specifying the preconditions, positive effects, and negative effects, respectively. The function $\mathrm{cost} : A \to \mathbb{N}_0$ assigns a non-negative cost to each action.

A state is modeled as a subset of X. The successor state s' resulting from executing an action a in a state s, denoted also by $\mathrm{exec}_a(s)$, is defined as $s' = (s \setminus \mathrm{del}(a)) \cup \mathrm{add}(a)$, only if $\mathrm{pre}(a) \subseteq s$. A sequence of actions (a plan) $\pi = \langle a_1, a_2, \ldots, a_n\rangle$ is *executable* from s if $\mathrm{exec}_{a_n}(\cdots \mathrm{exec}_{a_2}(\mathrm{exec}_{a_1}(s)) \cdots)$ is defined. The plan π is a solution for Π if executing it on the initial state I yields a state in which every $g \in G$ holds. Its cost is the sum of the individual action costs, and an *optimal plan* minimizes this total cost. The *delete relaxation* of Π, denoted Π^+, is formed by replacing, for each action $a \in A$, the delete list $\mathrm{del}(a)$ with the empty set. A plan for Π^+ is called a *relaxed plan* for Π, and its minimal cost is denoted by $h^+(\Pi)$ (with $h^+(\Pi) = \infty$ if no relaxed plan exists).

Example 1 (Blocksworld Domain)**.** *The blocksworld domain is a classical planning benchmark in which a robot arm can move blocks between a table and other blocks, with the goal of reaching a desired stacking configuration. There are three primitive action schemas:*

- `move-b-b(A,B,C)`*: pick up block A from the top of block B and place it onto the top of block C.*
  - $\mathrm{pre}(\texttt{move-b-b}(A,B,C)) = \{on(A,B),\ clear(A),\ clear(C)\}$
  - $\mathrm{add}(\texttt{move-b-b}(A,B,C)) = \{on(A,C),\ clear(B)\}$
  - $\mathrm{del}(\texttt{move-b-b}(A,B,C)) = \{on(A,B),\ clear(C)\}$
- `move-b-t(A,B)`*: pick up block A from the top of block B and place it onto the table.*
  - $\mathrm{pre}(\texttt{move-b-t}(A,B)) = \{on(A,B),\ clear(A)\}$
  - $\mathrm{add}(\texttt{move-b-t}(A,B)) = \{onTable(A),\ clear(B)\}$
  - $\mathrm{del}(\texttt{move-b-t}(A,B)) = \{on(A,B)\}$
- `move-t-b(A,B)`*: pick up block A from the table and place it onto the top of block B.*
  - $\mathrm{pre}(\texttt{move-t-b}(A,B)) = \{onTable(A),\ clear(A),\ clear(B)\}$
  - $\mathrm{add}(\texttt{move-t-b}(A,B)) = \{on(A,B)\}$
  - $\mathrm{del}(\texttt{move-t-b}(A,B)) = \{onTable(A),\ clear(B)\}$

In this example problem, we have three blocks A, B, and C. Initially, block A is on the table, block B is on block A, and block C is on block B. Formally, the initial state is $I = \{onTable(A), on(B,A), on(C,B), clear(C)\}$. The goal is to stack block A on block C while retaining block C on block B. Formally, $G = \{on(A,C),\ on(C,B)\}$.

The optimal valid plan with cost 4 is $\langle \texttt{move-b-t}(C,B), \texttt{move-b-t}(B,A), \texttt{move-t-b}(C,B), \texttt{move-t-b}(A,C)\rangle$. An optimal relaxed plan (ignoring delete effects) is $\langle \texttt{move-b-t}(C,B), \texttt{move-b-t}(B,A), \texttt{move-t-b}(A,C)\rangle$ with cost 3, since the delete effect of $on(C,B)$ by $\texttt{move-b-t}(C,B)$ is ignored in the relaxation.

2.2 Maximum Satisfiability (MaxSAT)

For a Boolean variable x there are two literals, x and $\neg x$. A clause C is a disjunction ($\vee$) of literals. A CNF formula F is a conjunction ($\wedge$) of clauses. We denote by $V(F)$ variables of F. An assignment $\tau\colon V(F) \to \{0,1\}$ maps variables to 0 (false) or 1 (true), and extending to literals via $\tau(\neg x) = 1 - \tau(x)$, to clauses via $\tau(C) = \max\{\tau(l) \mid l \in C\}$, and to formulas via $\tau(F) = \min\{\tau(C) \mid C \in F\}$. The *Boolean satisfiability* problem (SAT) (Biere et al. 2021) asks if a given formula F is satisfiable, i.e., is there an assignment τ with $\tau(F) = 1$, i.e., a model of F; if not, F is unsatisfiable. In (weighted partial) MaxSAT (Bacchus, Järvisalo,

and Martins 2021) the input consists of "hard" clauses F_{hard}, "soft" clauses F_{soft}, and a weight function w mapping each soft clause $C \in F_{\text{soft}}$ to an integer $w(C) \geq 0$. The task is to find an assignment τ which satisfies F_{hard} and minimizes cost $c(\tau) = \sum_{C \in F_{\text{soft}}} w(C)(1 - \tau(C))$ incurred by not satisfying soft clauses.

Central to MaxSAT solving are unsatisfiable cores (Bacchus, Järvisalo, and Martins 2021). For a MaxSAT instance $(F_{\text{hard}}, F_{\text{soft}}, w)$, a subset $\kappa \subseteq F_{\text{soft}}$ is a core for the instance if $F_{\text{hard}} \wedge \kappa$ is unsatisfiable. In the encodings we present, the soft clauses will be unit clauses (i.e., consist of a single literal) encoding a linear objective function; hence cores are essentially clauses over subsets of objective literals. A core κ is minimal, i.e., a minimal unsatisfiable subset (MUS), if $F_{\text{hard}} \wedge \kappa'$ is satisfiable for each $\kappa' \subset \kappa$.

2.3 Answer Set Programming (ASP)

ASP (Gelfond and Lifschitz 1988; Niemelä 1999) is a declarative paradigm for solving search problems via logic programs. In our context, a logic program consists of rules that can be expressed in one of the following forms:

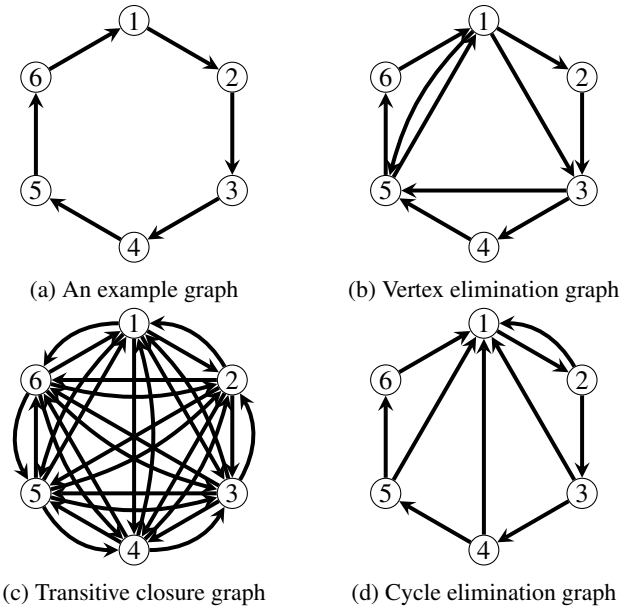
$$a \leftarrow b_1, \ldots, b_n, \text{ not } c_1, \ldots, \text{not } c_m, \quad (1)$$

$$\{a\} \leftarrow b_1, \ldots, b_n, \text{ not } c_1, \ldots, \text{not } c_m. \quad (2)$$

Here a, $b_1, \ldots, b_n$, and $c_1, \ldots, c_m$ are propositional atoms, and `not` represents default negation. Equation (1) is referred to as a *normal rule*, while Equation (2) is a *choice rule*. For a rule r, we denote by $\text{body}^+(r)$ the set of atoms appearing positively, and by $\text{body}^-(r)$ the set of atoms occurring under negation. A rule is *positive* if $\text{body}^-(r) = \emptyset$.

