Partial Weighted MaxSAT for Optimal Planning

Nathan Robinson[†], Charles Gretton[‡], Duc Nghia Pham[†], and Abdul Sattar[†]

[†] ATOMIC Project, Queensland Research Lab, NICTA and
Institute for Integrated and Intelligent Systems, Griffith University, QLD, Australia
[‡] School of Computer Science, University of Birmingham

Abstract. We consider the problem of computing optimal plans for propositional planning problems with action costs. In the spirit of leveraging advances in general-purpose automated reasoning for that setting, we develop an approach that operates by solving a sequence of *partial weighted MaxSAT* problems, each of which corresponds to a step-bounded variant of the problem at hand. Our approach is the first SAT-based system in which a proof of cost-optimality is obtained using a MaxSAT procedure. It is also the first system of this kind to incorporate an admissible planning heuristic. We perform a detailed empirical evaluation of our work using benchmarks from a number of International Planning Competitions.

1 Introduction

Recently there have been significant advances in the direction of optimal planning procedures that operate by making multiple queries to a decision procedure, usually a Boolean SAT procedure. For example, the work of Hoffman *et al.* [1] answers a key challenge from Kautz [2] by demonstrating how existing SAT-based planning techniques can be made effective solution procedures for fixed-horizon planning with metric resource constraints. In the same vein, Russell & Holden [3] and Giunchiglia & Maratea [4] develop optimal SAT-based procedures for *net-benefit* planning in fixed-horizon problems. In that case actions can have costs and goal utilities can be interdependent. Moreover, in the direction of improving the scalability and efficiency of SAT-based approaches in step-optimal (and indeed fixed-horizon) planning, Robinson *et al.* [5] presents an encoding of step-bounded planning problems that shows significant performance gains over previous results. Large performance gains have also been demonstrated where efficient and sophisticated query strategies are employed [6, 7]. Summarising, in the settings of step-optimal and fixed-horizon planning, recent works have demonstrated that SAT-based techniques inspired by systems like BLACKBOX [8] continue to dominate other approaches.

Considering the planning literature more generally, numerous distinct criteria for plan optimality have been proposed. These include: (1) Minimise *makespan* (a.k.a. *step-optimality*); The objective is to find a plan of minimal length. (2) Minimise *plan cost*; Each action has a numeric cost, a plan's cost is the sum of the costs of its constituent actions, and an optimal plan has minimal cost. (3) Maximise *net-benefit*; States (resp. actions) have rewards (resp. costs), and an optimal plan is a sequence of actions executable from the starting state that induces a behaviour of maximal *utility* – These

problems are sometimes called *oversubscribed*, and were recently shown to be equivalent (using a compilation) to the cost-optimising setting [9]. One key observation to be made is that the above optimality criteria are often conflicting. For example, a plan with minimal *makespan* is not guaranteed to be *cost-* or *utility*-optimal. Indeed, in the general case there is no link between the number of plans steps (planning horizon) and plan quality.

Existing SAT-based planning procedures are limited to *makespan*-optimal and *fixed-horizon* settings – i.e., either the objective is to minimise the number of plan-steps, or valid optimal solutions are constrained to be of, or less than, a fixed length. Thus, the use of SAT-based techniques is limited in practice. For example, optimal SAT-based planning procedures were unable to participate effectively at the International Planning Competition (IPC) in 2008 due to the adoption of a single optimisation criteria (cost-optimality). This paper overcomes that restriction, developing COS-P, the fist sound and complete cost-optimal planning procedure based solely on a Boolean SAT(isfiability) procedure. Thus, we open the door to leveraging SAT technology in planning settings with arbitrary optimisation criteria.

The remainder of this paper is organised as follows. We first give an overview of optimal propositional planning with action costs, delete relaxations of that problem, and the partial weighted MaxSAT optimisation problem. We then describe our approach in detail, developing compilations to partial weighted MaxSAT of the fixed-horizon planning problem, and of the fixed horizon problem with a relaxed suffix. Following this we develop our novel MaxSAT solution procedure PWM-RSAT. We then empirically evaluate our approach on planning benchmarks from a number of IPCs. Finally we discuss some related work and propose some interesting directions for future research.

2   Background and Notations

2.1   Propositional planning with action costs

A propositional planning problem with costs is a 5-tuple $\Pi = \langle P, \mathcal{A}, s_0, \mathcal{G}, \mathcal{C} \rangle$. Here, P is a set of propositions that characterise problem states; $\mathcal{A}$ is the set of actions that can induce state transitions; $s_0 \subseteq P$ is the starting state; And $\mathcal{G} \subseteq P$ is the set of propositions that characterise the goal. The function $\mathcal{C} : \mathcal{A} \to \Re_0^+$ is a bounded cost function that assigns a non-negative cost-value to each action. This value corresponds to the cost of executing the action.

Each action $a \in \mathcal{A}$ is described in terms of its preconditions $pre(a) \subseteq P$, positive effects $eff_\bullet(a) \subseteq P$, and negative effects $eff_\circ(a) \subseteq P$. An action a can be executed at a state $s \subseteq P$ when $pre(a) \subseteq s$. We write $\mathcal{A}(s)$ for the set of actions that can be executed at state s – Formally, $\mathcal{A}(s) \equiv \{a | a \in \mathcal{A}, pre(a) \subseteq s\}$. When $a \in \mathcal{A}(s)$ is executed at s the successive state is $(s \cup eff_\bullet(a)) \backslash eff_\circ(a)$. Actions cannot both add and delete the same proposition – i.e., $eff_\bullet(a) \cap eff_\circ(a) \equiv \emptyset$.[1] A state s is a *goal state* iff $\mathcal{G} \subseteq s$.