The signature of a program P, denoted $\text{At}(P)$, is the set of all atoms that occur in P. Its *positive dependency graph*, $DG^+(P)$, is the directed graph whose nodes are the atoms in $\text{At}(P)$ and where there is an edge from an atom b to an atom a if there exists a rule r in P with head a and $b \in \text{body}^+(r)$.

An *interpretation* is a subset $I \subseteq \text{At}(P)$; the atoms in I are assigned the truth value `true`, while those not in I are `false`. An interpretation satisfies a normal rule if, in the case that all literals in the body are true, the head is also true. Any interpretation satisfies a choice rule unconditionally. A (classical) model of a program is an interpretation that satisfies every rule in P. Every positive normal program, in which rules contain no negative body literals, has a unique *least model*, denoted by $LM(P)$, which is the intersection of all its models. Given an interpretation I, the *Gelfond–Lifschitz reduct* of a rule r, denoted r^I, is obtained by removing the negative literals; if any atom in $\text{body}^-(r)$ is in I, then r^I is discarded. If r is a choice rule, r^I is also discarded if its head is not in I. The reduct of a program P, written P^I, is the set of all such reducts. An interpretation I is a *stable model* if it satisfies $I = LM(P^I)$. Finally, an interpretation is said to be *supported* if it consists exactly of the heads of those rules whose bodies are satisfied; note that while every stable model is supported, the converse is not necessarily true.



(a) An example graph (b) Vertex elimination graph

(c) Transitive closure graph (d) Cycle elimination graph

Figure 1: Comparison of acyclicity encodings

2.4 Vertex Elimination Graphs

To encode acyclicity constraints effectively—as also employed in the recent ASP-based approach to cost-optimal delete-free planning—we employ the concept of vertex elimination graphs (Rose and Tarjan 1975). Let $G = \langle V, E \rangle$ be a directed graph, where V is the set of vertices and E is the set of directed edges. An ordering of the vertices is given by a bijection $\alpha : \{1, \ldots, n\} \to V$. For each vertex $v \in V$, the *fill-in* of v is defined as $F(v) = \{\langle x, y \rangle \mid \langle x, v \rangle \in E,\ \langle v, y \rangle \in E,\ x \neq y\}$. The elimination of v from G is performed by removing v and adding its fill-in, yielding the graph $G(v) = \langle V \setminus \{v\},\ E(v) \cup F(v) \rangle$, where $E(v) = \{\langle x, y \rangle \in E \mid x \neq v \text{ and } y \neq v\}$.

Given an ordering α, the vertex elimination process produces a sequence $G = G_0, G_1, \ldots, G_{n-1}$, where for each $i = 1, \ldots, n-1$, the graph G_i is the $\alpha(i)$-elimination graph of G_{i-1}. The cumulative fill-in with respect to α is defined as $F_\alpha(G) = \bigcup_{i=1}^{n-1} F_{i-1}(\alpha(i))$, with $F_{i-1}(\alpha(i))$ denoting the fill-in computed in G_{i-1} for vertex $\alpha(i)$. Finally, the *vertex elimination graph* corresponding to the ordering α is given by $G^*_\alpha = \langle V,\ E \cup F_\alpha(G) \rangle$. A key property of this construction is that if the original graph G contains a cycle, then for any ordering α, the graph G^*_α will contain a cycle of length two. This property is exploited to enforce acyclicity in our encoding.

Example 2. *Consider the cyclic graph presented in Figure 1a. Assume that elimination order is* $2, 4, 6, 3, 5, 1$. *The vertex elimination graph, produced by eliminating vertices in this order is depicted in Figure 1b. The vertex elimination graph has the cycle* $1, 5, 1$ *of length 2.*

3 From ASP to MaxSAT Encoding of h^+

In this section, we review the relaxed-planning heuristic h^+ as an answer set program whose supported models correspond exactly to relaxed plans. We begin by revisiting the diagnostic encoding of h^+ (Rankooh and Janhunen 2023) which inverts the usual causal propagation and performs a backward "diagnosis" from goal atoms to the actions that support them. While it is already established that supported models of such an encoding represent relaxed plans, we demonstrate here that its classical models also admit a one-to-one correspondence with valid relaxed plans. After presenting the core rules of the diagnostic encoding and their MaxSAT-equivalent clauses, we enforce acyclicity via the vertex elimination method to eliminate invalid cyclic plans.

Although the SAT encoding of Rankooh and Rintanen (2022b) may seem like a natural starting point for a MaxSAT formulation, we deliberately base our encoding on the ASP-based formulation of Rankooh and Janhunen (2023). While both approaches share key mechanisms, such as vertex elimination for enforcing acyclicity, they differ significantly in motivation and formal underpinnings. The ASP encoding builds on supported-model semantics and uses rule-based ASP constructs, whereas the SAT formulation directly encodes the causal relaxed plan representation in propositional logic. Consequently, several distinctions emerge: (1) some bidirectional implications in the SAT encoding become single-directional in the ASP encoding and our formulation, (2) the mutual exclusion constraint in the SAT encoding is absent in the ASP encoding and our version, as it is only necessary when minimizing proposition costs rather than action costs, and (3) auxiliary variables used in the SAT encoding's acyclicity constraints are omitted in the ASP encoding and in ours. Moreover, the ASP encoding of Rankooh and Janhunen is more recent, reflects current state-of-the-art in h^+ computation, and—as reported later on—in our experiments led to more efficient solving than adapting the SAT encoding of Rankooh and Rintanen (2022b).

3.1 The Diagnostic Encoding

For the sake of simplicity, and without loss of generality, assume that for any given relaxed planning problem, the atoms in the initial state have always been removed from preconditions and effects of all actions, and also from the goal. Following the diagnostic encoding of relaxed planning in (Rankooh and Janhunen 2023), given a relaxed planning problem $\Pi^+ = \langle X, \emptyset, A, G, \mathrm{cost}\rangle$, the encoding P_d consists of the following rules, where X is the set of atomic propositions, A the set of actions, and $G \subseteq X$ the goals:

$$\{\, p\}, \quad p \in X, \tag{3}$$

$$\{\, ws(a,p)\} \leftarrow p, \quad p \in \mathrm{add}(a), \tag{4}$$

$$f \leftarrow p,\ \mathrm{not}\ ws(a_1,p), \ldots, \mathrm{not}\ ws(a_m,p),\ \mathrm{not}\ f,$$
$$\{a_1,\ldots,a_m\} = \{a \mid p \in \mathrm{add}(a)\}, \tag{5}$$

$$\mathtt{dep}(p,q) \leftarrow ws(a,p), \quad q \in \mathrm{pre}(a), \tag{6}$$

$$a \leftarrow ws(a,p), \quad p \in \mathrm{add}(a), \tag{7}$$

$$q \leftarrow a, \quad q \in \mathrm{pre}(a), \tag{8}$$

$$g \leftarrow \mathrm{not}\, g, \quad g \in G. \tag{9}$$

These rules allow any atom p to be assumed true (rule 3), enforce that each true atom is provided by at least one well-supporting action (rules 4–5), propagate support from actions back to their preconditions (rules 6 and 8), link each well-support atom back to its action so that action costs can be minimized (rule 7), and force every goal g to hold (rule 9). Note that the symbol f of rule 5 is an auxiliary atom used to ensure contradiction in the case that the body is true.