Usually any two actions $a_1, a_2 \in \mathcal{A}$ are permitted to be executed instantaneously in parallel at a state provided any serial execution of the actions is valid and achieves an identical outcome. When two actions cannot be executed in parallel we say they

---

[1] In practice this case is given a special semantics, the details of which shall not be considered further here.

conflict. Supposing non-conflicting actions can be executed instantaneously in parallel, a *plan* π is a discrete sequence of time-indexed sets of non-conflicting actions which, when applied to the start state, lead to a goal state. We say a plan is *serial* (a.k.a. *linear plan*), denoted $\boldsymbol{\pi}$, if each time-indexed set contains one action. Finally, where $\mathcal{A}^i$ is the set of actions at step i of $\pi = [\mathcal{A}^1, \mathcal{A}^2, .., \mathcal{A}^h]$, the cost of π, written $\mathcal{C}(\pi)$, is:

$$\mathcal{C}(\pi) = \sum_{i=1}^{h} \sum_{a \in \mathcal{A}^i} \mathcal{C}(a)$$

A number of different conditions for plan optimality can be defined. In particular, a plan is *parallel step-optimal* if no shorter plan of the same parallel format exists. The definition for *serial step-optimality* is identical, but also respects the condition that a valid plan has only one action executed at each step. A plan π^* is *cost-optimal* if there is no plan π s.t. $\mathcal{C}(\pi) < \mathcal{C}(\pi^*)$. Finally, we draw the reader's attention to the fact that the definition of cost-optimality is not dependent on the plan format.

2.2 The relaxed planning problem

A *delete relaxation* Π^+ of a planning problem Π is an equivalent problem in all respects except the definition of actions. In particular, the set of actions $\mathcal{A}^+$ in Π^+ comprises the elements $a \in \mathcal{A}$ from Π altered so that $\mathit{eff}_\circ(a) \equiv \emptyset$. The relaxed problem has two key properties of interest here. First, the cost of an optimal plan from any reachable state in Π is greater than or equal to the cost of the optimal plan from that state in Π^+. Consequently relaxed planning can yield a useful admissible heuristic in search. For example, a best-first search such as A^* can be heuristically directed towards an optimal solution by using the costs of relaxed plans to arrange the priority queue. Second, although NP-hard to solve optimally in general [10], in practice optimal solutions to the relaxed problem Π^+ are more easily computed than for Π.

2.3 Partial weighted MaxSAT

A Boolean SAT problem is a decision problem, instances of which are typically expressed as a CNF propositional formula. A CNF corresponds to a conjunction over clauses, each of which corresponds to a disjunction over literals. A literal is either a proposition (i.e., Boolean variable symbol) or its negation. Where $\models$ denotes semantic entailment for propositional logic, a solution associated with a formula ϕ is an assignment (a.k.a. valuation) $\mathcal{V}$ of truth values to propositions with the property $\mathcal{V} \models \phi$.

A Boolean MaxSAT problem is an optimisation problem related to SAT. In practice a problem instance is again typically expressed as a CNF, however the objective now is to compute a valuation that maximises the number of satisfied clauses. In detail, writing $\kappa \in \phi$ if κ is a clause in formula ϕ, and taking $\mathcal{V} \models \kappa$ to have numeric value 1 when valid, and 0 otherwise, a solution $\mathcal{V}^*$ to a MaxSAT problem has the property:

$$\mathcal{V}^* = \arg\max_{\mathcal{V}} \sum_{\kappa \in \phi} (\mathcal{V} \models \kappa)$$

A *weighted* MaxSAT problem [11], denoted ψ, is a MaxSAT problem where each clause $\kappa \in \psi$ has a bounded positive numerical weight $\omega(\kappa)$. The optimal solution $\mathcal{V}^*$ to some ψ satisfies the following equation:

$$\mathcal{V}^* = \arg\max_{\mathcal{V}} \sum_{\kappa \in \psi} \omega(\kappa)(\mathcal{V} \models \kappa)$$

Finally, the *partial* weighted MaxSAT problem [12] is a variant of weighted MaxSAT that distinguishes between *hard* and *soft* clauses. Only soft clauses are given a weight.

In these problems a solution is valid iff it satisfies all hard clauses. Therefore we have a notion of satisfiability. In particular, if the *hard* problem fragment of a partial weighted MaxSAT formula is unsatisfiable, then we say the formula is unsatisfiable. The definition of satisfiable follows naturally. An optimal solution to a partial weighted MaxSAT problem is an assignment $\mathcal{V}^*$ that is both valid and satisfies the above equation.

3   COS-P

We now describe COS-P, our planner that operates by iteratively solving variants of n-step-bounded instances of the problem at hand for successively larger n. Solutions to the intermediate step-bounded instances are obtained by compiling them into equivalent partial weighted MaxSAT problems, and then using our own MaxSAT procedure PWM-RSAT to compute their optimal solutions.

COS-P compiles and solves two variants, VARIANT-I and VARIANT-II, of the intermediate instances. Those are characterised in terms of their optimal solutions. Adopting the notation Π_n for the n-step-bounded variant of Π, VARIANT-I admits optimal solutions that correspond to minimal cost plans in the parallel format for Π_n. VARIANT-II admits optimal plans with the following structure. Each has a prefix which corresponds to n *sets* of actions from Π_n.[2] Plans can have an arbitrary length suffix (including length 0) comprised of actions from the delete relaxation Π^+.