This encoding has been called "diagnostic" because it inverts the causal inference used in earlier encodings: rather than propagating information from preconditions to dependencies, from dependencies to well-support atoms, and from well-support atoms to effects, it performs a backward analysis from effects through well-supports to dependencies and then to preconditions (Rankooh and Janhunen 2023). By viewing each achieved atom as an observation to be explained, the diagnostic encoding identifies supporting actions and traces their preconditions in reverse, thereby "diagnosing" which actions must have occurred to produce the desired effects. This reversal of direction aligns with classical notions of diagnostic reasoning in logic-based systems (Russell and Norvig 2010).

Let us define "acyclic models" as those models in which the directed graph formed by the atoms $\mathtt{dep}(p,q)$ is acyclic. It has been shown by Rankooh and Janhunen (2023) that the diagnostic encoding admits an acyclic supported model exactly when there exists a relaxed plan: every subset of actions that can be ordered into a valid relaxed plan corresponds to an acyclic supported model in which precisely those action atoms are true, and conversely every acyclic supported model identifies a valid relaxed plan. This one-to-one correspondence ensures that searching for acyclic supported models of the diagnostic program is equivalent to enumerating all relaxed plans of the original planning task.

To guarantee that all produced models are acyclic, the encoding adds two auxiliary rules that (i) simulate the vertex elimination process given an elimination order α, and (ii) forbid any cycle of length two:

$$\mathtt{dep}(p,q) \leftarrow \mathtt{dep}\big(p,\alpha(i)\big),\ \mathtt{dep}\big(\alpha(i),q\big),$$
$$\langle p,q\rangle \in F_{i-1}(\alpha(i)). \tag{10}$$

$$f \leftarrow \mathtt{dep}(p,q),\ \mathtt{dep}(q,p),\ \mathrm{not}\ f. \tag{11}$$

We demonstrate with an example why guaranteeing acyclicity in the produced models is essential for finding valid relaxed plans.

Example 3 (Cyclic Diagnostic Model)**.** *Consider the Blocksworld problem of Example 1. Note that after removing the atoms in the initial state from the goal, the new goal becomes* $G = \{on(A,C)\}$*. Allowing cyclic models in the diagnostic encoding admits the plan* $\langle \mathtt{move\text{-}b\text{-}t}(A,C),\ \mathtt{move\text{-}t\text{-}b}(A,C)\rangle$*, of cost 2, which induces a cycle of length two between* $on(A,C)$ *and* $clear(A)$*. A corresponding supported model in* P_d *contains exactly*

- ***Propositions:*** $on(A,C)$, $clear(A)$
- ***Action atoms:*** $\mathtt{move\text{-}b\text{-}t}(A,C)$, $\mathtt{move\text{-}t\text{-}b}(A,C)$
- ***Well-support atoms:*** $\mathtt{ws}(\mathtt{move\text{-}b\text{-}t}(A,C),\ clear(A))$, $\mathtt{ws}(\mathtt{move\text{-}t\text{-}b}(A,C),\ on(A,C))$

- ***Dependency atoms:*** $\texttt{dep}(clear(A),\ on(A,C))$, $\texttt{dep}(on(A,C),\ clear(A))$

However, it should be easy to check that if one only considers acyclic models, the valid relaxed plan $\langle \texttt{move-b-t}(C,B), \texttt{move-b-t}(B,A), \texttt{move-t-b}(A,C)\rangle$ *can be produced.*

3.2 Equivalent MaxSAT Encoding

To obtain a MaxSAT encoding that faithfully mirrors the supported-model semantics of P_d, one could apply Clark's completion (Clark 1977) to the logic program before translating it into CNF, enforcing that each predicate holds if and only if at least one of its defining rule bodies is satisfied, thereby eliminating unsupported classical models. However, we show that the bijection in the case of supported models also holds under the classical-model semantics of P_d: each subset of actions admitting an ordering into a relaxed plan induces an acyclic classical model in which precisely those action atoms are true, and conversely every acyclic classical model corresponds to a valid relaxed plan. This latter bijection means that P_d can readily be translated into MaxSAT in a straightforward manner to produce a MaxSAT encoding of h^+ computation.

Theorem 1. *Let $A' \subseteq A$. There is a permutation π of the actions in A' that is a relaxed plan for Π iff P_d admits a classical model $\mathcal{M}$ with $A' = \{a \in A \mid \mathcal{M} \models a\}$.*

Proof. ($\Rightarrow$) This direction is trivial, as every supported model of P_d is also a classical model of it.

($\Leftarrow$) Let $\mathcal{M}$ be a classical model of P_d. Rules (10) and (11) guarantee that $\mathcal{M}$ is acyclic, and thus, X can be ordered by topological sorting. Let α be such an order. Let $A' = \{a \in A \mid \mathcal{M} \models a\}$. We order all members of A' based on the maximum index of their preconditions in α breaking ties arbitrarily, and produce the sequence $\pi = a_1, ..., a_m$. We show that π is a relaxed plan for Π. By rules (5) and (9) all goals are produced by some action. By rules (5), (7), and (8), preconditions of actions must be produced by some other actions. We only need to show that preconditions of actions are provided when applying them. Assume the contrary: for some i, there exists a precondition p of a_i not produced by a_j for $j < i$. Then there exists some $k > i$ such that $\mathcal{M} \models ws(a_k, p)$. Assume that q is the precondition of a_k with maximum index according α. By rule (6), p must have a larger index than q according to α resulting in $k < i$, a contradiction. □

In order to translate the diagnostic ASP encoding to CNF, each normal rule is translated into an equivalent CNF clause. Since the choice rule $\{p\}\quad p \in X$ yields the tautological clause $\neg p \vee p$, it does not impose any restriction and is thus omitted. A similar argument is valid for the case of formula (4). We also use the symmetry-reduced variables

$$e^*_{p,q} = \begin{cases} \texttt{dep}(p,q) & \text{if } p \le q, \\ \neg\,\texttt{dep}(q,p) & \text{if } p > q. \end{cases}$$

By representing each unordered pair $\{p, q\}$ with a single variable $e^*_{p,q}$ the total number of dependency variables can drop from $|X|^2$ to $|X|(|X|+1)/2$. This method introduces new orders to enforce a total order on members of X. However, since the sole purpose of orders in our encoding is to prevent acyclicity, new orders can indeed be introduced as long as they do not conflict with the original partial order. Besides these modifications, the CNF clauses correspond directly to rules (5) – (9) in the logic program

$$\neg p \vee \bigvee_{a:\, p \in \mathrm{add}(a)} ws(a,p), \quad p \in X \tag{12}$$

$$\neg ws(a,p) \vee e^*_{p,q}, \quad p \in \mathrm{add}(a),\ q \in \mathrm{pre}(a),\ a \in A \tag{13}$$

$$\neg ws(a,p) \vee a, \quad p \in \mathrm{add}(a),\ a \in A \tag{14}$$

$$\neg a \vee q, \quad q \in \mathrm{pre}(a),\ a \in A \tag{15}$$

$$g, \quad g \in G \tag{16}$$

3.3 CNF Encoding of Acyclicity

Let ϕ be a propositional formula such that some atoms in it represent edges of some directed graph $G = (V, E)$. Here, we call G the induced graph of ϕ and assume that $e^*_{p,q}$ in ϕ represents the edge $\langle p, q\rangle$ in G. For any model $\mathcal{M}$ of ϕ, we define the induced graph of $\mathcal{M}$ as $G_{\mathcal{M}} = (V, E_{\mathcal{M}})$, such that $E_{\mathcal{M}} = \{\langle p, q\rangle | e^*_{p,q} \in \mathcal{M}\}$. We say that $\mathcal{M}$ is acyclic if $G_{\mathcal{M}}$ is acyclic. Our goal is to find ϕ^{acyc} such that ϕ^{acyc} encodes the acyclicity of ϕ, i. e., $\mathcal{M} \models \phi \wedge \phi^{acyc}$ if and only if $\mathcal{M}$ is an acyclic model of ϕ.