Both variants can be categorised as *direct*, *constructive*, and *tightly sound*. They are *direct* because we have a Boolean variable in the MaxSAT problem for every action and state proposition at each plan step. They are *constructive* because any satisfying model and its cost in the MaxSAT instances corresponds to a plan and its cost in the source problem. Critically, our compilations are *tightly sound*, in the sense that every plan with cost c in the source planning problem has a corresponding satisfying model of cost c in the MaxSAT encoding and *vice versa*. This permits two key observations about VARIANT-I and VARIANT-II. First, when both variants yield an optimal solution, and both those solutions have identical cost, then the solution to VARIANT-I is a cost-optimal plan for Π. Second, if Π is soluble, then there exists some n for which the observation of global optimality shall be made by COS-P. Finally, we have that COS-P is a sound and complete optimal planning procedure for propositional problems with action costs.

For the remainder of this section we present the compilation for VARIANT-I and VARIANT-II. In the following section we describe the MaxSAT procedure PWM-RSAT that we developed for use by COS-P.

3.1   VARIANT-I**: bounded cost-optimal planning**

We now describe a direct compilation of the bounded propositional planning problem with action costs to a partial weighted MaxSAT formula ψ. The source of our compilation is the plangraph. This is an obvious choice because *reachability* and *neededness* analysis performed during construction of the plangraph yields important mutex constraints between action and propositional variables [13]. Such constraints are not deduced independently by modern SAT procedures such as RSAT2.02 [14].

---

[2] i.e., an n-step plan prefix in the parallel format.

Below, we develop our compilation in terms of a list of 6 Schemata. The first 5 schemata capture the *hard* logical planning constraints, and Schema 6 reflects the action costs. Overall, the schemata we develop below make use of the following propositional variables. For each action occurring at a step $t = 0, .., n-1$ (excluding *noop* actions), we have a variable a^t. We define a fluent to be a state proposition whose truth value can be modified by action executions. For each fluent occurring at step $t = 0, .., n$ we have a variable p^t. Also, we have $make(p) \equiv \{a | a \in \mathcal{A}, p \in \textit{eff}_\bullet(a)\}$, and $break(p) \equiv \{a | a \in \mathcal{A}, p \in \textit{eff}_\circ(a)\}$. Below we avoid annotating variables with their time index if it is clear from the context. Lastly, all constraints are hard unless stated otherwise.

1. Goal and start state axioms: We have a unit clause containing p^0 for every $p \in s_0$ and p^n for every $p \in \mathcal{G}$.

2. Precondition and effect axioms: For every action a at each plan step t, we have clauses that require: (i) the action implies its precondition, (ii) the action implies its positive effects, and (iii) the action implies its negative effects:
$$[a^t \to \bigwedge_{p \in pre(a)} p^t] \wedge [a^t \to \bigwedge_{p \in \textit{eff}_\bullet(a)} p^{t+1}] \wedge [a^t \to \bigwedge_{p \in \textit{eff}_\circ(a)} \neg p^{t+1}]$$

3. Mutex axioms: For every pair of mutex symbols (actions or fluents) p_1 and p_2 at step t, we have a clause: $\quad \neg p_1^t \wedge \neg p_2^t$

4. At least one action axioms: Where $\mathcal{A}^t$ is the set of actions at step t, we have a clause that requires at least one action be executed at step t: $\quad \bigvee_{a^t \in \mathcal{A}^t} a^t$

5. Frame axioms: These constrain how the truth values of fluents change over successive plan steps. For each proposition $p^t, t > 0$ we include the following clauses:
$$[p^t \to (p^{t-1} \vee \bigvee_{a \in make(p)} a^{t-1})] \wedge [\neg p^t \to (\neg p^{t-1} \vee \bigvee_{a \in break(p)} a^{t-1})]$$

6. Action cost axioms (soft): Finally, we have a set of soft constraints for actions. In particular, for each action variable a^t such that $\mathcal{C}(a) > 0$, we have a unit clause $\kappa_i := \{\neg a^t\}$ with weight $\omega(\kappa_i) = \mathcal{C}(a)$.

3.2 VARIANT-II: n-step with a relaxed suffix

We now describe a direct compilation of the problem Π_n from the previous section, along with the addition of a causal encoding of the delete relaxation, that we make available from step n.[3] From hereon we refer to the latter as the relaxed suffix.

Our encoding of the relaxed suffix is *causal* in the sense developed in [15] for their ground parallel *causal* encoding of propositional planning in SAT. This requires additional variables to those developed for VARIANT-I. In particular, for each fluent p and relaxed action $a \in \mathcal{A}^+$ we have corresponding variables p^+ and a^+. That p_i^+ is true intuitively means: (1) That p_i^n was false (see VARIANT-I), and (2) That $p_i \in \mathcal{G}$, or p_i^+ is the cause of another fluent p_j^+ in a relaxed suffix to the goal. That a^+ is true means that a is executed in the relaxed suffix. We also require a set of causal link variables. These are best introduced in terms of a recursively defined set S^∞ as follows.
$$S^0 \equiv \{\mathcal{K}(p_i, p_j) | a \in \mathcal{A}^+, p_i \in pre(a), p_j \in \textit{eff}_\bullet(a_i)\}$$
$$S^{i+1} \equiv S^i \cup \{\mathcal{K}(p_j, p_l) | \mathcal{K}(p_j, p_k), \mathcal{K}(p_k, p_l) \in S^i\}$$
For each $\mathcal{K}(p_i, p_j) \in S^\infty$ we have a corresponding variable. Intuitively, if proposition $\mathcal{K}(p_i, p_j)$ is true then p_i is the cause of p_j in the plan suffix.