To enforce acyclicity in the MaxSAT encoding, since we use the symmetry-reduced variables, it suffices to include a single family of clauses capturing the fill-in closure:

$$\neg e^*_{p,\alpha(i)} \vee \neg e^*_{\alpha(i),q} \vee e^*_{p,q}, \quad \langle p, q\rangle \in F_{i-1}(\alpha(i)). \tag{17}$$

These clauses implement the fill-in step of the vertex-elimination process for each eliminated vertex $\alpha(i)$: whenever the dependencies $p \to \alpha(i)$ and $\alpha(i) \to q$ hold, it forces the new "fill-in" dependency $p \to q$. Conjoining these clauses for all i simulates the entire vertex elimination process and thus guarantees acyclicity without needing a separate two-cycle prohibition.

3.4 Encoding the Objective Function

Each action $a \in A$ is represented by a soft clause $\neg a$, so that minimizing the total weight of violated soft clauses corresponds directly to minimizing the cost of actions in the plan. This "action-based" objective thus uses one soft clause per action and can be rather large.

4 An Alternative Objective Representation

As an alternative to having action variables as soft clauses to encoding the objective function, we propose an objective representation which, by using extra clauses, often results in a smaller number of soft clauses as well as smaller MUSes.

Concretely, we introduce for each proposition $p \in X$ and each cost c of actions that add p, a Boolean variable $\mathrm{cost}(p, c)$, and use $\neg\,\mathrm{cost}(p, c)$ as our soft clauses instead of $\neg a$. In most planning problems $|X| \ll |A|$. For example, in the Blocksworld domain of Example 1 with n blocks there are $O(n^3)$ actions but only $O(n^2)$ propositions.

This yields far fewer soft clauses. Even more concretely, consider a simplistic case where goal is to move a specific block on another specific block, with n actions whose effects include this goal. To shift the cost accounting from actions to propositions, for each action, we have exactly one proposition responsible for representing its effect. Under the action-based objective representation, the MUSes state that at least one of the n actions must be true. In contrast, under the proposition-based representation an analogous core is unit, stating that the effect must be true. With these intuitions, the proposition-based representation expectedly most often results in smaller MUSes, which can have a positive effect on both core-guided MaxSAT solvers (Morgado, Dodaro, and Marques-Silva 2014; Martins et al. 2014; Narodytska and Bacchus 2014; Alviano, Dodaro, and Ricca 2015; Ansótegui and Gabàs 2017) (resulting in more compact reformulations of the working formulas) and implicit hitting set MaxSAT solvers (Davies and Bacchus 2013a; Davies and Bacchus 2013b; Saikko, Berg, and Järvisalo 2016) (resulting in tighter hitting set problems).

Since actions are included in the plan to prepare well-support for at least one of their effects, such effects are the natural candidates to represent the costs of the actions in the objective function. Therefore, we introduce for each proposition $p \in X$ and each cost c of actions adding p, a Boolean variable $\mathrm{cost}(p, c)$, intended to be true exactly when an action of cost c is chosen to prepare well-support for p and have p account for its cost. Concretely, for each action a with $\mathrm{add}(a) = \{p_1, \ldots, p_k\}$ (in some fixed order) and $\mathrm{cost}(a) = c$, for $i = 1..k$, we enforce

$$\neg\, ws(a, p_i) \;\vee \bigvee_{j<i} ws(a, p_j) \;\vee\; \mathrm{cost}(p_i, c). \tag{18}$$

Formula (18) guarantees that at least one of the propositions well-supported by a represents its cost in the objective function. However, a single proposition may still represent multiple actions. To prevent this, we impose mutual exclusion among the well-support atoms relevant to each proposition with the following linear encoding based on the encoding of sequential counters (Sinz 2005). For every $p \in X$, we add formulas (19)–(21), where $\{a_1, \ldots, a_m\} = \{a \mid p \in \mathrm{add}(a)\}$.

$$\neg\, ws(a_i, p) \;\vee\; visSup(a_{i+1}, p), \quad i = 1..m-1, \tag{19}$$

$$\neg\, visSup(a_i, p) \;\vee\; visSup(a_{i+1}, p),\; i = 2..m-1, \tag{20}$$

$$\neg\, visSup(a_i, p) \;\vee\; \neg\, ws(a_i, p), \quad i = 2..m. \tag{21}$$

Formula (19) states that if action a_i provides well-support for p, then the well-support for p must already have been visited before reaching a_{i+1} in the sequence $a_1, \ldots, a_m$. Formula (20) propagates the "visited" marker along the sequence, carrying it forward to each subsequent action once it appears. Formula (21) forbids any action a_i from providing well-support for p if the support has already been marked as visited upon reaching a_i, thus enforcing mutual exclusion between all pairs of actions providing well-support for p.

Let ϕ^{act} be the conjunction of formulas (12)–(17), and ϕ^{prop} be the conjunction of formulas (12)–(21). Note that while ϕ^{act} allows for multiple actions providing well-support for a single proposition, ϕ^{prop} disallows it via the mutual exclusion formulas. The proposition-based encoding remains correct as multiple well-support for a single proposition can only result in a higher value of the objective function. Since ϕ^{act} subsumes ϕ^{prop}, based on the above we have the following.

Proposition 2. *Every model of ϕ^{prop}, when restricted to the atoms of ϕ^{act}, is a model of ϕ^{act}. Further, for every model $\mathcal{M}$ of ϕ^{act} there is a model $\mathcal{M}'$ of ϕ^{prop} with*

$$\sum_{\substack{p \in X \\ \mathcal{M}' \models \mathrm{cost}(p,c)}} c \;\leq\; \sum_{\substack{a \in A \\ \mathcal{M} \models a}} \mathrm{cost}(a)\,.$$

MUSes and Landmarks. As noted by Zhang and Bacchus (2012), when representing the objective function via soft clauses over actions, unsatisfiable cores correspond precisely to *disjunctive action landmarks*, i.e., sets of actions at least one of which must be executed in every valid plan to achieve the goal (Helmert and Domshlak 2009). In contrast, with the proposition-based objective representation, cores can be interpreted as *disjunctive fact landmarks*, i.e., sets of fact at least one of which must hold at some point in any solution plan (Hoffmann, Porteous, and Sebastia 2004). In classical planning, disjunctive action landmarks often encompass a very large number of alternatives, reflecting diverse ways to satisfy certain preconditions or effects (Büchner et al. 2023). By comparison, disjunctive fact landmarks typically consist of only a few facts: for example, the LAMA planner's landmark generator limits each disjunctive fact landmark to at most four facts with the intuition that smaller disjunctions represent stronger necessary conditions (Richter, Helmert, and Westphal 2008).

5 Alternative Encoding of Acyclicity

We base our new encoding on the encoding of the transitive closure graph. The transitive closure graph of a directed graph $G = \langle V, E \rangle$ is the graph $G^+ = \langle V, E^+ \rangle$, where $E^+ = \{\langle u, v \rangle \mid \text{there exists a non-empty path from } u \text{ to } v \text{ in } G\}$. The transitive closure graph of a directed graph captures all reachability relationships in G, making it a comprehensive representation of the graph's connectivity.