VARIANT-II includes all schemata from VARIANT-I except the *goal axioms* of Schema 1. We also suppose Schema 6 is now inclusive of a^+ symbols. Additionally we have the following Schemata.

---

[3] In VARIANT-II goal constraints from Schema 1 are omitted from Π_n.

7. Relaxed goal axioms: For each fluent $p \in \mathcal{G}$ we assert that it is either achieved at the planning horizon n, or using a relaxed action in $\mathcal{A}^+$. This is expressed with a clause:
$$p^n \vee p^+$$
8. Relaxed fluent support axioms: For each fluent p we have a clause:
$$p^+ \to (\bigvee_{a \in make(p)} a^+)$$
9. Causal link axioms: For all fluents p_i, taking all $a \in make(p_i)$ and $p_j \in \text{PRE}(a)$, we have the following clause: $\quad (p_i^+ \wedge a^+) \to (p_j^n \vee \mathcal{K}(p_j^+, p_i^+))$
This constraint asserts that if action a_1^+ is executed, then its preconditions must be true at horizon n, or be supported by some other action a_2^+ with $p_2 \in \textit{eff}_\bullet(a_2)$.

10. Causality implies cause and effect axiom: For each causal link variable $\mathcal{K}(p_1^+, p_2^+)$ we have a clause: $\quad \mathcal{K}(p_1^+, p_2^+) \to (p_1^+ \wedge p_2^+)$

11. Transitive closure and anti-reflexivity axioms: For causal link variable $\mathcal{K}(p^+, p^+)$ we have a unit clause containing that variable negated. For pairs of causal link variables $(\mathcal{K}(p_1^+, p_2^+), \mathcal{K}(p_2^+, p_3^+))$: $\quad (\mathcal{K}(p_1^+, p_2^+) \wedge \mathcal{K}(p_2^+, p_3^+)) \to \mathcal{K}(p_1^+, p_3^+)$

12. Only necessary relaxed fluent axioms: For each fluent p we have a constraint:
$$\neg p^+ \vee \neg p^n$$
13. Relaxed action cost dominance axioms: Let $\overrightarrow{P}$ be a set of non-mutex fluents at horizon n. Relaxed action a_1^+ is *redundant* in an optimal solution to a VARIANT-II instance, if the fluents in $\overrightarrow{P}$ are true at horizon n and there exists a relaxed action a_2^+ such that: (1) $cost(a_2) \leq cost(a_1)$, (2) $pre(a_2) \backslash \overrightarrow{P} \subseteq pre(a_1) \backslash \overrightarrow{P}$, and (3) $\textit{eff}_\bullet(a_1) \backslash \overrightarrow{P} \subseteq \textit{eff}_\bullet(a_2) \backslash \overrightarrow{P}$. For relaxed action a^+ that is *redundant* for $\overrightarrow{P_1}$ and not redundant for any $\overrightarrow{P_2}$, if $|\overrightarrow{P_2}| < |\overrightarrow{P_1}|$ we have a clause:[4] $\quad (\bigwedge_{p \in \overrightarrow{P_1}} p^n) \to \neg a^+$

The schemata we have given thus far are theoretically sufficient for our purpose. However, in a relaxed suffix most causal links are not relevant to the relaxed cost of reaching the goal from a particular state at horizon n. For example, in a logistics problem, if a truck t at location l_1 needs to be moved directly to location l_2, then the fact that the truck is at any other location should not support it being at l_2 – i.e. $\neg \mathcal{K}(\text{at}(t, l_3), \text{at}(t, l_2)), l_3 \neq l_1$.

The following schemata provide a number of *layers* that actions and fluents in the relaxed suffix can be assigned to. Fluents and actions are forced to occur as early in the set of layers as possible and are only assigned to a layer if all supporting actions and fluents occur at earlier layers. The orderings of fluents in the relaxed layers is used to restrict the truth values of the causal link variables. The admissibility of the heuristic estimate of the relaxed suffix is independent of the number of relaxed layers.

We pick an horizon $k > n$ and generate a copy a^{+l} of each relaxed action a^+ at each layer $l \in \{n, ..., k-1\}$ and a copy p^{+l} of each fluent p^+ at each layer $l \in \{n+1, ..., k\}$. We also have an auxiliary variable $aux(p^{+l})$ for each fluent p^{+l} at each suffix layer $n+1, ..., k$. Intuitively, proposition $aux(p^{+l})$ says that p is false at every layer in the relaxed suffix from n to l.[5]

14. Layered relaxed action axioms: For each layered relaxed action a^{+l} we have a clause: $\quad a^{+l} \to a^+$

15. Layered relaxed actions only once axioms: For each relaxed action a^+ and pair of layers $l_1, l_2 \in \{n, ..., k-1\}$, where $l_1 \neq l_2$, we have: $\quad \neg a^{+l_1} \vee \neg a^{+l_2}$

---

[4] In practise we limit $|\overrightarrow{P_1}|$ to 2.

[5] There are no cost constraints associated with the layered copies of relaxed action variables.

16. Layered relaxed action precondition axioms: For each layered relaxed action a^{+l_1} we have a set of clauses: $\quad a^{+l_1} \rightarrow \bigwedge_{p \in \mathrm{PRE}(a)} \bigvee_{l_2 \in \{n, ..., l_1\}} p^{+l_2}$

17. Layered relaxed action effect axioms: For each layered relaxed action a^{+l_1} and $p \in \mathrm{ADD}(a)$ there is a clause: $\quad (a^{+l_1} \wedge p^+) \rightarrow \bigvee_{l_2 \in n+1, ..., l+1} p^{+l_2}$

18. Layered relaxed action as early as possible axioms: For each layered relaxed action a^{+l_1}, if $l_1 = n$, we have a clause: $\quad a^+ \rightarrow \bigvee_{p \in \mathrm{PRE}(a)} \neg p^n \ \vee a^{+n}$

if $l_1 > n$, we add: $\quad a^+ \rightarrow \bigvee_{l_2 \in n, ..., l_1 - 1} a^{+l_2} \vee \bigvee_{p \in \mathrm{PRE}(a)} aux(p^{+l_1}) \vee a^{+l_1}$

19. Auxiliary variable axioms: For each auxiliary variable $aux(p^{+l_1})$ there is a set of clauses: $\quad aux(p^{+l_1}) \longleftrightarrow (p^n \wedge \bigwedge_{l_2 \in \{n+1, ..., l_1\}} \neg p^{+l_2})$

20. Layered fluent axioms: For each layered fluent p^{+l} we add: $\quad p^{+l} \rightarrow p^+$

21. Layered fluent frame axioms: For each layered fluent p^{+l} there is a clause:
$$p^{+l} \rightarrow \bigvee_{a \in make(p)} a^{+l-1}$$

22. Layered fluent as early as possible axioms: For each layered fluent p^{+l_1} there is a set of clauses: $\quad p^{+l_1} \rightarrow \bigwedge_{a \in make(p)} \bigwedge_{l_2 \in n, ..., l_1 - 2} \neg a^{+l_2}$

23. Layered fluent only once axioms: For each fluent p and pair of layers $l_1, l_2 \in \{n+1, ..., k\}$, where $l_1 \neq l_2$, there is a clause: $\quad \neg p^{+l_1} \vee \neg p^{+l_2}$

24. Layered fluents prohibit causal links axioms: For each layered fluent $p_1^{+l_1}$ and fluent p_2 such that $p_1 \neq p_2$ and $\exists \mathcal{K}(p_2^+, p_1^+)$ there is a clause:
$$p_1^{+l_1} \rightarrow (\bigvee_{l_2 \in \{n+1, ..., l-1\}} p_2^{+l_2} \vee \neg \mathcal{K}(p_2^+, p_1^+))$$

4   PWM-RSat

We find that branch-and-bound procedures for *partial weighted MaxSAT* [11, 12] are ineffective at solving our direct encodings of bounded planning problems. Thus, taking the RSAT2.02 codebase as a starting point, we developed PWM-RSAT, a more efficient optimisation procedure for this setting. An outline of the algorithm is given in Algorithm 1. Based on RSAT [16], PWM-RSAT can broadly be described as a backtracking search with Boolean unit propagation. It features common enhancements from state-of-the-art SAT solvers, including conflict driven *clause learning* with *non-chronological* backtracking [17, 18], and *restarts* [19].

Algorithm 1 outlines two variants of PWM-RSAT for solving VARIANT-I and VARIANT-II formulas: lines 5-6 will only be invoked if the input formula is a VARIANT-II encoding. These lines prevent the solver from exploring assignments implying that the same state occurs at more than one planning layer.

Apart from the above difference, the two variants of PWM-RSAT work as follows. At the beginning of the search, the current partial assignment $\mathcal{V}$ of truth values to variables in ψ is set to empty and its associated cost c is set to 0. We use $\hat{c}$ to track the best result found so far for the minimum cost of satisfying ψ^∞ given ψ^+. $\mathcal{V}^*$ is the total assignment associated with $\hat{c}$. Initially, $\mathcal{V}^*$ is empty and $\hat{c}$ is set to an input non-negative weight bound $\hat{c}^I$ (if none is known then $\hat{c} = \hat{c}^I := \infty$). Note that the set of *asserting clauses* Γ is initiated to empty as no clauses have been learnt yet.

The solver then repeatedly tries to expand the partial assignment $\mathcal{V}$ until either the optimal solution is found or ψ is proved unsatisfiable (line 4-21). At each iteration, a call to *SatUP*$(\mathcal{V}, \psi, \kappa)$ applies unit propagation to a unit clause $\kappa \in \psi$ and adds new variable assignments to $\mathcal{V}$. If κ is not a unit clause, *SatUP*$(\mathcal{V}, \psi, \kappa)$ returns 1 if κ is