If there exists a cycle $v_1, v_2, \ldots, v_k, v_1$ in G, then G^+ will include edges $\langle v_i, v_j \rangle$ and $\langle v_j, v_i \rangle$ for all i and j in the cycle because each vertex is reachable from every other vertex in the cycle through paths in G. On a side note, while G^+ may also be defined with self-loops for each v_i in the cycle (as each vertex is reachable from itself through a non-empty path), we focus here on cycles of length two because they form the basis for cycle detection in the transitive closure-based encoding, as explained next.

5.1 Transitive Closure-Based Encoding

We start by reviewing an encoding based on the transitive closure method as employed e.g. in (Cussens 2008). As in the case of the vertex elimination based encoding, we also use variables $e^*_{x,y}$ to reduce symmetry in its representation. If there is an edge in G from p to q, and an edge in G^+ from

q to r, then there must be an edge from p to r in G^+:

$$\neg e^*_{p,q} \vee \neg e^*_{q,r} \vee e^*_{p,r}, \quad \langle p,q\rangle \in E, \langle q,r\rangle \in E^+. \quad (22)$$

This formula ensures that all paths in G are correctly reflected in edges of G^+. Due to the reduced symmetry in the representation, achieved by distinguishing $e^*_{p,q}$ from $\mathtt{dep}(p,q)$, there is no need to separately detect cycles of length two, as such cycles are inherently prohibited. Encoding the entire transitive closure of a graph can be expensive as it includes reachability information for all pairs of vertices even when unnecessary for detecting cycles. A straightforward analysis shows that the encoding requires $\mathcal{O}(|V|^2)$ propositional atoms and $\mathcal{O}(|V||E|)$ clauses. This motivates the development of alternative, more succinct methods for encoding acyclicity to avoid the overhead associated with constructing G^+ in its entirety.

5.2 A Novel Cycle Elimination based Encoding

Let ϕ be a propositional formula with $G = (V, E)$ as its induced directed graph and $G^+ = (V, E^+)$. The *transpose* (or *reverse graph*) G^T is defined as $G^T = (V, E^T)$, where $E^T = \{(u,v) \mid (v,u) \in E\}$. Here, we use G^T mainly as an illustrational tool to describe how, given a vertex p, one can construct a formula that eliminates all cycles going through p. To encode acyclicity, instead of encoding the transitive closure of G, one may encode the transitive closure of G^T by the method described above. Replacing G with G^T in formulas (22), we construct ϕ^+ as the encoding of the transitive closure graph of G^T. Formally, after due index renaming, we can express ϕ^+ as $\bigwedge_{p\in V} \phi^G_p$, where ϕ^G_p is the conjunction of

$$\neg e^*_{p,q} \vee \neg e^*_{q,r} \vee e^*_{p,r}, \quad \langle p,q\rangle \in E^+, \langle q,r\rangle \in E. \quad (23)$$

A key observation here is that for any model $\mathcal{M}$ of $\phi \wedge \phi^G_p$, and every outgoing edge $\langle p,q\rangle$ of p in $G^+_{\mathcal{M}}$, we must have $\mathcal{M} \models e^*_{p,q}$. This automatically eliminates all cycles including p from all possible models.

Lemma 3. *Let $u \in V$, $\mathcal{M}$ be a model for $\phi \wedge \phi^G_u$, $G_{\mathcal{M}} = (V, E_{\mathcal{M}})$ the induced graph of $\mathcal{M}$, and $G^+_{\mathcal{M}} = (V, E^+_{\mathcal{M}})$ the transitive closure of $G_{\mathcal{M}}$. If $\langle u,v\rangle \in E^+_{\mathcal{M}}$ then $\mathcal{M} \models e^*_{u,v}$.*

Proof. If $\langle u,v\rangle \in E^+_{\mathcal{M}}$, then there must exist a path $u = u_0, u_1, \ldots, u_k = v$ in $E_{\mathcal{M}}$. By induction on k, we show that $\mathcal{M} \models e^*_{u,v}$. For $k = 1$, the claim holds trivially. Assume that the claim holds for $k = m$. We show that it also holds for $k = m+1$. Let $u = u_0, u_1, \ldots, u_{m+1} = v$ be a path in $E_{\mathcal{M}}$. Then clearly $u = u_0, u_1, \ldots, u_m$ is a path in $E_{\mathcal{M}}$, and by induction hypothesis, $\mathcal{M} \models e^*_{u,u_m}$. Since $\mathcal{M} \models e^*_{u_m,v}$, by (23) we have $\mathcal{M} \models e^*_{u,v}$. □

We immediately have that ϕ^+ encodes acyclicity of ϕ.

Proposition 4. *Every model of $\phi \wedge \phi^+$, when restricted to the atoms of ϕ, is an acyclic model of ϕ. Moreover, every acyclic model of ϕ can be extended to a model of $\phi \wedge \phi^+$.*

Intuitively, one can think of ϕ^+ as eliminating cycles "vertex by vertex": for each v, ϕ^G_v rules out all cycles containing v, and since every cycle must visit some vertex, the conjunction over all v removes all cycles. However, one may observe that once all cycles containing v have been eliminated, there will be no point in considering edges adjacent to v when eliminating further cycles.

Recall that a strongly connected component (SCC) of a directed graph is a maximal subgraph in which every vertex is reachable from every other vertex. Since no cycle can span multiple SCCs, cycle detection, and therefore cycle elimination, can be performed independently within each SCC. However, with edges adjacent to v having been eliminated, the SCCs of G containing v may no longer be strongly connected, and the newly found SCCs might be smaller than before. Based on these intuitions, we introduce the cycle elimination process of G.

For a given graph G, let the cycle elimination fill-in of vertex v be the set of all incoming edges of v in G^+. Given an ordering α on vertices, let $G = G^{ce}_0, G^{ce}_1, \ldots, G^{ce}_{n-1}$ be a series of graphs such that for each $i = 1, \ldots, n-1$, the graph G^{ce}_i is produced by first removing $\alpha(i)$ and all its adjacent edges from G^{ce}_{i-1}, and then, removing all edges going out of any strongly connected components. The cumulative fill-in with respect to α is defined as $F^{ce}_\alpha(G) = \bigcup_{i=1}^{n-1} F^{ce}_{i-1}(\alpha(i))$, with $F^{ce}_{i-1}(\alpha(i))$ denoting the cycle elimination fill-in of vertex $\alpha(i)$ in G^{ce}_{i-1}. Finally, the cycle elimination graph corresponding to the ordering α is given by $G^+_\alpha = \langle V, F^{ce}_\alpha(G)\rangle$.

Note that G^+_α is trivially a subgraph of G^+. However, G^+_α can be a strict subgraph of G^+. That is because $F^{ce}_{i-1}(\alpha(i))$ can be a strict subset of the set of incoming edges of $\alpha(i)$ in G^+. Despite this, we show that, G^+_α preserves the acyclicity properties of G regardless of ordering α,

Lemma 5. *If G has a cycle, G^+_α has a cycle of length two.*

Proof. Let $u_1, u_2, \ldots, u_k, u_1$ be a cycle in G, and $u_1 = \alpha(m)$. Without loss of generality, we assume that u_1 is ordered before all other vertices of the cycle. Therefore the cycle must exist in G^{ce}_{m-1}. Therefore, the edge $\langle u_1, u_k\rangle$ must exist in $F^{ce}_{m-1}(u_1)$. We conclude that G^+_α has the cycle u_1, u_k, u_1 of length two. □

Lemma 5 shows that to encode acyclicity, one only needs to encode the cycle elimination process, with cycles of length two inherently being prevented by the definition of atoms $e^*_{x,y}$. Let ϕ^+_α be defined as

$$\bigwedge_{i=1..|V|} \phi^{G^{ce}_{i-1}}_{\alpha(i)}.$$

We show that ϕ^+_α encodes the acyclicity of ϕ.

Theorem 6. *Every model of $\phi \wedge \phi^+_\alpha$, when restricted to the atoms of ϕ, is an acyclic model of ϕ. Moreover, every acyclic model of ϕ can be extended to a model of $\phi \wedge \phi^+_\alpha$.*

Proof. Let $\mathcal{M}^+$ be a model of $\phi \wedge \phi^+_\alpha$, $\mathcal{M}$ be the restriction of $\mathcal{M}^+$ to the atoms of ϕ, and $G^+_{\mathcal{M}}$ and $G_{\mathcal{M}}$ be the induced graph of $\mathcal{M}^+$ and $\mathcal{M}$, respectively. Assume that $\mathcal{M}$ is cyclic. Lemma 3 shows that $G^+_{\mathcal{M}}$ is the cycle elimination graph of $G_{\mathcal{M}}$ corresponding to the ordering α. Therefore,

by Lemma 5, $G^+_{\mathcal{M}}$ has a cycle of length two, which is a contradiction given the definition of the e^* atoms.

On the other hand, assume that $\mathcal{M}$ is an acyclic model of ϕ. By Proposition 4, $\mathcal{M}$ can be extended to a model of $\phi \wedge \phi^+$. However, by definition, ϕ^+_α subsumes ϕ^+. We conclude that $\mathcal{M}$ can be extended to a model of $\phi \wedge \phi^+_\alpha$. □

Although the worst-case size complexity of ϕ^+_α remains $\mathcal{O}(|V||E|)$, the cycle-elimination process considers the sequence of graphs $G^{ce}_1, \ldots, G^{ce}_{n-1}$, where each G^{ce}_i is obtained by removing a single vertex from G^{ce}_{i-1}. As i increases, the strongly connected components of G^{ce}_i become both more numerous and smaller, which in practice often yields significantly more compact encodings.

Example 4. *Let $G = G^{ce}_0$ be the graph in Figure 1a. The transitive closure G^+ of G is shown in Figure 1c. For cycle elimination, assume that α orders the vertices by their label. The cycle elimination fill-in of vertex* 1 *is the incoming edges of* 1 *in G^+, which can be seen added to G in Figure 1d. To produce G^{ce}_1, note that after removing* 1 *from G, the resulting graph is acyclic. Therefore, G^{ce}_1 has five singleton strongly connected components and no edges, and the cycle elimination process adds no further edges to Figure 1d. The cycle elimination graph has the cycle* $1, 2, 1$ *of length two.*

6 Empirical Evaluation

We empirically evaluate our MaxSAT approaches to computing h^+. Our goals are to (i) evaluate the relative performance of state-of-the-art MaxSAT solvers against the ASP-based approach representing the current state of the art for computing h^+; and (ii) evaluate the relative performance of our MaxSAT encoding variants, including combinations of the encodings in both sequential and parallel settings. We integrated the encodings into the HSP* planner (Haslum 2015). The implementation is publicly available at https://bitbucket.org/coreo-group/opth. We use the STRIPS planning benchmarks from the *planning repository* (https://github.com/AI-Planning/classical-domains) used in the previous works proposing computing h^+ via SAT, integer linear programming, and ASP, and also both the optimal and so-called *satisficing* tracks of IPC 2023. As the ASP approach has been shown to outperform the other approaches on the same benchmark set we focus on comparing our MaxSAT approaches to the ASP approach. Following (Rankooh and Janhunen 2023), we use version Clasp 3.3.10 as the solver in its core-guided optimization mode (Andres et al. 2012). The experiments were run on 2.50-GHz Intel Xeon Gold 6248 CPUs under per-instance and per-solver 1800-second time and 16 GB memory limits.

MaxSAT vs ASP on Action-Based Encoding. First, we analyze the relative performance of various modern MaxSAT solvers and Clasp on the ASP and MaxSAT formulations of the action-based objective representation using the vertex elimination (VE) encoding of acyclicity. In particular, this is the encoding variant shown to perform best using ASP (Rankooh and Janhunen 2023). We consider four modern MaxSAT solvers as state-of-the-art representatives of solvers implementing different types of MaxSAT algorithms: MaxHS (Davies and Bacchus 2013a; Davies and Bacchus 2013b) (version 5.0.0) implementing the implicit hitting set approach to MaxSAT based on iteratively computing cores and hitting sets of the so-far found cores; Pacose (Paxian, Reimer, and Becker 2018) (version MSE2024) implementing the solution-improving approach based on iteratively querying a SAT solver for better solutions; UWrMaxSAT (Piotrów 2020) (version 1.7.0) implementing the core-guided approach in terms of the OLL (Andres et al. 2012; Morgado, Dodaro, and Marques-Silva 2014) algorithm; and WMaxCDCL (Li et al. 2022; Li et al. 2025) (version MSE2024) implementing a clause learning boosted branch-and-bound approach to MaxSAT.

Figure 2 shows the number of instances solved (y-axis) by each solver under different per-instance time limits (x-axis). We observe that on this encoding variant, Clasp outperforms the MaxSAT solvers up to a per-instance time limit of approximately 30 seconds, beyond which UWrMaxSAT outperforms the other solvers. The solution-improving approach (Pacose) and implicit hitting set approach (MaxHS) exhibit weakest performance. Among the MaxSAT solvers, The core-guided UWrMaxSAT solver outperforms the other solvers consistently. Based on this, we fix UWrMaxSAT as the MaxSAT solver for the remaining experiments.

Core extraction is the main source of empirical complexity in the core-guided approach implemented in UWrMaxSAT. In principle, the solver would benefit from computing a subset-minimal unsatisfiable core at each iteration, as the sizes of cores computed directly translates into a blow-up of the working formula at each iteration. However, in practice core minimization cannot always be performed exhaustively, as the minimization requires additional SAT solver calls and decreases the overall performance of the solver especially because it forces the SAT solver to prove minimality. This is why UWrMaxSAT includes a hard-coded conflict limit (of 500 conflicts) for each SAT solver call within the core minimization routine. However, we observed that this does not reflect well actual time spent in core minimization on the heterogeneous planning benchmark set. In particular, by enforcing a wall-clock limit on the whole core-minimization routing at each iteration resulted in noticeably better improvements; see Figure 3 where the default UWrMaxSAT limit is compared to a 0.1-second limit. For the remaining experiments, we employ UWrMaxSAT with 0.1-second per-iteration time limit for core minimization. We emphasize that, beyond observing that 0.1 seconds and 0.01 yielded similar performance, improving somewhat over a 1-second limit, we did not attempt to further optimize the value of this parameter for performance—it is likely that more fine-grained strategies for choosing this parameter could result in further performance improvements.