---

Algorithm 1 Cost-Optimal RSat —- PWM-RSAT

1: Input:
   - A given non-negative weight bound $\hat{c}^I$. If none is known: $\hat{c}^I := \infty$
   - A CNF formula ψ consists of the *hard* clause set ψ^∞ and the *soft* clause set ψ^+
2: $c \leftarrow 0; \hat{c} \leftarrow \hat{c}^I$;
3: $\mathcal{V}, \mathcal{V}^* \leftarrow []; \Gamma \leftarrow \emptyset$;
4: **while** true **do**
5:     **if** solving Variant-II && *duplicating-layers($\mathcal{V}$)* **then**
6:         pop elements from $\mathcal{V}$ until ¬*duplicating-layers($\mathcal{V}$)*; *continue*;
7:     $c \leftarrow \sum_{\kappa \in \psi+} \omega(\kappa) SatUP(\mathcal{V}, \psi, \kappa)$;
8:     **if** $c \geq \hat{c}$ **then**
9:         pop elements from $\mathcal{V}$ until $c < \hat{c}$; *continue*;
10:    **if** $\exists \kappa \in (\psi^\infty \wedge \Gamma)$ s.t. ¬$SatUP(\mathcal{V}, \psi^\infty \wedge \Gamma, \kappa)$ **then**
11:        **if** restart **then** $\mathcal{V} \leftarrow []$; *continue*;
12:        learn clause with *assertion level m*; add it to Γ;
13:        pop elements from $\mathcal{V}$ until $|\mathcal{V}| = m$;
14:        **if** $\mathcal{V} = []$ **then**
15:            **if** $\mathcal{V}^* \neq []$ **then return** $\langle \mathcal{V}^*, \hat{c} \rangle$ as the solution;
16:            **else return** UNSATISFIABLE;
17:    **else**
18:        **if** $\mathcal{V}$ is total **then**
19:            $\mathcal{V}^* \leftarrow \mathcal{V}; \hat{c} \leftarrow c$;
20:            pop elements from $\mathcal{V}$ until $c < \hat{c}$;
21:        add a new variable assignment to $\mathcal{V}$;

---

satisfied by $\mathcal{V}$, and 0 otherwise. The current cost c is also updated (line 7). If $c \geq \hat{c}$, then the solver will perform a backtrack-by-cost to a previous point where $c < \hat{c}$ (line 8-9).

During the search, if the current assignment $\mathcal{V}$ violates any clause in $(\psi^\infty \wedge \Gamma)$, then the solver will either (i) restart if required (line 11), or (ii) try to learn the conflict (line 12) and then backtrack (line 13). If the backtracking causes all assignments in $\mathcal{V}$ to be undone, then the solver has successfully proved that either (i) $(\mathcal{V}^*, \hat{c})$ is the optimal solution, or (ii) ψ is unsatisfiable if $\mathcal{V}^*$ remains empty (line 14-16). Otherwise, if $\mathcal{V}$ does not violate any clause in $(\psi^\infty \wedge \Gamma)$ (line 17), then the solver will heuristically add a new variable assignment to $\mathcal{V}$ (line 21) and repeat the loop in line 4. Note that if $\mathcal{V}$ is already complete, the better solution is stored in $\mathcal{V}^*$ together will the new lower cost $\hat{c}$ (line 19). The solver also performs a backtrack by cost (line 20) before trying to expand $\mathcal{V}$ in line 21.

5   Experimental Results

We implemented both COS-P and PWM-RSAT in C++. We now discuss our experimental comparison of COS-P with IPC baseline planner BASELINE,[6] and a version of COS-P called H-ORACLE. The latter is given (by an oracle) the shortest horizon that yields a globally optimal plan. Planning benchmarks included in our evaluation include: IPC-6: ELEVATORS, PEG SOITAIRE, and TRANSPORT; IPC-5: STORAGE, and TPP; IPC-3: DEPOTS, DRIVERLOG, FREECELL, ROVERS, SATELLITE, and ZENOTRAVEL; and IPC-1: BLOCKS, GRIPPER, and MICONIC. We also developed our own domain, called FTB, that demonstrates the effectiveness of the factored problem representations employed by SAT-based systems such as COS-P. This domain has the following important properties: (1) it has exponentially many states in the number of problem objects, (2) if there are n objects, then the branching factor is such that a breadth-first search

---

[6] The *de facto* winning entry at the last IPC.

encounters all the states at depth n, and (3) all plans have length n, and plan optimality is determined by the first and last actions (only) of the plan. This domain cripples state-based systems such as HSP, BASELINE, and GAMER, either because they are doing a non-factored forward heuristic search, or because —i.e., in the case of GAMER and BASELINE— they perform a breadth-first search. Finally, experiments were run on a cluster of AMD Opteron 252 2.6GHz processors, each with 2GB of RAM. All plans computed by COS-P, H-ORACLE, and BASELINE were verified by the Strathclyde Planning Group plan verifier VAL, and computed within a timeout of 30 minutes.

The results of our experiments are summarised in Table 1. For each domain there is one row for the hardest problem instance solved by each of the three planners. Here, we measure problem hardness as the time it takes each solver to return the optimal plan. In some domains we also include additional instances. Using the same experimental data as for Table 1, Figure 1 plots the cumulative number of instances solved over time by each planning system, supposing invocations of the systems on problem instances are made in parallel. It is important to note that the size of the CNF encodings required by COS-P (and H-ORACLE) are not prohibitively large – i.e, where the SAT-based approaches fail, this is typically because they exceed the 30 minutes timeout, and not because they exhaust system memory.
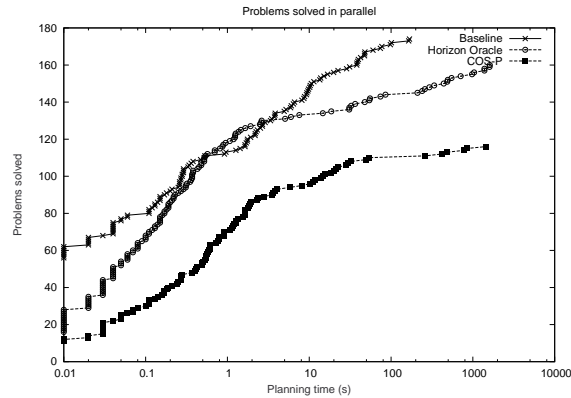

Fig. 1. The number of problems solved in parallel after a given planning time for each approach.