Impact of Objective Representations. We now turn to evaluating the impact of the choice of objective representation (action-based vs proposition-based) on MaxSAT solver performance, using the vertex elimination encoding of acyclity in both cases. The action-based MaxSAT encoding (see Figure 4) dominates the proposition-based ASP encoding,

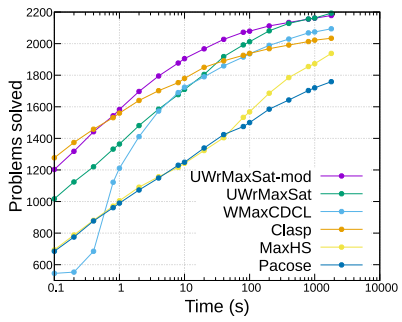


Figure 2: MaxSAT vs Clasp on action encoding and acyclicity by VE

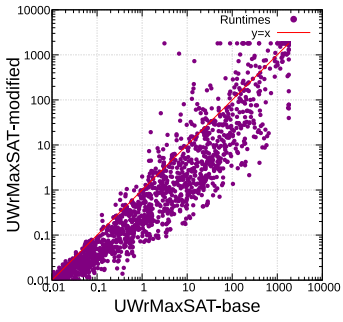


Figure 3: Impact of core minimization limit on UWrMaxSAT: default (x) vs 0.1 s (y)

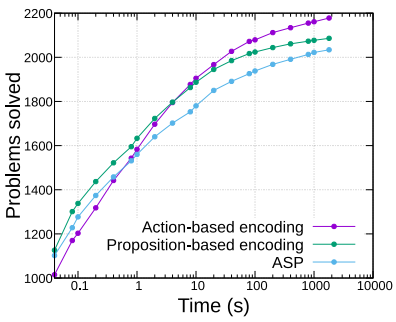


Figure 4: Action vs proposition-based MaxSAT encoding vs action-based ASP

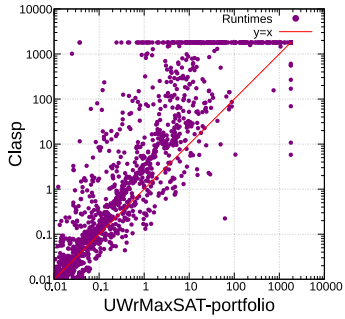


Figure 5: Action and proposition-based MaxSAT in parallel vs two-threaded Clasp

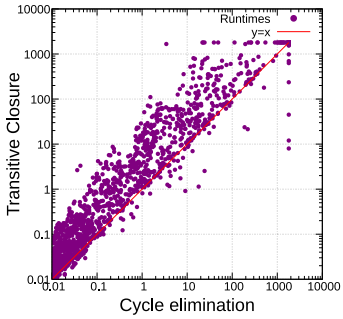


Figure 6: Cycle elimination vs transitive closure

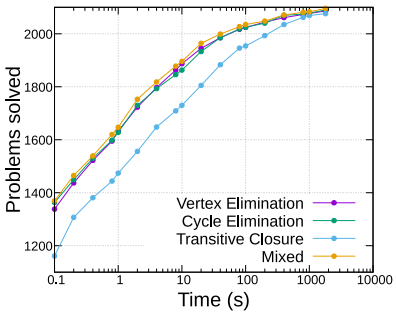


Figure 7: Impact of acyclicity encodings

with more instances solved using the MaxSAT solver at any per-instance time limit. The same holds for the action-based encoding solved with MaxSAT for time limit > 1 second.

Interestingly, the proposition-based encoding yields better performance for time limits less that ≈ 5 seconds, and the action-based encoding yields better performance for time limits > 5 seconds. This motivates investigating whether the two encodings can be used for further overall runtime improvements across different time limits. For this, we run the MaxSAT solver on both encodings in parallel, allocating one core for each of the encodings, terminating both cores when the first of the two terminates. This straightfoward approach turns out to significantly outperform running Clasp in parallel mode on two threads (as directly offered by Clasp) on the action-based ASP encoding; see Figure 5.

Impact of Acyclicity Encodings. Finally, we investigate the impact of the acyclicity encodings on MaxSAT performance. We present results here using the proposition-based encoding; the results under the action-based encoding are similar. Figure 6 shows pair-wise comparisons of MaxSAT solver performance under cycle elimination based against transitive closure based acyclicity. Cycle elimination is consistently more effective than transitive closure, showing that it, as a refinement of the transitive closure based encoding, indeed yields better performance. Figure 7 shows the number of instances solved (y-axis) for different time limits (x-axis) for each of the encodings. While noticeably different in terms of the clauses the encodings introduce, overall performance using either cycle elimination or vertex elimination is quite similar. Here we also include a straightforward "mixed encoding", which refers to making an instance-specific decision on whether to use cycle elimination or vertex elimination by choosing the encoding resulting in fewer clauses. While the size of SAT encodings does not necessarily correlate with MaxSAT solver performance in practice, this simplistic heuristic improves on the two individual encodings in terms of the number of instances solved for time limits < 1 second; such fast runtimes are of interest in particular towards employing the MaxSAT-based approach to computing h^+ as heuristic guidance repeatedly in state-space planners.

7 Conclusions

We proposed MaxSAT for cost-optimal delete-free classical planning. We detailed a MaxSAT encoding analogous to a state-of-the-art ASP approach, and proposed the proposition-based encoding of cost-optimality which can result in a smaller numbers of soft clauses and thereby allow MaxSAT solvers leverage smaller unsatisfiable cores during search. We also proposed an alternative cycle elimination encoding for plan acyclicity. A state-of-the-art core-guided MaxSAT solver using the proposition-based encoding outperforms the ASP approach. Further performance improvements (also over multi-threaded ASP solving) are achieved by solving in parallel the action-based and the proposition-based encodings. The new cycle elimination based encoding of acyclicity may be of interest also beyond planning.

Acknowledgements

The work is financially supported by Research Council of Finland under grants 347588 (AN) and 356046 (MFR, MJ), and Helsinki Institute for Information Technology HIIT (MFR). The authors wish to thank the Finnish Computing Competence Infrastructure (FCCI) for supporting this project with computational and data storage resources.

References

Alviano, M.; Dodaro, C.; and Ricca, F. 2015. A MaxSAT algorithm using cardinality constraints of bounded size. In Yang, Q., and Wooldridge, M. J., eds., *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, 2677–2683. AAAI Press.

Andres, B.; Kaufmann, B.; Matheis, O.; and Schaub, T. 2012. Unsatisfiability-based optimization in clasp. In Dovier, A., and Costa, V. S., eds., *Technical Communications of the 28th International Conference on Logic Programming, ICLP 2012, September 4-8, 2012, Budapest, Hungary*, volume 17 of *LIPIcs*, 211–221. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Ansótegui, C., and Gabàs, J. 2017. WPM3: an (in)complete algorithm for weighted partial MaxSAT. *Artif. Intell.* 250:37–57.

Bacchus, F.; Järvisalo, M.; and Martins, R. 2021. Maximum satisfiability. In Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds., *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*. IOS Press. 929–991.

Betz, C., and Helmert, M. 2009. Planning with h^+ in theory and practice. In *KI 2009: Advances in Artificial Intelligence, 32nd Annual German Conference on AI, Paderborn, Germany, September 15-18, 2009.*, volume 5803 of *Lecture Notes in Computer Science*, 9–16. Springer.

Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds. 2021. *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*. IOS Press.

Bonet, B., and Geffner, H. 1999. Planning as heuristic search: New results. In Biundo, S., and Fox, M., eds., *Recent Advances in AI Planning, 5th European Conference on Planning, ECP'99, Durham, UK, September 8-10, 1999, Proceedings*, volume 1809 of *Lecture Notes in Computer Science*, 360–372. Springer.

Büchner, C.; Christen, R.; Eriksson, S.; and Keller, T. 2023. DALAI – disjunctive action landmarks all in. In *Proceedings of the International Planning Competition 2023 (IPC-2023) – Planner Abstracts*.

Bylander, T. 1994. The computational complexity of propositional STRIPS planning. *Artif. Intell.* 69(1-2):165–204.

Clark, K. L. 1977. Negation as failure. In Gallaire, H., and Minker, J., eds., *Logic and Data Bases, Symposium on Logic and Data Bases, Centre d'études et de recherches de Toulouse, France, 1977*, Advances in Data Base Theory, 293–322. New York: Plemum Press.