COS-P outperforms the BASELINE in the BLOCKS and FTB domains. For example, on BLOCKS-18 BASELINE takes 39.15 seconds while COS-P takes only 3.47 seconds. In other domains BASELINE outperforms COS-P, sometimes by several orders of magnitude. For example, on problem ZENOTRAVEL-4 BASELINE takes 0.04 seconds while COS-P takes 841.2. More importantly, we discovered that it is relatively easy to find a cost-optimal solution compared to proving its optimality. For example, on MICONIC-23 COS-P took 0.53 seconds to find the optimal plan but spent 1453 seconds proving cost-optimality. More generally, this observation is indicated by the performance of H-ORACLE.

Overall, we find that clause learning procedures in PWM-RSAT cannot exploit the presence of the *very* effective delete relaxation heuristic from Π^+. Consequently, a serious bottleneck of our approach comes from the time required to solve VARIANT-II

Table 1. C^* is the optimal cost for each problem. All times are in seconds. For BASELINE t is the solution time. For H-ORACLE, n is the horizon returned by the oracle and t is the time taken to find the lowest cost plan at n. For COS-P, t_t is the total time for all SAT instances, t_π is the total time for all SAT instances where the system was searching for a plan, while t_* is the total time for all SAT instances where the system is performing optimality proofs. '-' indicates that a solver either timed out or ran out of memory.

		BASELINE	H-ORACLE		COS-P			
Problem	C^*	t	n	t	n	t_t	t_π	t_*
blocks-17	28	39.83	28	0.59	28	3.61	3.61	0
blocks-18	26	39.15	26	0.53	26	3.47	3.47	0
blocks-23	30	-	30	4.61	30	32.11	32.11	0
blocks-25	34	-	34	3.43	34	29.49	29.49	0
depots-7	21	98.08	11	64.79	-	-	-	-
driverlog-3	12	0.11	7	0.043	7	484.8	0.08	484.7
driverlog-6	11	9.25	5	0.046	-	-	-	-
driverlog-7	13	100.9	7	1.26	-	-	-	-
elevators-2	26	0.33	3	0.01	3	14	0.01	13.99
elevators-5	55	167.9	-	-	-	-	-	-
elevators-13	59	28.59	10	378.6	-	-	-	-
freecell-4	26	47.36	-	-	-	-	-	-
ftb-17	401	38.28	17	0.08	17	0.27	0.09	0.18
ftb-30	1001	-	25	0.7	25	1.95	0.7	1.24
ftb-38	601	-	33	0.48	33	1.65	0.49	1.15
ftb-39	801	-	33	0.7	33	2.35	0.67	1.69
gripper-1	11	0	7	0.02	7	15.7	0.14	15.56
gripper-3	23	0.05	15	34.23	-	-	-	-
gripper-7	47	73.95	-	-	-	-	-	-
miconic-17	13	0	11	0.07	11	785.4	0.30	785.1
miconic-23	15	0.04	10	0.12	10	1454	0.53	1453
miconic-33	22	2.19	17	2.17	-	-	-	-
miconic-36	27	9.62	22	1754	-	-	-	-
miconic-39	28	10.61	24	484.1	-	-	-	-
pegsol-7	3	0	12	0.08	12	1.63	0.23	1.41
pegsol-9	5	0.02	15	7.07	15	416.6	12.25	404.4
pegsol-13	9	0.14	21	1025	-	-	-	-
pegsol-26	9	42.44	-	-	-	-	-	-
rovers-3	11	0.02	8	0.1	8	53.21	0.08	53.13
rovers-5	22	164.1	8	69.83	-	-	-	-
satellite-1	9	0	8	0.08	8	0.92	0.1	0.82
satellite-2	13	0.01	12	0.23	-	-	-	-
satellite-4	17	6.61	-	-	-	-	-	-
storage-7	14	0	14	0.45	14	1.16	1.16	0
storage-9	11	0.2	9	643.2	-	-	-	-
storage-13	18	3.47	18	112.1	18	262.8	262.8	0
storage-14	19	60.19	-	-	-	-	-	-
TPP-5	19	0.15	7	0.01	-	-	-	-
transport-1	54	0	5	0.02	5	0.27	0.03	0.24
transport-4	318	47.47	-	-	-	-	-	-
transport-23	630	0.92	9	1.28	-	-	-	-
zenotravel-4	8	0.04	7	1.07	7	843.7	2.47	841.2
zenotravel-6	11	8.77	7	54.35	-	-	-	-
zenotravel-7	15	5.21	8	1600	-	-	-	-

instances. On a positive note, those proofs are possible, and in domains such as BLOCKS and FTB, where the branching factor is high and useful plans long, the factored problem representations and corresponding solution procedures in the SAT-based setting payoff. Moreover, in fixed-horizon cost-optimal planning, the SAT approach continues to show good performance characteristics in many domains.

6 Concluding Remarks

In this paper we demonstrate that a general theorem-proving technique, particularly a DPLL procedure for Boolean SAT, can be modified to find cost-optimal solutions to propositional planning problems encoded as SAT.[7] In particular, we modified SAT solver RSAT2.02 to create PWM-RSAT, an effective partial weighted MaxSAT procedure for problems where all *soft* constraints are unit clauses. This forms the underlying optimisation procedure in COS-P, our cost-optimal planning system that, for successive horizon lengths, uses PWM-RSAT to establish a candidate solution at that horizon, and then to determine if that candidate is globally optimal. Each candidate is a minimal cost step-bounded plan for the problem at hand. That a candidate is globally optimal is known if no step-bounded plan with a relaxed suffix has lower cost. To achieve that, we developed a MaxSAT encoding of bounded planning problems with a relaxed suffix. This constitutes the first application of causal representations of planning in propositional logic [15].

Existing work directly related to COS-P includes the hybrid solver CO-PLAN [20] and the fixed-horizon optimal system PLAN-A. Those systems placed 4th and last respectively out of 10 systems at IPC-6. CO-PLAN is hybrid in the sense that it proceeds in two phases, each of which applies a different search technique. The first phase is SAT-based, and identifies the least costly step-optimal plan. PLAN-A also performs that computation, however assumes that a least cost step-optimal plan is globally optimal – Therefore PLAN-A was not competitive because it could not find globally optimal solutions, and thus forfeited in many domains. The first phase of CO-PLAN and the PLAN-A system can be seen as more general and efficient versions of the system described in [21]. The second phase of CO-PLAN breaks from the planning-as-SAT paradigm. It corresponds to a cost-bounded anytime best-first search. The cost bound for the second phase is provided by the first phase. Although competitive with a number of other competition entries, CO-PLAN is not competitive in IPC-6 competition benchmarks with the BASELINE – The *de facto* winning entry, a brute-force A^* in which the distance-plus-cost computation always takes the distance to be zero.

Other work related to COS-P leverages SAT modulo theory (SMT) procedures to solve problems with metric resource constraints [22]. SMT-solvers typically interleave calls to a *simplex* algorithm with the *decision steps* of a backtracking search, such as DPLL. Solvers in this category include the systems LPSAT [22], TM-LPSAT [23], and NUMREACH/SMT [1]. SMT-based planners also operate according to the BLACKBOX scheme, posing a series of step-bounded decision problems to an SMT solver until an optimal plan is achieved. Because they are not globally optimal, existing SMT systems are not directly comparable to COS-P.

The most pressing item for future work is a technique to exploit SMT —and/or branch-and-bound procedures from weighted MaxSAT— in proving the optimality of candidate solutions that PWM-RSAT yields in bounded instances. We should also exploit recent work in using useful admissible heuristics for state-based search when evaluating whether horizon n yields an optimal solution [24].

---

[7] This was supposed to be possible, although in a very impractical sense (final remarks of [4]).

References

1. Hoffmann, J., Gomes, C.P., Selman, B., Kautz, H.A.: Sat encodings of state-space reachability problems in numeric domains. In: Proc. IJCAI. (2007)
2. Kautz, H.A.: Deconstructing planning as satisfiability. In: *Proc.* AAAI. (2006)
3. Russell, R., Holden, S.: Handling goal utility dependencies in a satisfiability framework. In: Proc. ICAPS. (2010)
4. Giunchiglia, E., Maratea, M.: Planning as satisfiability with preferences. In: Proc. ICAPS. (2007)
5. Robinson, N., Gretton, C., Pham, D.N., Sattar, A.: Sat-based parallel planning using a split representation of actions. In: Proc. ICAPS. (2009)
6. Streeter, M., Smith, S.: Using decision procedures efficiently for optimization. In: Proc. ICAPS. (2007)
7. Rintanen, J.: Evaluation strategies for planning as satisfiability. In: *Proc.* ECAI. (2004)
8. Kautz, H., Selman, B.: Unifying SAT-based and graph-based planning. In: *Proc.* IJCAI. (1999)
9. Keyder, E., Geffner, H.: Soft goals can be compiled away. Journal of Artificial Intelligence Research **36**(1) (2009)
10. Bylander, T.: The computational complexity of propositional strips planning. Artificial Intelligence **69** (1994) 165–204
11. Josep Argelic and, C.M.L., Manya, F., Planes, J.: The first and second max-sat evaluations. Journal on Satisfiability, Boolean Modeling and Computation **4** (2008) 251–278
12. Fu, Z., Malik, S.: On solving the partial max-sat problem. In: SAT 2006. (August 2006) 252–265
13. Blum, A., Furst, M.: Fast planning through planning graph analysis. Artificial Intelligence (90) (1997) 281–300
14. Rintanen, J.: Planning graphs and propositional clause learning. In: *Proc.* KR. (2008)
15. Kautz, H., McAllester, D., Selman, B.: Encoding plans in propositional logic. In: *Proc.* KR. (1996)
16. Pipatsrisawat, K., Darwiche, A.: Rsat 2.0: SAT solver description. Technical Report D–153, Automated Reasoning Group, Computer Science Department, UCLA (2007)
17. Moskewicz, M.W., Madigan, C.F., Zhao, Y., Zhang, L., Malik, S.: Chaff: Engineering an Efficient SAT Solver. In: Proc. DAC. (2001)
18. Marques-Silva, J.P., Sakallah, K.A.: Grasp - a new search algorithm for satisfiability. In: Proc. ICCAD. (1996)
19. Huang, J.: The effect of restarts on the efficiency of clause learning. In: Proc. IJCAI. (2007)
20. Robinson, N., Gretton, C., Pham, D.N.: Co-plan: Combining SAT-based planning with forward-search. In: Proc. IPC-6. (2008)
21. Büttner, M., Rintanen, J.: Satisfiability planning with constraints on the number of actions. In: Proc. ICAPS. (2005)
22. Wolfman, S.A., Weld, D.S.: The LPSAT engine and its application to resource planning. In: Proc. IJCAI. (1999)
23. Shin, J.A., Davis, E.: Processes and continuous change in a sat-based planner. Artif. Intell. **166**(1-2) (2005) 194–253
24. Helmert, M., Domshlak, C.: Landmarks, critical paths and abstractions: What's the difference anyway? In: Proc. ICAPS. (2009)