Cussens, J. 2008. Bayesian network learning by compiling to weighted MAX-SAT. In McAllester, D. A., and Myllymäki, P., eds., *UAI 2008, Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence, Helsinki, Finland, July 9-12, 2008*, 105–112. AUAI Press.

Davies, J., and Bacchus, F. 2013a. Exploiting the power of MIP solvers in MaxSAT. In Järvisalo, M., and Gelder, A. V., eds., *Theory and Applications of Satisfiability Testing - SAT 2013 - 16th International Conference, Helsinki, Finland, July 8-12, 2013. Proceedings*, volume 7962 of *Lecture Notes in Computer Science*, 166–181. Springer.

Davies, J., and Bacchus, F. 2013b. Postponing optimization to speed up MaxSAT solving. In Schulte, C., ed., *Principles and Practice of Constraint Programming - 19th International Conference, CP 2013, Uppsala, Sweden, September 16-20, 2013. Proceedings*, volume 8124 of *Lecture Notes in Computer Science*, 247–262. Springer.

Gefen, A., and Brafman, R. I. 2011. The minimal seed set problem. In *Proceedings of the 21st International Conference on Automated Planning and Scheduling, ICAPS 2011, Freiburg, Germany June 11-16, 2011*. AAAI.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Logic Programming, Proceedings of the Fifth International Conference and Symposium, Seattle, Washington, USA, August 15-19, 1988 (2 Volumes)*, 1070–1080.

Ghallab, M.; Nau, D. S.; and Traverso, P. 2016. *Automated Planning and Acting*. Cambridge, UK: Cambridge University Press.

Haslum, P. 2012. Incremental lower bounds for additive cost planning problems. In *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling, ICAPS 2012, Atibaia, São Paulo, Brazil, June 25-19, 2012*. AAAI.

Haslum, P. 2015. HSP* code and documentation http://users.cecs.anu.edu.au/patrik/un-hsps.html.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In *Proceedings of the 19th International Conference on Automated Planning and Scheduling, ICAPS 2009, Thessaloniki, Greece, September 19-23, 2009*. AAAI.

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered landmarks in planning. *J. Artif. Intell. Res.* 22:215–278.

Hoffmann, J. 2005. Where "ignoring delete lists" works: Local search topology in planning benchmarks. *Journal of Artificial Intelligence Research* 24:685–758.

Imai, T., and Fukunaga, A. 2015. 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* 54:631–677.

Kautz, H. A., and Selman, B. 1999. Unifying SAT-based and graph-based planning. In Dean, T., ed., *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 pages*, 318–325. Morgan Kaufmann.

Li, C.; Xu, Z.; Coll, J.; Manyà, F.; Habet, D.; and He, K. 2022. Boosting branch-and-bound MaxSAT solvers with clause learning. *AI Commun.* 35(2):131–151.

Li, S.; Li, C.; Coll, J.; Habet, D.; and Manyà, F. 2025. Improving the lower bound in branch-and-bound algorithms for MaxSAT. In Walsh, T.; Shah, J.; and Kolter, Z., eds., *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, 11272–11281. AAAI Press.

Martins, R.; Joshi, S.; Manquinho, V.; and Lynce, I. 2014. Incremental cardinality constraints for MaxSAT. In O'Sullivan, B., ed., *Principles and Practice of Constraint Programming - 20th International Conference, CP 2014, Lyon, France, September 8-12, 2014. Proceedings*, volume 8656 of *Lecture Notes in Computer Science*, 531–548. Springer.

Mirkis, V., and Domshlak, C. 2007. Cost-sharing approximations for h^+. In *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling, ICAPS 2007, Providence, Rhode Island, USA, September 22-26, 2007*, 240–247. AAAI.

Morgado, A.; Dodaro, C.; and Marques-Silva, J. 2014. Core-guided MaxSAT with soft cardinality constraints. In O'Sullivan, B., ed., *Principles and Practice of Constraint Programming - 20th International Conference, CP 2014, Lyon, France, September 8-12, 2014. Proceedings*, volume 8656 of *Lecture Notes in Computer Science*, 564–573. Springer.

Narodytska, N., and Bacchus, F. 2014. Maximum satisfiability using core-guided MaxSAT resolution. In Brodley, C. E., and Stone, P., eds., *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 - 31, 2014, Québec City, Québec, Canada*, 2717–2723. AAAI Press.

Niemelä, I. 1999. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence* 25(3-4):241–273.

Paxian, T.; Reimer, S.; and Becker, B. 2018. Dynamic polynomial watchdog encoding for solving weighted MaxSAT. In Beyersdorff, O., and Wintersteiger, C. M., eds., *Theory and Applications of Satisfiability Testing - SAT 2018 - 21st International Conference, SAT 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 9-12, 2018, Proceedings*, volume 10929 of *Lecture Notes in Computer Science*, 37–53. Springer.

Piotrów, M. 2020. UwrMaxSAT: Efficient solver for MaxSAT and pseudo-boolean problems. In *32nd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2020, Baltimore, MD, USA, November 9-11, 2020*, 132–136. IEEE.

Rankooh, M. F., and Janhunen, T. 2023. Capturing (optimal) relaxed plans with stable and supported models of logic programs. *Theory Pract. Log. Program.* 23(4):782–796.

Rankooh, M. F., and Rintanen, J. 2022a. Efficient computation and informative estimation of h^+ by integer and linear programming. In *Proceedings of the Thirty-Second International Conference on Automated Planning and Scheduling, ICAPS 2022, Singapore (virtual), June 13-24, 2022*, 71–79. AAAI Press.

Rankooh, M. F., and Rintanen, J. 2022b. Efficient encoding of cost optimal delete-free planning as SAT. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Virtual Event, February 22 - March 1, 2022*, 9910–9917. AAAI Press.

Rankooh, M. F., and Rintanen, J. 2022c. Propositional encodings of acyclicity and reachability by using vertex elimination. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, 2022 Virtual Event, February 22 - March 1, 2022*, 5861–5868. AAAI Press.

Richter, S.; Helmert, M.; and Westphal, M. 2008. Landmarks revisited. In Fox, D., and Gomes, C. P., eds., *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, 975–982. AAAI Press.

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

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

Rose, D. J., and Tarjan, R. E. 1975. Algorithmic aspects of vertex elimination. In Rounds, W. C.; Martin, N.; Carlyle, J. W.; and Harrison, M. A., eds., *Proceedings of the 7th Annual ACM Symposium on Theory of Computing, May 5-7, 1975, Albuquerque, New Mexico, USA*, 245–254. ACM.

Russell, S. J., and Norvig, P. 2010. *Artificial Intelligence - A Modern Approach, Third International Edition*. Pearson Education.

Saikko, P.; Berg, J.; and Järvisalo, M. 2016. LMHS: A SAT-IP hybrid MaxSAT solver. In Creignou, N., and Berre, D. L., eds., *Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference, Bordeaux, France, July 5-8, 2016, Proceedings*, volume 9710 of *Lecture Notes in Computer Science*, 539–546. Springer.

Sinz, C. 2005. Towards an optimal CNF encoding of boolean cardinality constraints. In van Beek, P., ed., *Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings*, volume 3709 of *Lecture Notes in Computer Science*, 827–831. Springer.

Suda, M. 2014. Property directed reachability for automated planning. *J. Artif. Intell. Res.* 50:265–319.

Zhang, L., and Bacchus, F. 2012. MaxSAT heuristics for cost optimal planning. In Hoffmann, J., and Selman, B., eds., *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada*, 1846–1852. AAAI Press.